



## **FAULT TOLERANT OVERLAY NETWORKS DESIGN**

**RAAD SADI AZIZ AL-AGELE**

**JULY 2015**

**FAULT TOLERANT OVERLAY NETWORKS DESIGN**

**A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED  
SCIENCES OF  
ÇANKAYA UNIVERSITY**

**BY  
RAAD SADI AZIZ AL-AGELE**

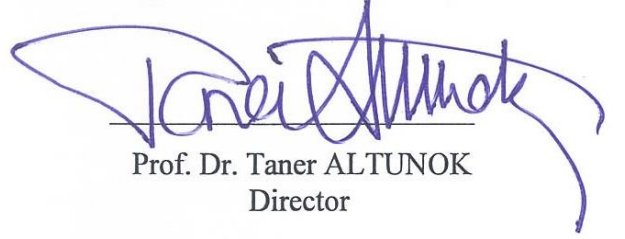
**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF  
MASTER OF SCIENCE  
IN  
THE DEPARTMENT OF  
MATHEMATICS AND COMPUTER SCIENCE  
INFORMATION TECHNOLOGY PROGRAM**

**JULY 2015**

Title of the Thesis : **Fault Tolerant Overlay Networks Design**

Submitted by **Raad Sadi Aziz AL-AGELE**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.



Prof. Dr. Taner ALTUNOK  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.



Prof. Dr. Billur KAYMAKÇALAN  
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.



Assist. Prof. Dr. Melih ONUŞ  
Supervisor

**Examination Date: 22.07.2015**

**Examining Committee Members:**

|                                     |                 |                                                                                       |
|-------------------------------------|-----------------|---------------------------------------------------------------------------------------|
| Assist. Prof. Dr. Melih ONUŞ        | (Çankaya Univ.) |  |
| Assist. Prof. Dr. Abdül Kadir GÖRÜR | (Çankaya Univ.) |  |
| Assoc. Prof. Dr. Fahd JARAD         | (UTAA)          |  |

### STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Raad Sadi Aziz, AL-AGELE

Signature : 

Date : 22.07.2015

## **ABSTRACT**

### **FAULT TOLERANT OVERLAY NETWORKS DESIGN**

AL-AGELE, Raad Sadi Aziz

M.Sc. Department of Mathematics and Computer Science / Information Technology  
Program

Supervisor: Assist. Prof. Dr. Melih ONUŞ

July 2015, 39 pages

In this thesis designs a reliable and scalable overlay network with fault-tolerance incorporation to support topic-based publish/subscribe communication. For scalability and efficiency, it is important to keep the degree of the nodes in the publish/subscribe system low. We proposes a new optimization problem named Fault-Tolerant Overlay Networks Design, where the trade-offs among several key dimensions such as fault tolerance, scalability, performance, and message dissemination are captured by it. The Fault-Tolerant Overlay Networks design problem is: given a set of nodes and their topic subscriptions connect the nodes to create a topic 2-connected overlay for pub/sub systems with minimum maximum degree, i.e., for each topic the sub-overlay induced by nodes interested in the topic is 2- connected. It presents an algorithm, namely GM3 for this problem which guarantees that the overlay network will be topic 2-connected and which aims at keeping the maximum node degree low.

Experimental results show that GM3 algorithm is able to achieve low maximum node degree of publish/subscribe overlay systems.

**Keywords:** Fault Tolerant Overlay Networks, Publish / Subscribe Systems, Topic 2-Connected Overlay, Low Maximum Node Degree.

## ÖZ

### **HATA TOLERANSLI BAŞKA BİR AĞIN ÜSTÜNE BİNA EDİLEN AĞ TASARIMI**

AL-AGELE, Raad Sadi Aziz

Yüksek Mühendis, Matematik ve Bilgisayar Bilimleri Bölümü / Bilgi Teknolojisi  
Programı

Danışman: Asist. Prof. Dr. Melih ONUŞ

Temmuz 2015, 39 sayfa

Bu tezde, konuya göre yayınlama/abone olma iletişimini destekleyecek, hatadan etkilenmez bir bileşime sahip güvenilir ve ölçeklendirilebilir bir başka bir ağın üstünde inşa edilecek ağ tasarlanmaktadır. Ölçeklendirilebilirlik ve verimlilik için, yayınlama/abone olma sistemindeki düğümlerin derecesini düşük seviyede tutmak önemlidir. Hata toleransı, ölçeklendirilebilirlik, performans ve mesaj yayılımı gibi birkaç temel boyut arasındaki ödünleşimlerin (değiş tokuşların) bunun vasıtasıyla yapıldığı Hata Toleranslı Başka bir Ağ üzerine inşa edilen Ağların Tasarımı adındaki yeni bir optimizasyon problemi gösterilmektedir. Bu Hata Toleranslı Yer Paylaşımlı Ağların tasarım problem şudur: verilen bir dizi ağ ve onların konu aboneliği, minimum maksimum derecesi ile yayınlama/abone olma sistemi için 2. Konu ile bağlantılı bir yer paylaşımı oluşturmak için ağları bağlar, ör: her konu için, bu konu ile ilgili olan ağlar tarafından uyarılan alt-yer paylaşımı, 2.si ile bağlantılıdır. Yer paylaşım ağının 2. Konuya bağlanmış olacağını temin eden ve maksimum

ağseviyesini düşük seviyede tutmayı amaçlayan bu problem için GM3 olarak adlandırılan bir algoritma sunmaktayız.

Deney sonuçlarımız da, algoritmamızın yer paylaşımı yayınlama/abone olma sistemlerinin maksimum ağseviyesini düşük seviyede tutabildiğini göstermektedir.

**Anahtar Kelimeler:** Hata Toleranslı Yer Paylaşım Ağları, Yayınlama / Abonelik Sistemleri, 2. Konuya bağlı Yer paylaşımı, Düşük Maksimum Ağ Seviyesi.



## **ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude to Assist. Prof. Dr. Melih ONUŞ for his supervision, special guidance, suggestions, and encouragement through the development of this thesis.

It is a pleasure to express my special thanks to my family for their valuable support.

## TABLE OF CONTENTS

|                                  |      |
|----------------------------------|------|
| STATEMENT OF NON PLAGIARISM..... | iii  |
| ABSTRACT.....                    | iv   |
| ÖZ.....                          | vi   |
| ACKNOWLEDGEMENTS.....            | viii |
| TABLE OF CONTENTS.....           | ix   |
| LIST OF FIGURES.....             | xi   |
| LIST OF ABBREVIATIONS.....       | xiii |

### CHAPTERS:

|                                                                  |    |
|------------------------------------------------------------------|----|
| 1. INTRODUCTION.....                                             | 1  |
| 2. PEER TO PEER SYSTEM.....                                      | 4  |
| 2.1. Background.....                                             | 4  |
| 2.2. Overview.....                                               | 4  |
| 2.3. Basic Definition.....                                       | 6  |
| 2.4. Peer to Peer Classification.....                            | 6  |
| 2.5. Peer to Peer Applications.....                              | 8  |
| 2.6. Peer to Peer Characteristics.....                           | 10 |
| 2.7. Examples of well-known peer to peer systems.....            | 11 |
| 2.7.1. Napster.....                                              | 11 |
| 2.7.2. Gnutella.....                                             | 11 |
| 2.7.3. Free Net.....                                             | 12 |
| 2.7.4. Search for Extra-terrestrial Intelligence (SETI@home).... | 12 |
| 2.7.5. Groove.....                                               | 12 |
| 2.7.6. JXTA.....                                                 | 13 |
| 2.7.7. Chord.....                                                | 13 |
| 2.7.8. Content Addressable Network (CAN) .....                   | 13 |
| 2.7.9. Bit Torrent.....                                          | 14 |
| 2.7.9.1. Sharing by torrent.....                                 | 14 |
| 2.7.9.2. Upload and share torrent files.....                     | 14 |
| 3. PUBLISH/SUBSCRIBE SYSTEM.....                                 | 15 |
| 3.1. Existing Topic-based Publish/Subscribe Systems.....         | 16 |
| 3.2. Existing Content-based Publish/Subscribe Systems.....       | 19 |
| 4. PROBLEM FORMULATION.....                                      | 22 |
| 4.1. Preliminaries.....                                          | 22 |

|        |                                                                                     |    |
|--------|-------------------------------------------------------------------------------------|----|
| 4.2.   | Fault Tolerant Overlay Networks Design Problem and Greedy Merge (GM) Algorithm..... | 23 |
| 4.3.   | 2TCO Problem and GM2 Algorithm.....                                                 | 23 |
| 4.4.   | Fault Tolerant Overlay Networks Design Problem and Our Algorithm GM3.....           | 25 |
| 4.4.1. | Example of GM3 Algorithm.....                                                       | 26 |
| 5.     | EXPERIMENTAL RESULTS.....                                                           | 33 |
| 5.1.   | Maximum Node Degree.....                                                            | 33 |
| 5.2.   | Average Node Degree.....                                                            | 35 |
| 5.3.   | Subscription Size.....                                                              | 36 |
| 6.     | CONCLUSION.....                                                                     | 38 |
|        | REFERENCES.....                                                                     | R1 |
|        | APPENDICES.....                                                                     | A1 |
| A.     | CURRICULUM VITAE.....                                                               | A1 |

## LIST OF FIGURES

### FIGURES

|                  |                                                                                                                                         |    |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------|----|
| <b>Figure 1</b>  | Types of computer system.....                                                                                                           | 5  |
| <b>Figure 2</b>  | Client-Server Model vs. different Peer to Peer Models .....                                                                             | 8  |
| <b>Figure 3</b>  | Basic Pub-Sub System.....                                                                                                               | 16 |
| <b>Figure 4a</b> | Example of GM3 algorithm step1 nodes without any edge.....                                                                              | 26 |
| <b>Figure 4b</b> | Example of GM3 algorithm step 2 add edge 1 with max intersection. ....                                                                  | 27 |
| <b>Figure 4c</b> | Example of GM3 algorithm step 3 add edge 2 with max intersection. ....                                                                  | 27 |
| <b>Figure 4d</b> | Example of GM3 algorithm step 4 add edge 3 with max intersection. ....                                                                  | 28 |
| <b>Figure 4e</b> | Example of GM3 algorithm step 5 add edge 4 with max intersection. ....                                                                  | 28 |
| <b>Figure 4f</b> | Example of GM3 algorithm step 6 add edge 5 with max intersection. ....                                                                  | 29 |
| <b>Figure 4g</b> | Example of GM3 algorithm step 7 add edge 6 with max intersection .....                                                                  | 29 |
| <b>Figure 4h</b> | Example of GM3 algorithm step 8 add edge 7 with maximum weight .....                                                                    | 30 |
| <b>Figure 4i</b> | Example of GM3 algorithm step 9 add edge 8 with maximum weight .....                                                                    | 30 |
| <b>Figure 4j</b> | Example of GM3 algorithm step 10 add edge 9 with maximum weight .....                                                                   | 31 |
| <b>Figure 4k</b> | Example of GM3 algorithm step 11 add edge 10 with maximum weight .....                                                                  | 31 |
| <b>Figure 4l</b> | Example of GM3 algorithm step 12 add edge 11 with maximum weight .....                                                                  | 32 |
| <b>Figure 4m</b> | Figure 4m Example of GM3 algorithm step13 add last edge with maximum weight there is a 2-connected component in each of the topics..... | 32 |
| <b>Figure 5</b>  | Maximum node degree for GM, GM2 and GM3 when number of topics is 100.....                                                               | 34 |
| <b>Figure 6</b>  | Maximum node degree for GM, GM2 and GM3 when number of topics is 200.....                                                               | 34 |
| <b>Figure 7</b>  | Average node degree for GM, GM2 and GM3 when the number of topics is 100 .....                                                          | 35 |
| <b>Figure 8</b>  | Average node degree for GM, GM2 and GM3 when the number of topics is 200.....                                                           | 36 |

|                  |                                                                                                               |    |
|------------------|---------------------------------------------------------------------------------------------------------------|----|
| <b>Figure 9</b>  | Maximum node degree for different subscription size (Number of nodes (200) and number of topics is 100) ..... | 37 |
| <b>Figure 10</b> | Average node degree for different subscription size (Number of nodes (200) and number of topics is 100).....  | 37 |

## LIST OF ABBREVIATIONS

|                   |                                                    |
|-------------------|----------------------------------------------------|
| <b>AOL</b>        | America Online                                     |
| <b>BOINC</b>      | Berkeley Open Infrastructure for Network Computing |
| <b>CAN</b>        | Content Addressable Network                        |
| <b>CONN.COMPS</b> | Content Addressable Network                        |
| <b>DHT</b>        | Distributed Hash Tables                            |
| <b>DOS</b>        | Denial of Service                                  |
| <b>FLOPS</b>      | Floating -Point Operations Per Second              |
| <b>FTP</b>        | File Transport Protocol                            |
| <b>GM</b>         | Greedy Merge                                       |
| <b>KBR</b>        | Key Based Routing                                  |
| <b>LAN</b>        | Local Area Network                                 |
| <b>MIN AV-TCO</b> | Minimum Average Degree Topic Connected Overlay     |
| <b>MSN</b>        | Microsoft Network                                  |
| <b>NNTP</b>       | Network News Transfer Protocol                     |
| <b>NP</b>         | Nondeterministic Polynomial time                   |
| <b>Pup / sub</b>  | Publish / Subscribe                                |
| <b>RSS</b>        | Radio Service Software                             |
| <b>SETI@home</b>  | Search for Extra-Terrestrial Intelligence          |
| <b>TC</b>         | Topic-Connected                                    |
| <b>TCO</b>        | Topic Connected Overlay                            |
| <b>TCP</b>        | Transmission Control Protocol                      |
| <b>TPSO</b>       | Topic-Based Pub/Sub Overlay Network                |
| <b>UUCP</b>       | Unix To Unix Copy Protocol                         |

## CHAPTER 1

### INTRODUCTION

Publish / subscribe allows to decouple the provider of some information with the consumers of the same information. Publishers publish their messages through logical channels and subscribers receive the messages they are interested in by subscribing to the appropriate services, which deliver messages through these channels.

A pub/sub system is called a topic-based, when the messages are published to “topics”, where each topic is uniquely linked with a logical channel. Publishers in topic-based system post messages to any message broker or queue, for topic based pub-sub model messages, belonging to a specified topic to which subscribers can subscribe to. Subscribers can subscribe for the messages published by the publisher system through topic based subscription by a message broker or a queue. The responsibility of publisher is to define the classes of messages to which subscribers can subscribe. In a content-based system, messages are delivered to a subscriber whose their defined constraints are match the attributes of those messages only, for each logical channel there is a subset of these attributes to characterize it. The subscriber is responsible for categorizing the messages.

Pub/sub communication systems provide the opportunity for better scalability and easy application (see e.g., [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]). In these systems there are many real-world applications that are built such as highlighted abundance of accepted applications on the Internet, such as stock market players monitor, RSS [13] games, on-line and many others .

In this thesis, we will design a (peer-to-peer) overlay network , in the sense that for each topic, the subgraph made by the nodes interested in topic will be 2- connected.

This system has completely decentralized topic based pub/sub system where will be connected to any network-based overlay on a given topic, therefore nodes subscribed in a specific topic does not need to rely on other nodes to direct their messages, such this overlay network named a topic 2-connected.

Topic-connected overlay (TCO) defined, as an overlay, where all nodes interested in the same topic are ordered in a connected distribution sub-overlay, Gregory Chockler et al[14]. A TCO guarantees that nodes not interested in a topic not need to contribute to distributing information on that topic. Publication routing TCOs keeps bandwidth and computational resources otherwise lost on forwarding messages of no interest to the node. The result of a simpler matching engine design, smaller forwarding tables and more efficient routing protocols is a topic-connectivity. From a security view, TCOs are wanted when messages are to be shared through a network among a set of relied users without leaving this set.

The complexity of pub/sub overlay network can be estimated by the cost of broadcast of the topic-based on the network. Like other systems, trade-off between the space and time exists in two opposed measures: One of them, the total time taken by the broadcast wanted to be as small as possible, the other one, is to save the total degree of nodes small for memory and node bandwidth considerations. For instance, if there are many of nodes subscribed to the same topic, the result is a star overlay with best possible diameter but the degree for the node is so much. It is not easy to achieve a balanced structure (e.g., a balanced binary tree) for each subject without increasing the node degrees as the nodes subscription sets sizes.

Some of practical solutions failed in keeping the reduction of both the diameter and the node degree. A cycle construction (or a tree or any other isolated overlay structure) is a simple and popular solution where all nodes which interested in a topic connected autonomously for each given topic [12]:

This structure may result in a network with node degrees proportional to the nodes' subscription sizes, whereas a more careful structure, taking into account the correlations among the node subscription sets might result in more smaller node



degrees and total number of edges.

Low node degree is useful for both of process practices and bandwidth restrictions. Nodes able to manage large numbers of its adjacent links. For example, monitor the availability of its neighbors, incurring in heartbeats and keep-alive state costs, and connection state costs in TCP and each of the links pass through the traffic, with less advantage token from totaling the traffic in spite of reducing the number of packet headers, which could be responsible for important part of the traffic for small messages [15].

Well-correlation among the subscriptions of node decreases the node degrees and number of edges required by a topic 2-connected overlay network. To achieve that connect two nodes which have many same topic subscription, in this case just one edge fulfill connectivity of many topics for those two nodes. The suggestion of several recent empirical studies is that correlated workloads are actually common in practice [13].

In this work, we first study the creating topic-based pub/sub overlay networks problem with low node degrees. The following problem that we consider exactly:

**Fault Tolerant Overlay Networks Design Problem:**

Given  $V$  as set of nodes, a set of topics  $T$ , and the node interest assignment  $I$ , connect the nodes in  $V$  into a topic 2-connected overlay network  $G$  with least possible maximum degree. We design an algorithm for this problem named GM3. We did simulations and compared our algorithm with previous algorithms.

## **CHAPTER 2**

### **PEER TO PEER SYSTEMS**

#### **2.1 Background**

Over the recent decades, a body of knowledge regarding technology has been developed. Running various technological applications has never been easier as it is in the modern era. The main aim of this chapter is to explain the technological development of peer to peer systems. This is particularly useful for this thesis.

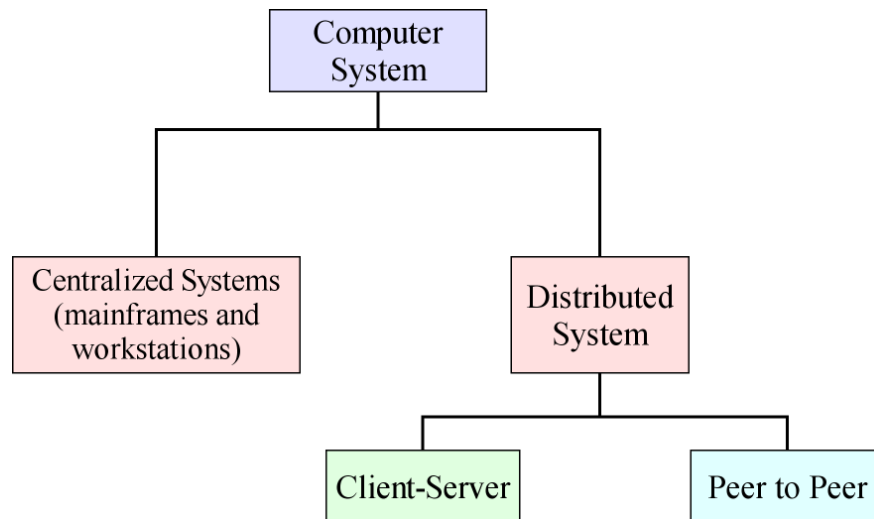
#### **2.2 Overview**

The Computer Systems are in many instances distributed across a wide range of applications. However, there are some which remain on a central position as illustrated in Figure 1.

The Distributed Systems fall into two classes of models. They include:

- (i) Client-Server Model.
- (ii) Peer to Peer Model.

Obtaining data is dependent on the protocol that dictates the communication policies with the client-server model. To access data, a client has to ensure that they access the main server. Some of the main servers include the web server or the FTP server. There are some issues that bedevil the client-server model. The challenges include fault tolerance and scalability. These challenges compel researchers to come up with a peer to peer mode as the other option.



**Figure 1** Types of computer system.

Figure 1 is a visual representation of the peer to peer system. According to the Figure, peer to peer system entails applications and systems that use distributed resources to conduct many of its activities especially those that should be decentralized. The resources entail the computing power, computers, data and the network bandwidth among others. All these are necessary in such functions. Peer to peer systems have existed for a long time. In fact, USENET is one of the ancient peer to peer systems that provide a news group service. USENET depends on Unix to Unix Copy Protocol (UUCP) when one needs to use the Unix machine to exchange files, dial another peer or send emails. In addition, it uses the Network News Transfer Protocol (NNTP). Through this, it can allow peers on Usenet network to spot a new newsgroup and share messages and files with the group. A peer to peer system is more advantageous when compared other centralized systems. Through the system, it is possible to bring together resources used by users and produce large bits of distributed information. The fact that it is possible for computers to communicate together makes it easy to use a network bandwidth without much ado. Many of the features of the peer to peer systems especially the file sharing application is popular among internet users. Peer to peer systems permeates through science and other academic fields. Hence, it is possible to use all the resources of a personal computer with ease. The peer to peer system allows internet users to embed devices, and accesses the real time data.

### **2.3 Basic Definition**

The peer to peer network that computers in the same network line use can perform dual functions both as the provider and the customer at ago. Hence, each of the devices in the network can request and still provide information from other applications and devices attached to the network. A peer-to-peer network refers to a local computer network LAN made up of many devices. These devices are equal and do not have any provider server in each of the network's devices. Such a network is also referred to as the workgroup. The work group refers to a number of devices that work together to perform a particular task. A work group is made up of at most ten devices. A peer to peer network is recommended for small networks and users who perform identical tasks. They are usually found in offices that use computers extensively.

### **2.4 Peer to Peer Classification**

Peer to peer systems are classified into structured and unstructured systems based on the architecture of the distributed nodes. An unstructured peer to peer network has no joint that joins the topology and data storage. None of the nodes in the network is responsible for managing or controlling information from its immediate neighbors. A blind search technique to unstructured peer to peer systems may breed quantifiable results. This inundates in all peers with a request before solving it. Gnutella [16] is a typical example of this type.

Structured peer to peer network functions as per the dictates of the Distributed Hash Tables (DHT). This makes the searching operations better off when compared to the experience when using the unstructured networks [17]. Files in such networks are well organized in the nodes [18]. However, one ought to incur more expenses to manage such information. In this case, it is necessary to maintain the routing table. Some of the examples of such a Chord [19], CAN [20] and Kademlia [21]. They are also classified into:

- a. Tight.**

**b. Loosely coupled system.**

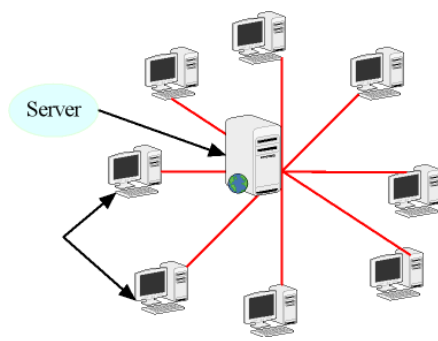
The two are utterly dependent on the degree of coupling. However, in tight systems, there is only one set at a time. Therefore, the user may either use or leave it alone. Identification logic is available for each of the peer. A unique mapping function is used when there is a need to store and retrieve data. This function helps other peers that are in the same group. The routing query is preferred over quitting an overlay network because there is a need to maintain the overhead of a network structure. On the other hand, loosely coupled systems beat tightly coupled limitation as it constrains the population of the peer [22]. Some of the main examples of the first class include Chord [19] and CAN [20]. On the contrary, Free Net [23] and Gnutella are some of the examples of the second. A peer to peer system has three models. These models include the decentralized, the hybrid and the super peer model as illustrated in figure 2.

In decentralized model, all its members have equal capabilities and duties. Examples include Free Net [23].

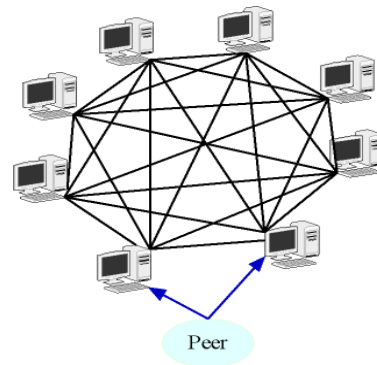
In the hybrid model, there exists an intermediary node which serves as the directory. Its duty is to facilitate the collaborative efforts of the central server/directory. For example Soft wax [24] and Napster [25]. They are referred to as the hybrid models because they use both client/server model and the peer system.

Super-peer model is one of the broker solutions that intermediate between the hybrid and the decentralized model. The super-peer model identifies the nodes which are stronger than normal codes. These nodes are referred to as the search hub. They are more effective as they retain a resource directory that can be used to resolve the user's queries.

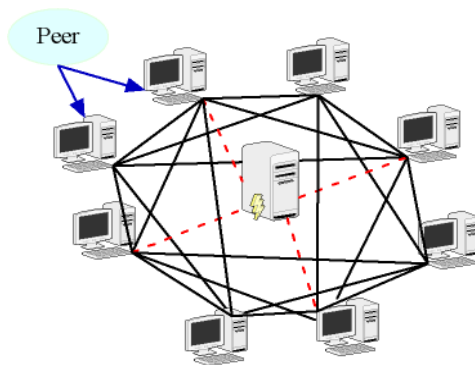
The KaZaa (implements Fast-track protocol) [26] and Jxta [27] are some of the examples which use the super peer model.



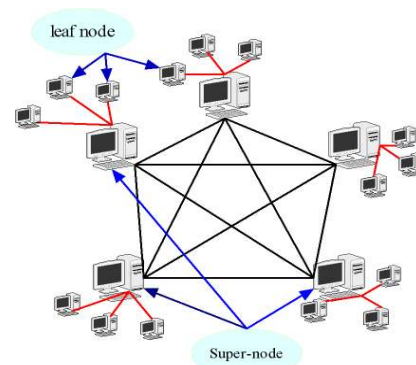
(a) Client-Server Paradigm



(b) Pure Peer to Peer



(c) Hybrid Peer to Peer



(d) Super-node Peer to Peer

**Figure 2** Client-Server Model vs. different Peer to Peer Models.

**Note:**

Some sources of information such as books, [28] brand the super peer model as being one of the hybrids. This is because none of the peers match their roles. Notably, the hybrid model is also known as the decentralized model.

## 2.5 Peer to Peer Applications

### • Content Sharing

File sharing applications have become more popular among users. Peer to peer systems' file sharing applications makes it easy for people using a similar network to share the files. Moreover, it allows them to search and share digital files. Using a

peer to peer network is possible through the use of a software that allows one download a file from a computer on the network. Research indicates that 30% (day-time) and 70% (nighttime) of all the internet traffic in Germany in October emanated from applications that enable sharing of files. The study was conducted by the IPOQUE survey. Statistics indicate that eDonkey and Bit Torrent are some of the most popular file sharing applications as they generate 95% of all the traffic. In fact, they have become more popular than other popular networks such as Kazaa's Fast-track [29] .

### • **Distributed Computing**

Distributed computing refers to a technique that empowers the peer network to work jointly to solve a computational problem. The aim of distributed computing is to make use of the idle resources and computers in other networks. Reports indicate that at the end of 2006, SETI@home computed for at least was 274.0. TFLOPS [30] (1 TFLOPS has a similar value to  $10^{12}$  FLOPS). Notably, the fastest super-computer computed 280 TFLOPS.

### • **Collaborative Systems**

Collaborative systems make it easier for the peers to share on-line games. It is also ideal when there is a need to use instant messaging programs such as MSN and AOL Messengers alongside the use of communications programs such as Skype that allow users to video chat.

### • **Platforms**

A platform is not an application. It is an architectural design popular in many network applications. A platform allows one to share files across the network. A platform supports the main components of peer to peer systems. These components include naming, security, communication, peer discovery and resource aggregation. Furthermore, they make it easy for one to operate many platforms at ago. Some of the major examples of the peer to peer platform include the Extra and Net.

## **2.6 Peer to Peer Characteristics**

The following are the characteristics of peer to peer

- **Symmetry**

Symmetry refers to the nodes' ability to serve the functions of both a server and a client at ago. The symmetry feature enables the peer to peer system to function in a decentralized system. It makes the peer to peer system exclusive among other systems depending on their position on the client/server model.

- **Decentralization**

This is the most popular of all the features of peer to peer systems. This feature does not have any central node. Hence, it does away with the boundaries that differentiate the server from the client. It is advantageous as it is more flexibility and has a better capacity to counter faults [31].

- **Scalability**

The scalability feature allows the servers and the nodes to interact with each other freely. Despite this, it does not affect the performance of the network.

- **Fault tolerant**

Disappearance of too much load on nodes and failure of the central node minimizes the network's performance. The loss of a peer is compensated by the pure and the super peer model.

- **Self-Organization**

Self-organization is quite useful as it controls the behavior of the system and its features. This is particularly useful for adaptability and self-maintenance. Self-



organization is a critical issue for the peer to peer systems as the nodes join the system spontaneously. Ledlie et.al. [32] observe that 80% of all the nodes stay in the peer to peer system for at least an hour. One of the causes of failure is the unpredictable number of users. Hence, management of such a fluctuating environment is not easy.

- **Resources Sharing**

Peer to peer network makes it easy to share sensor information through its many applications and services. File sharing applications allow the user to share many forms of information. The network's computing power increases with the number of participants.

- **Fast Resource Location to Determine Where to Find The Resource**

In the client/server model, the address of the server is not strange. Hence, requests are directed server without an interval. Peer to peer systems create overlay networks whose requests is to route any node efficiently. This means that means peers can discover and locate any roaming resources.

## **2.7 Examples of well-known peer to peer systems**

### **2.7.1 Napster**

Napster in 1999 [25] was one of the earliest peer to peer services that made file sharing easy. The Napster's central directory contains all the information that one searches for. It is the boots trap node which ensures that each node takes part in the network. A single point's failure may attack the system when there is a Denial of Service (DoS).

### **2.7.2 Gnutella**

Gnutella [16], [33] is one of the applications that provide a reliable distributed

system. The (V.6 or Gnutella2) uses the super peer model to improve the search experience. On the other hand, Gnutella (V.4) depends on query flooding for searching information across a range of networks. However, in Gnutella, it is not possible to control the peers. It is branded as being a peer to peer system due to its self-organization. It is evident that flood routing impacts negatively on the Gnutella network. Hence, it is likely to limit its scalability to thousands of peers.

### **2.7.3 Free Net**

Free Net is one of the peer to peer systems used for distributing, replicating and retrieving files. Its aim is to maintain an anonymous nature of the readers and authors of the data. Free Net creates an exploratory Document Routing Model used for maintaining the decentralized mode. This mode is also referred to as the Key Based Routing (KBR). Free Net has an excellent scalability as it does not have a central node. They are self-organized as the nodes are not easy to control.

### **2.7.4 Search for Extra-terrestrial Intelligence (SETI@home)**

SETI@home [34] is an ancient, large distributed computed project that joins many computers to the independent equations. Since December 2005, SETI@home uses BOINC. BOINC is one of the distributed platforms that support many applications in different scientific fields such as Biology and Astronomy. The BOINC architecture depends on the client server including the client software and the server system. Lack of direct communication has been blamed for the occurrence of conflicts between the peer to peer systems and the client server.

### **2.7.5 Groove**

Groove Virtual Office is an excellent method of creating a pool of workers on the same page. Groove refers to a desktop windows based on shared and direct messaging to peer to peer application targeted to the Internet users. Groove is one of the bastard models. It uses a centralized server for supporting centralized services.

### **2.7.6 Jxta**

Jxta refers to an overlay network that may create a decentralized peer to peer system. It makes it possible for distributed computing applications to function normally on many of the devices of the cellular devices such as Personal computers, personal digital assistants and cell phones. Unlike others Jxta is network independent. It is ideal for computing infrastructure and networking programming.

### **2.7.7 Chord**

Chord is one of the structured overlay networks that is found on the DHT. It is a decentralized system that can retrieve data using an  $O(\log(N))$  messages. In this case,  $N$  represents the number of nodes in a certain system. For a system to join or leave expensive operations in the chord overlay network, they demand  $O(\log(N)^2)$  messages. Chord network's codes have a  $2^m$  ring. In this case,  $m$  is the constant integer that can retrieve data in  $O(\log(N))$ . Each of the nodes may retain all the information of both successors and the predecessors.

### **2.7.8 Content Addressable Network (CAN)**

CAN is one of the decentralized peer to peer systems. CAN provides a DHT function as one of the algorithms that is used as a document routing model. It is a peer system that organizes the nodes into a toroidal space. Each of the nodes is directly linked to a hypercube. Therefore, its neighbors influence the neighboring ice cubes. The CAN algorithm depends on a hash function to draw a determinate point on the coordinate space. The node at the mid-zone is where the point lies as the data element. The point may be retrieved through a greedy forwarding pattern. CAN is potentially capable of routing a message in the neighboring peer depending on the target coordinates that are strategically located in  $O(d \cdot N^{1/d})$ . In this case,  $N$  represents all the nodes in an overlay network. On the other hand,  $d$  represents the dimensions.

### **2.7.9 Bit Torrent**

The Bit Torrent refers to a protocol that may be used for distributing large volumes of data through the internet to the users of peer to peer systems. Exchanging the files among a group of users does not necessarily need a broker. It only needs a tracker. A tracker refers to a program that is normally housed at the server. The tracker coordinates the communications between the peer users. All the data downloaded or uploaded is transmitted to the users without much ado.

Bram Cohen is the programmer who designed the Bit Torrent protocol back in the first quarter of 2001. However, the program was used for the first time in 2<sup>nd</sup> July 2001. The python was used as the programming language for the Bit Torrent Protocol.

#### **2.7.9.1 Sharing by Torrent**

The availability of the tracker determines whether the torrent can share the protocol or not. The duty of the tracker is to coordinate communication between the torrent program and the participants. The torrent is responsible for the assimilation process when there is a need to connect the tracker to the main ideal for purposes of developing a file extension. However, the file should not exceed 4MB and it should not be less than 64KB. Such a file is made available to the users of the software as one may download the files using the tracker and a torrent program.

#### **2.7.9.2 Upload and Share Torrent Files**

Internet users browse to search for torrent files that match with their most preferred downloads. The torrent program supports formulas such as the bit torrent, utorrent and the bit connect. The formulas are also referred to as the client in an instance where the program is linked with a tracker in a previous torrent file. Through this, the user may share the files with a group in the peer system.

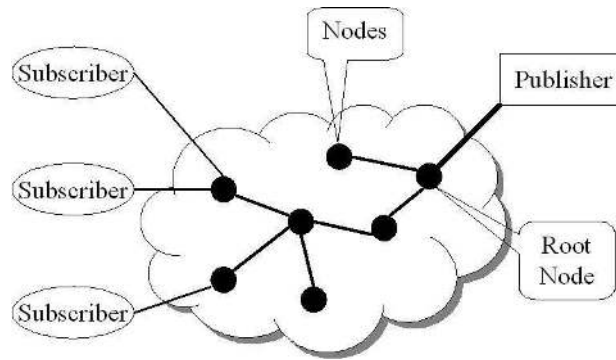
## **CHAPTER 3**

### **PUBLISH / SUBSCRIBE SYSTEM**

The subscribe system is one of the newest paradigms used for creating large scaled applications for various systems. A pub-system is mainly used on an overlay network to enable the publisher distribute information to the subscribers. Surprisingly, there is a crop of publishers who do not know that there are many consumers who use the published information. Some of them publish information without the knowledge that there are consumers who assess it through the system. They publish the information through examining the characteristics of the information that it yet to be published. Consumers/subscribers engage in various subscription mechanisms when they are interested in some form of information. They subscribe and wait to be informed of an upcoming event. Subscribe infrastructure has the duty of ensuring that the events match. Moreover, it links the consumer's subscriptions and sends the matching events that they express interest in receiving.

There are two subscribe systems models. One of the models is content based while the other one is topic based. A topic based model is identical to the newsgroups. Hence, each of the users may express their wishes through linking their topics of interest to a group. Notably, all the messages on a certain topic is available to the users of a group that subscribes to it.

On the other hand, content based systems introduce a scheme of subscriptions. Arguably, it is more desirable over the topic-based model. It has a competitive advantage over the topic-based model as its schemes are related to the real content. It is easier for users to express their concerns through creating special predicates of a certain number.



**Figure 3** Basic Pub-Sub System

### 3.1 Existing Topic-based Publish/Subscribe Systems

The old architectural designs for subscribed systems are based on the client and the broker. Therefore, its models are both broker and client based. A system that is operational on the basis of either of the models is dependent on the server. Essentially, a publisher transmits these events through the server. The server serves as the joint where events are channeled to the various subscribers and directed to them for ease of use.

System solutions such as Siena [35], Gryphon [36], Hermes [37] or Corona [38] fall in this category. The latest architectural designs for the subscribe systems utilize the peer overlay model. They prefer it over the broker based and client based models. This makes it easy to use the internet scale applications with ease to many users and across a range of topics. The peer overlays fall into two major categories. They include the unstructured and structured overlays.

Systems solutions such as Scribe [39] and Bayeux [40] are some of the major examples of structured overlay networks. On the other hand Tera [4], Rappel [41], StAN [42] and Spider Cast [43] are in the second category. There are some solutions such as Quasar [44] or our solution, Vitis. These solutions utilize gossiping to create high profile structured and unstructured overlays disseminate the events. Constructing an overlay is a challenging activity. One of the major challenges that bedevil this activity is to ensure that the subscribers receive the events which they

have subscribed to receive. They find it quite difficult to maintain an average load as there are many connections due to an extreme overhead.

Tera [4], Rappel [41], StAN [42], and SpiderCast [43] construct create a different overlay for either of the topics. A node joins the overlay when it becomes a subscriber of the topic. Hence, all the published events for a certain topic is transmitted to the subscriber nodes. In this case, the traffic overhead is eliminated. The nodes should be linked to the overlays depending on the number of topics which the users have subscribed to. Hence, the degree of the node alongside the maintenance paradigm of the overhead is linear when compared to the node subscription. However, it cannot be used in Internet scale applications especially when subscribers are signatories to many topics. An issue emergence when there is a need to address the Vitis. This is because the nodes maintain some of the connections regardless of the active subscriptions. To minimize the problems of the scale, Spider Cast [43] compares the nodes in regard to the similarity of the nodes. The Spider Cast authors argue that one link may be used to connect a node to many topic overlays depending on the subscription associations. Hence, the connection needed in each of the nodes is minimal. The user subscriptions can be presented through relating the traces [13], [45]. This is utterly successful when handling minimal node subscriptions. Despite this, the scalability of the Spider Cast is yet to be established especially if there are many subscriptions. To complete many of the subscriptions using Spider Cast, the application should have enough knowledge of other nodes that have been in the system. It should have at least 5% of all the information. On the other hand, the Vitis nodes do not utilize linear bits of information regarding other nodes operating in the system. Hence, it is possible to subscribe to a wide range of topics. There are other systems' solutions that may be used to account for the scalability. Some of them include bounding the node connections. For instance Quasar [44], which is a gossip-based solution, or Scribe [39] and Bayeux [40], which are DHT-based can be bounded.

For Quasar [44], each of the nodes may be exchanged with its neighboring nodes. It is a form of subscription where the neighboring nodes may move away. Hence, all the members of each of the nodes are available in the overlay. The node transmits

many copies of the event randomly. Quasar can function without an overlay structure used to encode information regarding the members of a group. Arguably, it does not deviate from its design model regardless of the immediate environment. Moreover, it records high traffic because it is not aware that there are node subscriptions. Similarly, it includes many other nodes when transmitting an event to the subscribers. On the other hand, in Vitis, it reaches the maximum ratio and reduces the traffic through organizing related nodes into a series of clusters.

In Scribe [39] or Bayeux [40], nodes appear in a Distributed Hash Table (Pastry [35] and Tapestry [46], respectively). Similarly, each of the nodes retains  $O(\log N)$  connections. Each of the topics should have a spanning tree and one rendezvous node next to the root. The rendezvous node communicates to other nodes connected to the spanning tree. However, using a spanning tree is not recommended as it compels the nodes to transmit information which the subscribers have not subscribed for because they appear on the path on the way of the rendezvous node. Hence, such a system records a high traffic overhead. On the other hand, Vitis nodes have a minimal degree and create a structure that resembles a tree for each topic. Noteworthy is the fact that Scribe and Bayeux do not have group nodes. Instead, they form single nodes depending on the topic's subscriptions.

Using Magnet [47], [48] is one of the solutions that avails identical ideas in a subscription link depending on the nodes being used. The magnet has to be strategically placed in an overlay as it cannot trap the subscription's link with ease. Therefore, it is bounded on one of the dimensional spaces. The space should be a point where a structured overlay originates from. The performance of a magnetic is limited in a volatile environment. Some such volatile environments which the magnet may not perform exceptionally well include the internet. On the contrary, the Vitis may perform well in any of the dimensions and maintain the subscription correlation owing to the fact that clustering is performed structurally. Lastly, research to determine the location of resources for clouds [49], is ongoing. The study is expected to reveal more information to enhance the understanding of the subscribe system and clear away the ambiguities.



### 3.2 Existing Content-based Publish / Subscribe Systems

Content-based subscribe system is one of the best platforms that may be used to determine the events which many subscribers prefer. Therefore, it may be used to determine which topics that should be published and delivered to the users. Understanding this enhances content delivery to the interested users. Ongoing research indicates peer to peer network is critical when handling technology based research that involves tolerance of faults and scalability. Research indicates that mixing subscriptions could be one of the ways of enhancing the performance of peer nodes.

There are many solutions that could be used to enhance the efficiency of the subscribe model [50], [47], [43], [4]. Many of the solutions enhance the performance of the content based subscriptions to its users. For example, in Meghdoot [51], each of the nodes emanates from a 2d dimensional space. Hence, each of the node subscriptions is directly attached to a certain point. In this case,  $d$  represents the route of the dimensional pattern. On the other hand, the CAN [52] overlay is useful when there is a need to route the messages. One of the major criticisms laid against the Meghdoot is its incompetent routing as there are many challenges that bedevil the CAN overlay. However, Meghdoot is ideal when there is a need to match the subscriptions. In addition, the node degree could liaise with many other features linearly. The load on some of the nodes is not stable. This depends on the location of the node in the CAN overlay.

Sub-2-Sub [53] is quite different from others. Unlike others, it divides the subscription space into many units. In each of the units, there are only nodes that have been subscribed into by the users. Each of the units function as a distinct topic when using the topic based approach. A ring is used in each of the units to disseminate the events that take place in each of the units. However, there are two types of problems which emerge. One of the problems emanate from the fact that it is difficult to create the units for complex subscriptions. Hyper [54], is one of the non-peer-to-peer solutions used to solve problems for the content-based publish/subscribe.

The NP complete is one of the most challenging problems especially in the peer to peer network as it has a churn. In addition, retaining a ring in each of the units is not easy as there are many overlays connected to the same node. Therefore, the cost of maintaining it is exceptionally high.

Ferry [55] is one of the approaches that may be used to solve problems that occur in the structured overlay network. It is particularly ideal when there are multiple subscriptions on the overlay network. In this case, each of the nodes creates hash values. Through this, it is possible to maintain the subscriptions that occur at the rendezvous nodes. Once the event is published, it is distributed to the rendezvous nodes and channeled to the subscribers in line with their subscriptions.

A procedure of solving the problems in Ferry is available in the eFerry [56]. The procedure is applicable when the problem is complex depending on the subscription registration. Other loadable properties can be identified through the use of certain proposed mechanisms when handling the normal systems.

CAPS [57] is one of the solutions that demand a restricted node degree. Just like Ferry, CAPS transmits the events to the subscribers. Similarly, it uses the meeting model to sort out its installation issues to its subscribers. The difference between Ferry and CAPS is that the latter generates values depending on each of the subscriptions. Moreover, the meeting points for the nodes are installed for each of the subscriptions. Later on, it matches them to the meeting nodes in the overlay link depending on the subscriptions. One of the problems of using CAPS emanates from the fact that it is easy to convert them to several other keys before installation. Hence, chances of having high traffic turnover for the network are very high. Moreover, matching can only be done centrally as there is no reliable method of balancing the load.

Unlike the Meghdoot and Sub-2-Sub, the Vinifera nodes retain a fixed number of connections. In this case, each of the nodes can only accommodate a small additional load. Just like eFerry, Ferry and CAPS, Vinifera depends on a restricted node degree.

Similarly, it uses an identical routing mechanism when installing subscription and delivering events to the subscribers. However, it does not hash the feature names. It only hashes the feature values using the order preserving hash function [58], [59]. This makes it easy to balance the load and distribute it through sharing the nodes uniformly.

## CHAPTER 4

### PROBLEM FORMULATION

#### 4.1 Preliminaries

Let  $V$  represent the node sets while  $T$  represents the topics. Let  $n = |V|$ . The function of interest  $I$  may be defined as  $I: V \times T \rightarrow \{0, 1\}$ . For the node  $v \in V$  and topic  $t \in T$ ,  $I(v, t) = 1$  if and only if node  $v$  has a valid subscription to a certain topic ( $t$ ) and  $I(v, t) = 0$  otherwise. For a set of nodes  $V$ , an overlay network  $G(V, E)$  is an undirected graph on the node set  $V$  with edge set  $E \subseteq V \times V$ . For a topic  $t \in T$ , let  $V_t = \{v \in V \mid I(v, t) = 1\}$ .

Given a topic  $t \in T$  and an overlay network  $G(V, E)$ , the number of topic-connected components of  $G$  for topic  $t$  is equal to the number of connected components of the subgraph of  $G$  induced by  $V_t$ . An overlay network is called topic-connected if for each topic  $t \in T$ ,  $G$  has at most one topic-connected component. The graph's diameter refers to the length of the longest shortest path in the graph. The  $v$  node's degree in an overlay network  $G(V, E)$  has equal value to the sum of the edges adjacent to  $v$  in  $G$ .

**Fault Tolerant Overlay Networks design problem:** Given a set of nodes  $V$ , a set of topic  $T$ , and the interest function  $I$ , construct a topic 2-connect overlay network  $G$  that has the least possible maximum degree.

## 4.2 Fault Tolerant Overlay Networks Design Problem and Greedy Merge (GM) Algorithm

Chockler et. al. [60] introduced the Min Av-TCO problem. Their main aim was to minimize the average degree of the node. In this chapter, the researcher presents the widely accepted definition of the Min Av-TCO problem. Moreover, they outline some of the major techniques that correspond to the Merge (GM) algorithm. The algorithm is useful in this approach when looking for solutions to Min Max-TCO.

First, the researcher examines the accepted definition of the problem of Min Av-TCO. When one has a set of nodes  $V$ , a set of topics and an interest assignment  $I$ , one should connect the  $V$  nodes to a topic-connected  $G$  overlay network. They should ensure that the overlay network has a minimum number of edges as they have a minimum average node degree.

The Greedy Merge (GM) Algorithm [60]: Initially we have a set of nodes  $V$  and no edges in between the nodes. In each of the steps, one should add the edge which reduces the number of topic-connected components maximally

The GM algorithm is not recommended for the Min Max- TCO problem as it does not find good solution. The maximum ratio and the approximate ratio of the GM algorithm may reach critical levels of  $\Theta(n)$ .

## 4.3 2TCO Problem and GM2 Algorithm

The Greedy Merge algorithm for the 2TCO is one of the effective methods applied to the Min Avg-2TCO problem, GM2 for short. Although GM2 has an identical structure as GM and other centralized algorithms that are used to build TCO [8], [9], [60], GM2 has a different progress measure.

When given a TPSO  $(V, T, In\ t, E)$ , the topic 2- connected of  $topic \in T$ , is identical to a maximal that is 2-connected sub-graph directed towards a topic (i.e, it is not available in large 2-connected sub-graph that are induced on  $t$ ). It is also referred to as the topic-connected component or topic-connected block. Therefore, each of the

TC blocks in  $t \in T$  may either have a maximum 2 -TC sub-graph a bridge. The bridge includes all the endpoints. There are instances where it may also have a lone node in  $G(t)$ . In addition, each the sub-graphs in  $G(t)$  is a TC block. The maximum property of the TC blocks of on  $t \in T$ , they overlap with at least one of the nodes in  $G(t)$ . Therefore, each of the edges  $e \in E(t)$  is available on the TC block on  $G(t)$ .

As illustrated in the Alg. 1,  $\text{TPSO}(V; T; \text{Int}; E)$  starts in GM2 in an overlay network. In this case,  $E = \emptyset$ . Hence, there are  $v | \text{Int}(v; t) \}$  singleton TC-blocks for each of the topics  $t \in T$ .

The sum of the TC-blocks at the beginning is

$$\mathbf{B}_{\text{start}} = \sum_{t \in T} |v \in V | \text{Int}(v, t) \}| = O(|V| |T|). \quad (1)$$

The algorithm computes for the edge of  $E$  iteration through iteration until  $\text{TPSO}(V; T; \text{Int}; E)$  remains with one TC-block for each  $t \in T$ , i.e, 2 -topic-connected. Hence, the sum of the TC blocks is minimized into being

$$\mathbf{B}_{\text{end}} = |\{t \in T | \exists v \in V \text{ s.t. } \text{Int}(v, t) = \text{true}\}|. \quad (2)$$

#### 4.4 Fault Tolerant Overlay Networks Design Problem and Our Algorithm GM3

Here, we presents our algorithm GM3 which is used to provide a solution to the fault-tolerant overlay networks design problem. GM3 starts with an overlay network  $G(V, \emptyset)$  at each iteration of GM3, an edge with maximum weight - where the edge's weight  $(u, v)$  is given by decrease the number of topic 2-connected components which would result from the addition of  $(u, v)$  to the current overlay network - among the ones which minimally increases maximum degree of the current graph is added to the overlay network's edge set. Let  $\text{NC}(V, E)$  indicate the sum of topic 2-connected components in an overlay network represented by  $(V, E)$ .

The first sixth steps of GM3 create an initial weighted graph  $G'(V, E', w)$  on  $V$ , where  $E' = V \times V$  and  $w(\{u, v\})$  has equal value of amount of reduction in the

number of topic 2-connected components causing by the addition of the edge  $(u, v)$  to the current overlay network (represented by the edges in (OverlayEdges). At first, this amount and the number of topics that nodes  $u$  and  $v$  have in common are equal.

**Algorithm 1:** Fault Tolerant Overlay Networks Design Algorithm (GM3)

```

1: Overlay Edges  $\leftarrow \emptyset$ 
2:  $V \leftarrow$  Set of all nodes
3:  $G(V, E') \leftarrow$  Complete graph on  $V$ 
4: for  $\{u, v\} \in E'$  do
5:  $w_{\{u, v\}} \leftarrow$  Number of topics that both of nodes  $u$  and  $v$  have
6: end for
7: while  $G(V, \text{Overlay Edges})$  is not topic-connected do
8: Find maximum-weighted edge  $e$  on  $G'(V, E', w)$  among the ones which increase
the maximum degree of  $G(V, \text{Overlay Edges})$  minimally.
9:  $\text{Overlay Edges} = \text{Overlay Edges} \cup e$ 
10:  $E' \leftarrow E' \cup e$ 
11: for  $\{u, v\} \in E'$  do
12:  $w_{\{u, v\}} \leftarrow \text{NC}(V, \text{Overlay Edges}) - \text{NC}(V, \text{Overlay Edges} \cup \{u, v\})$ 
13: end for
14: end while

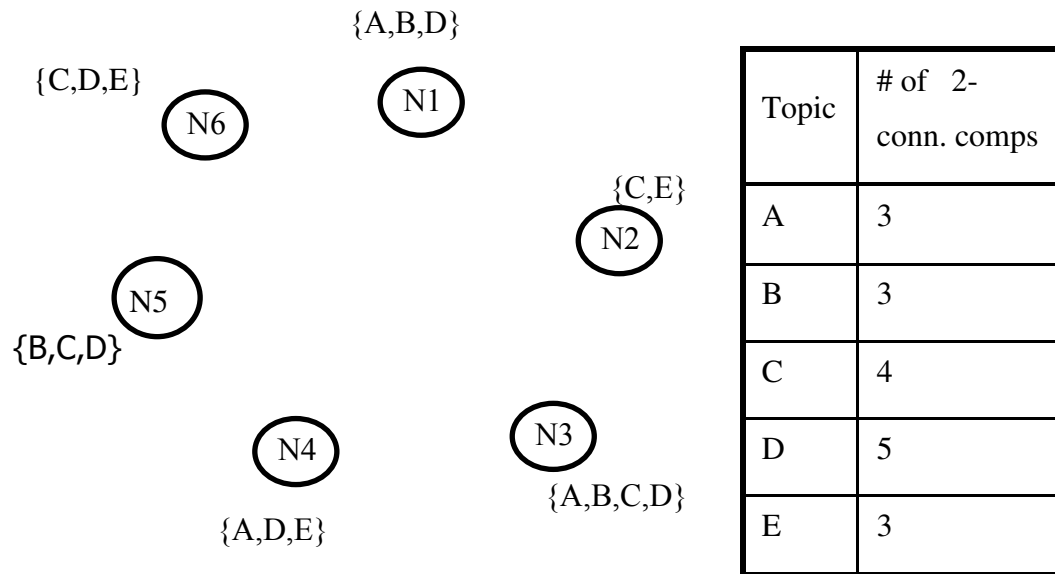
```

A look at GM3 similar that it is similar to GM. In fact, it can be presented in phases with each of the phases being a number of edges that match with the nodes in the sub-system connected to the connected components of various topics as illustrated below. Matching of the selected edges used for an approximate ratio analysis cannot be used for the GM.

It is evident that an algorithm is altered in  $O(|V|^4|T|)$  time.

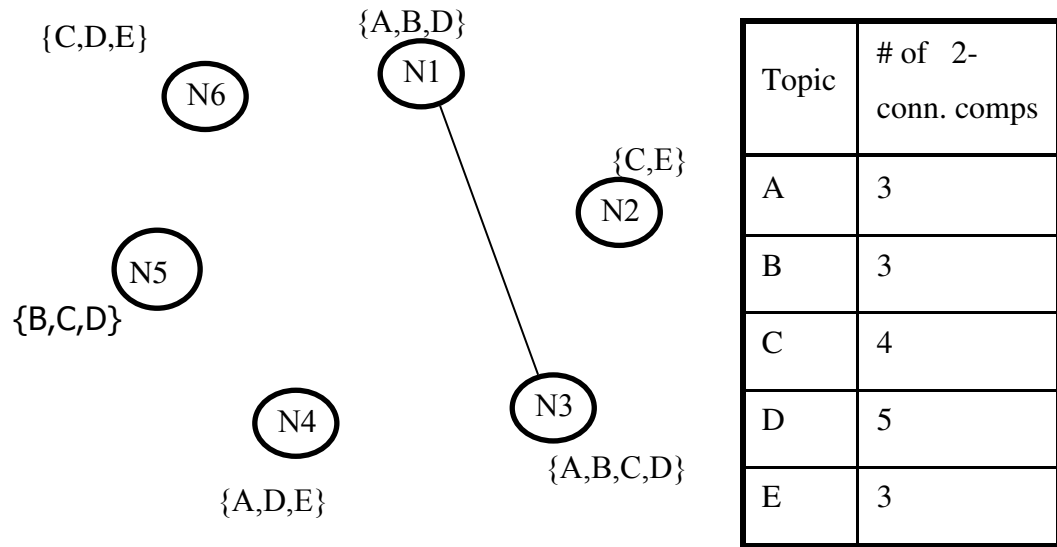
#### 4.4.1 Example of GM3 Algorithm

- Start off with a singleton in a component of each of them  $(v, t) \in V \times T$  Figure 4(a-g).
- At each iteration: add an edge with maximum-weight among the ones which increase the maximum degree minimally figure 4(h-m).
- Halt the process when there is a 2-connected component in each of the topics figure 4(m).

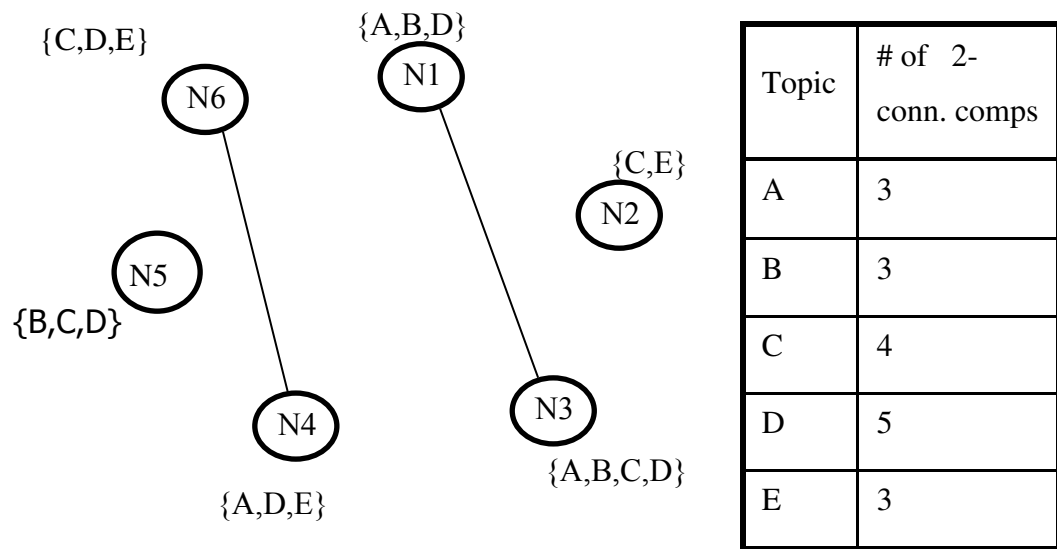


**Figure 4a** Example of GM3 algorithm step 1 nodes without any edge.

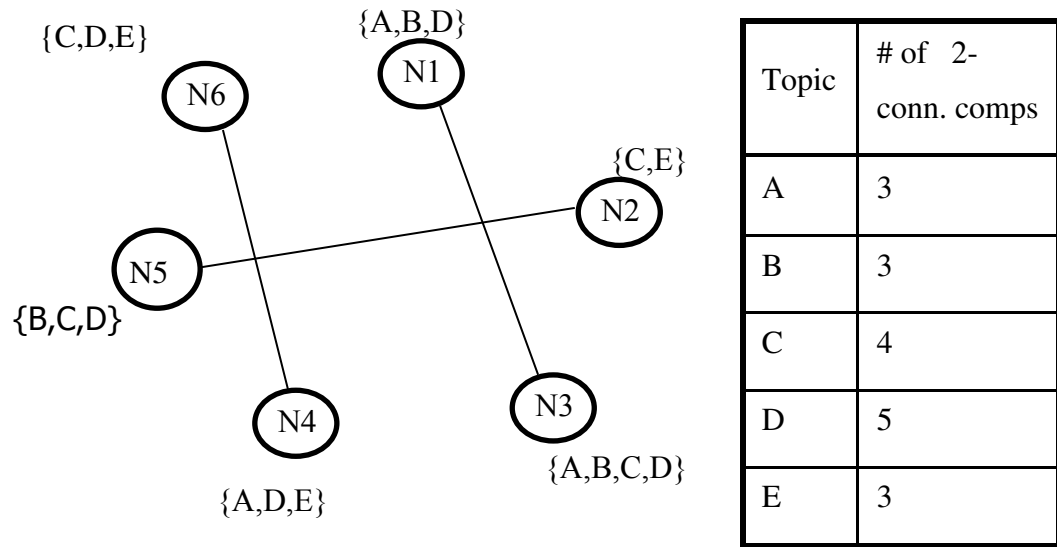




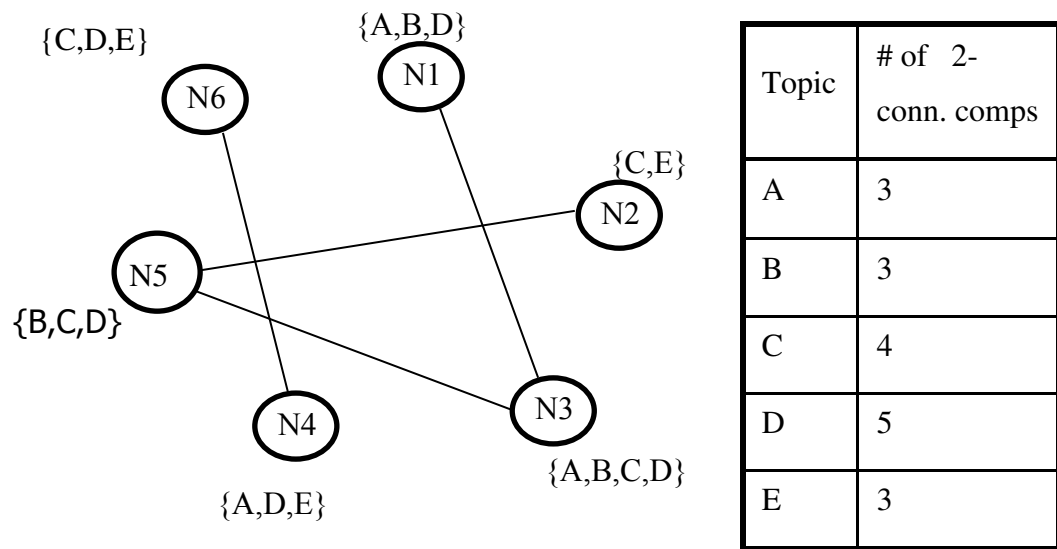
**Figure 4b** Example of GM3 algorithm step 2 add edge 1 with max intersection.



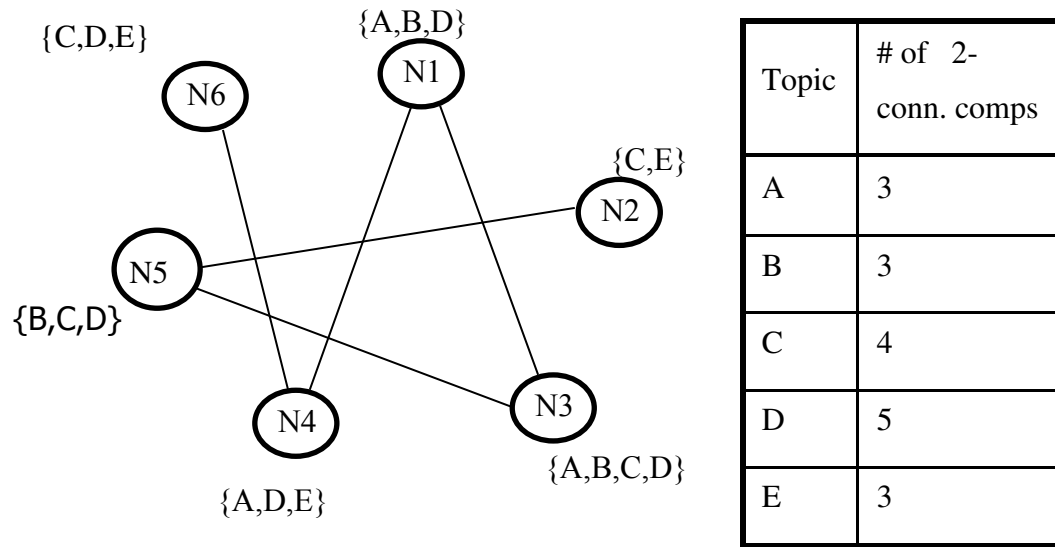
**Figure 4c** Example of GM3 algorithm step 3 add edge 2 with max intersection.



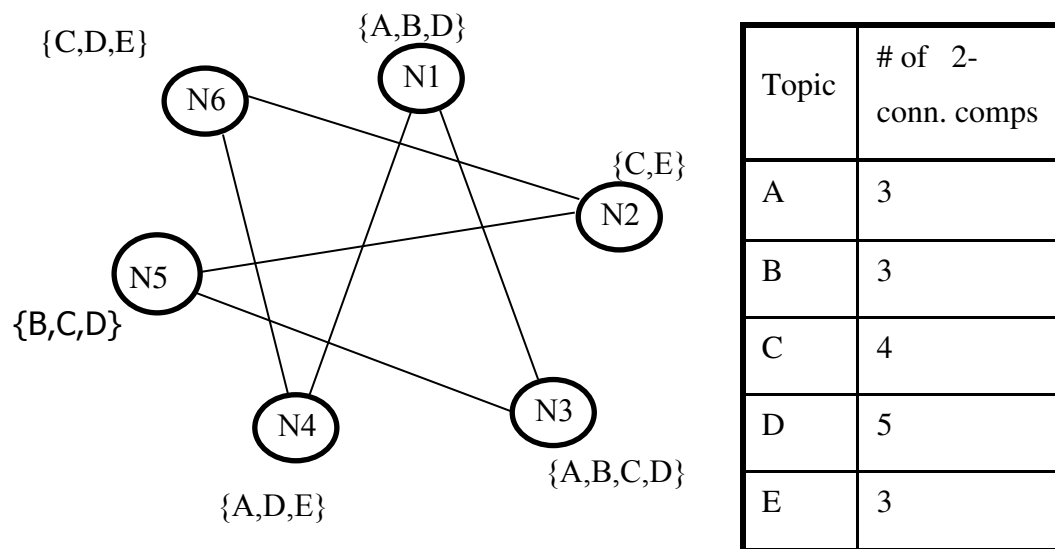
**Figure 4d** Example of GM3 algorithm step 4 add edge 3 with max intersection.



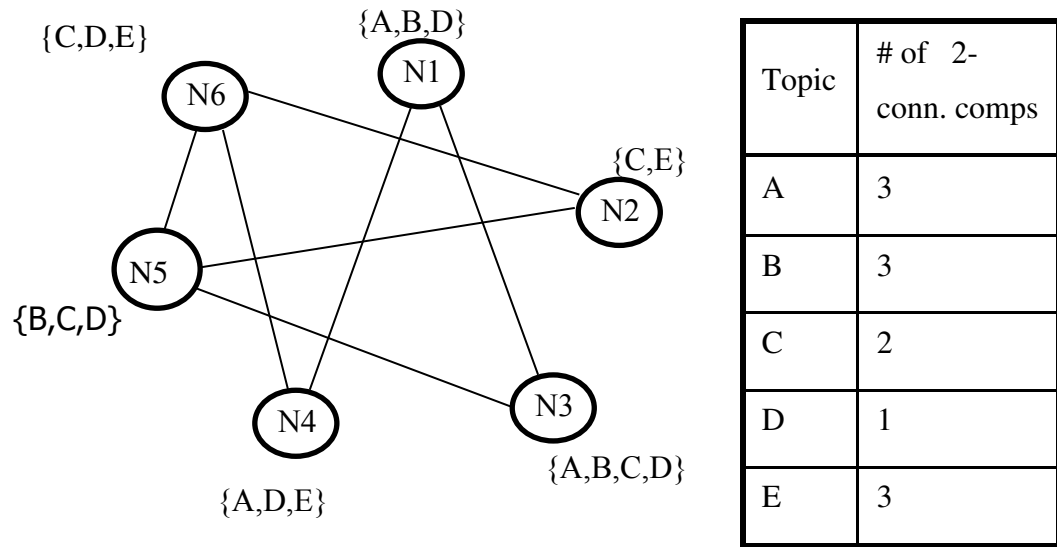
**Figure 4e** Example of GM3 algorithm step 5 add edge 4 with max intersection.



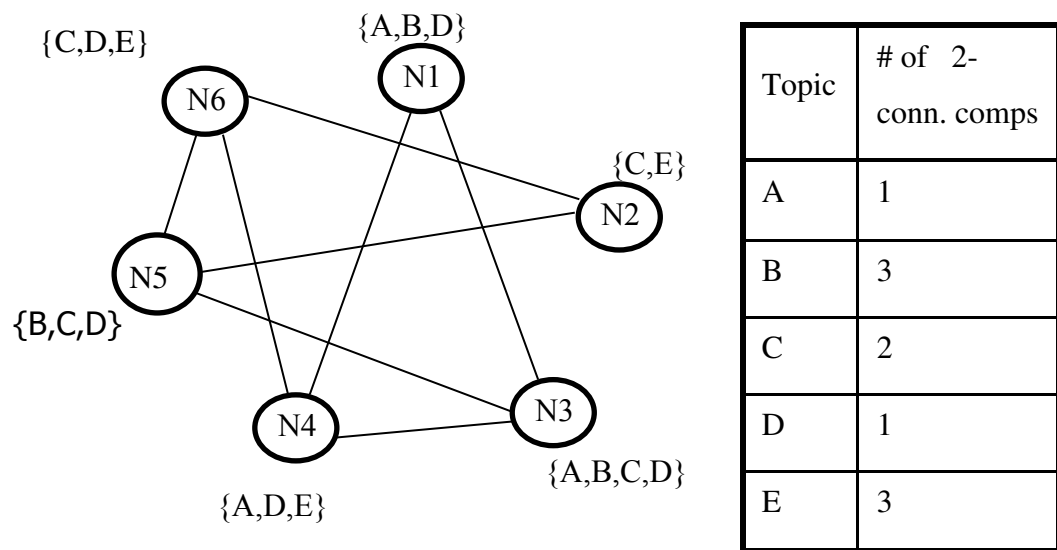
**Figure 4f** Example of GM3 algorithm step 6 add edge 5 with max intersection.



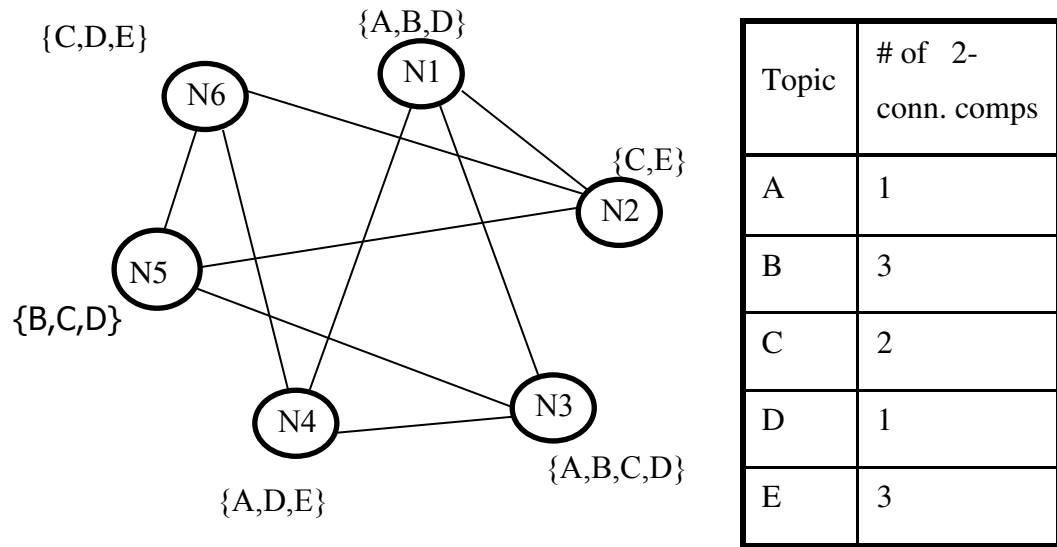
**Figure 4g** Example of GM3 algorithm step 7 add edge 6 with max intersection.



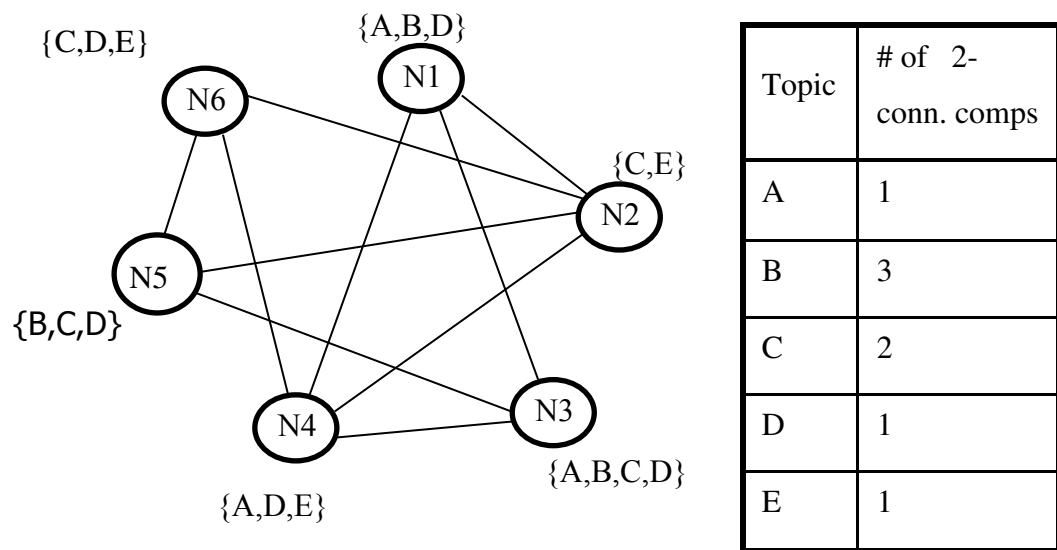
**Figure 4h** Example of GM3 algorithm step 8 add edge 7 with maximum weight.



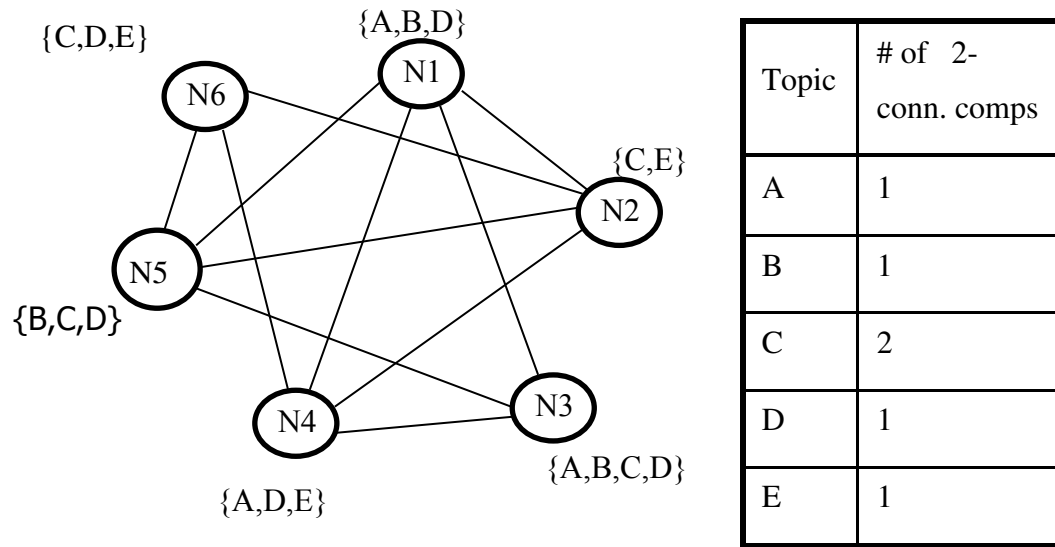
**Figure 4i** Example of GM3 algorithm step 9 add edge 8 with maximum weight.



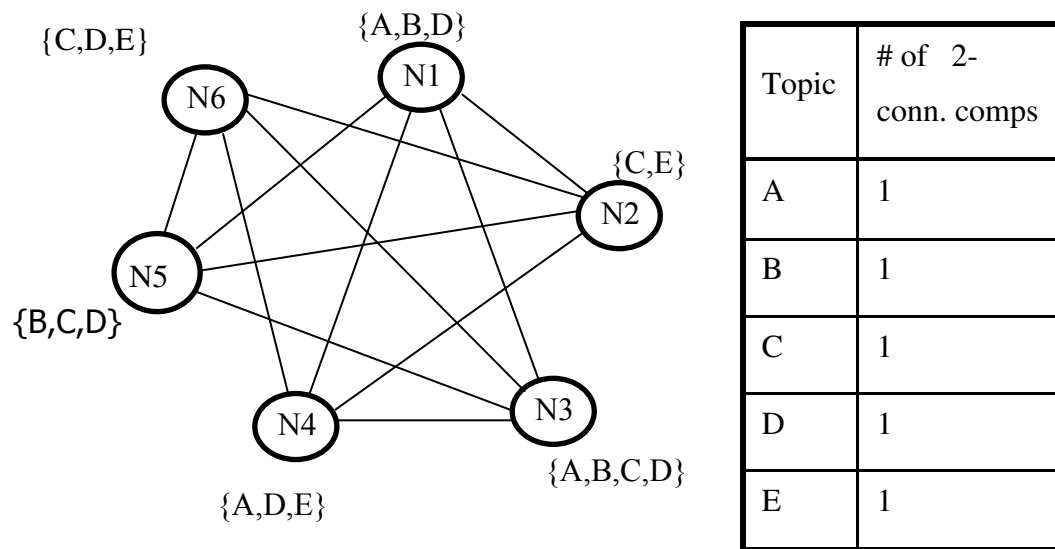
**Figure 4j** Example of GM3 algorithm step10 add edge 9.



**Figure 4k** Example of GM3 algorithm step11 add edge 10 with maximum weight.



**Figure 4l** Example of GM3 algorithm step12 add edge 11 with maximum weight.



**Figure 4m** Example of GM3 algorithm step13 add last edge with maximum weight  
there is a 2-connected component in each of the topics.

## CHAPTER 5

### EXPERIMENTAL RESULTS

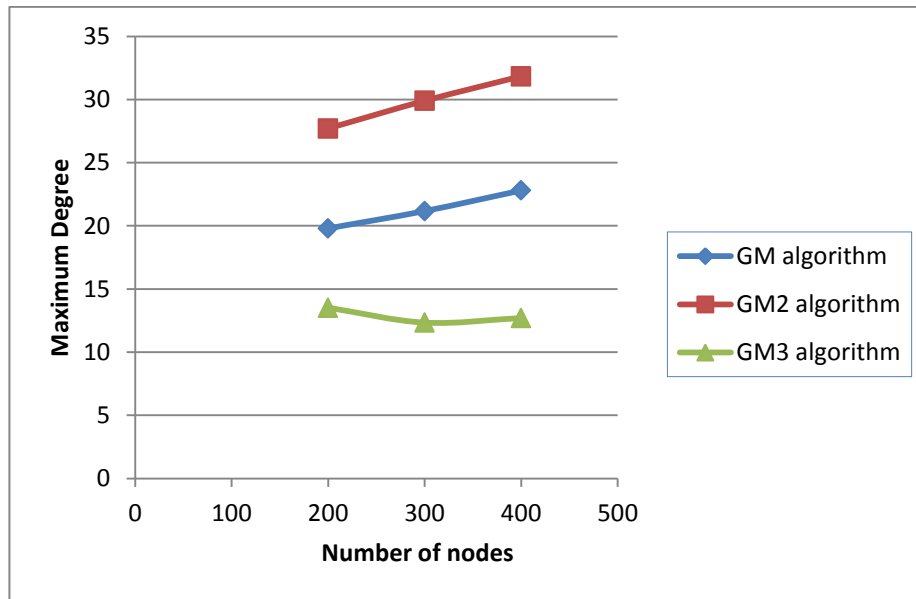
There are three major algorithms. They include the GM [60], the GM2 algorithm and our GM3 algorithm. We used C++ language for simulation. Comparing them depends on the degree generated by the overlay graph. The results of the experiments indicate that the GM3 decreases the maximum degree of an overlay network .

#### 5.1 Maximum Node Degree

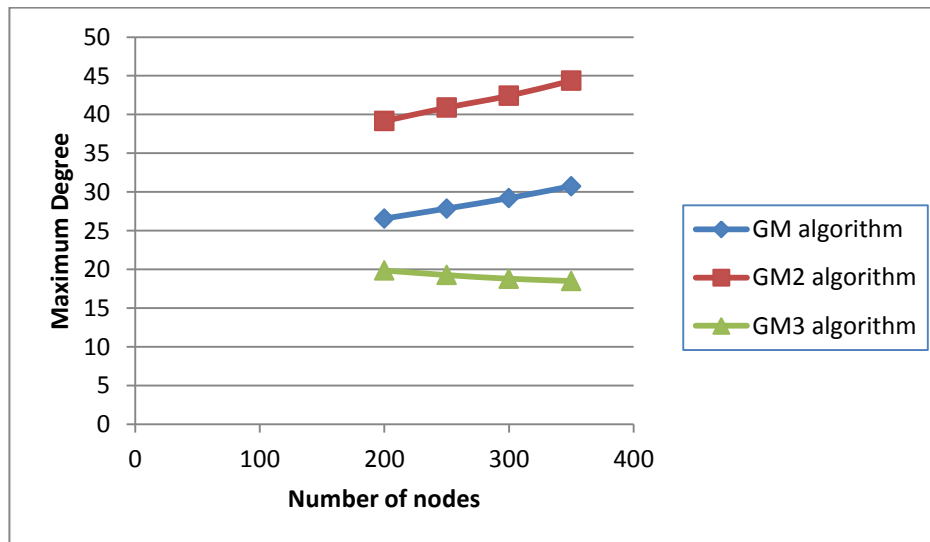
In the experiment, the nodes do not exceed 400. The minimum number of the nodes is 200. For the first experiment, (Figure 5) has a total of 100 topics. In the second experiment, (Figure 6) has a total of 200 topics. Number of the subscriptions fixed to  $s = 10$ . Each node is interested in each topic uniformly at random. The setting of this experimental looks like previous studies [60].

Figure 5 indicates that the maximum degree by comparing the three algorithms (GM, GM2 and GM3). A rise in the number of nodes translates into a decrease in the graph's maximum degree for the GM3. This is because the GM3 algorithm may move up to the edges that have a higher correlation when the number of the nodes rises. Noteworthy is the fact that an increase in the number of nodes causes the graph's maximum degree for the GM and GM2 algorithm to increase. Therefore, there are many nodes that have a higher correlation. In this case, the GM and GM2 algorithms assign edges to the nodes when their number rises. The GM3 optimizes the GM 38% and GM2 56% on max degree. This is only possible in an instance where the results of the GM3 and those of the GM and GM2 are compared in (Figure 5). Similar results are illustrated in (Figure 6). Both in Figure 5 and Figure 6, the maximum degree increase slightly for GM and GM2. On the other hand, the

maximum degree decreases for our algorithm the GM3. This is because there is minimal correlation because the topics have increased.



**Figure 5** Maximum node degree for GM, GM2 and GM3 when number of topics is 100.



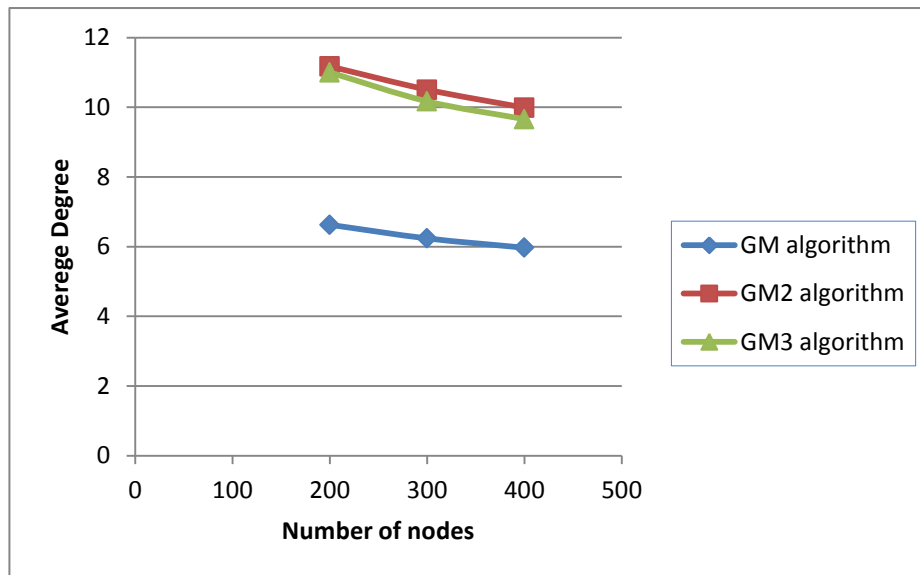
**Figure 6** Maximum node degree for GM, GM2 and GM3 when number of topics is 200.



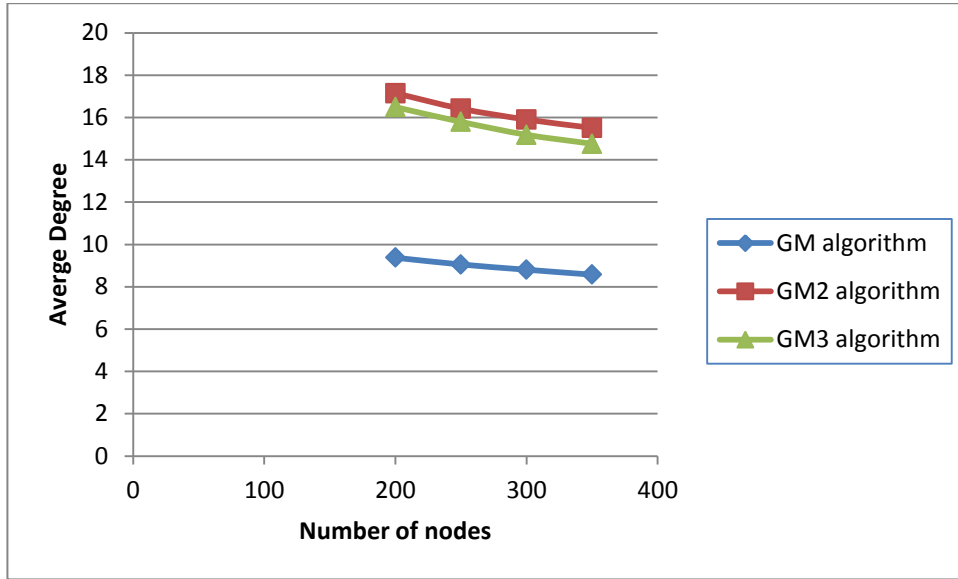
## 5.2 Average Node Degree

The settings of the experiment for this section are identical as those of the previous sub-section. Figure 7 shows the GM, GM2 and the GM3 algorithms. The comparison is dependent on their average degree. The algorithms prefer the edges that have a larger node correlation. This is done when the nodes increase because the graph's average degree decreases for all the three algorithms. This is expected when the nodes increase.

GM is better off when compared to GM2 and the GM3, 40% on average (Figure 7). Similar results are valid for Figure 8.



**Figure 7** Average node degree for GM, GM2 and GM3 when the number of topics is 100.



**Figure 8** Average node degree for GM, GM2 and GM3 when the number of topics is 200.

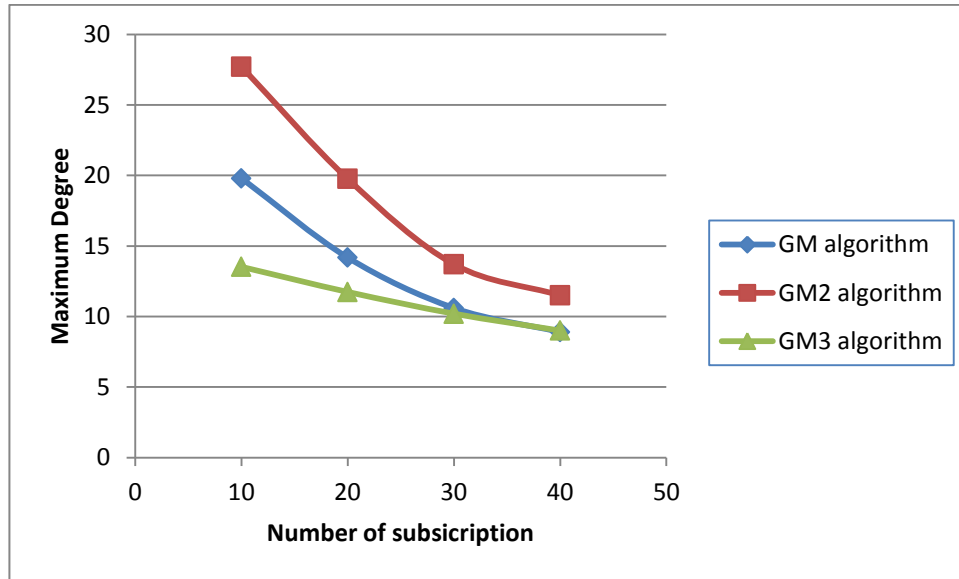
### 5.3 Subscription Size

In this experiment, the number of nodes is 200. In addition, the number of topics are fixed; that is, 100. The size of the subscription does not exceed 40. On the other hand, it is not less than 10. Research indicates that each of the nodes has an equal interest in the available topics despite their random nature.

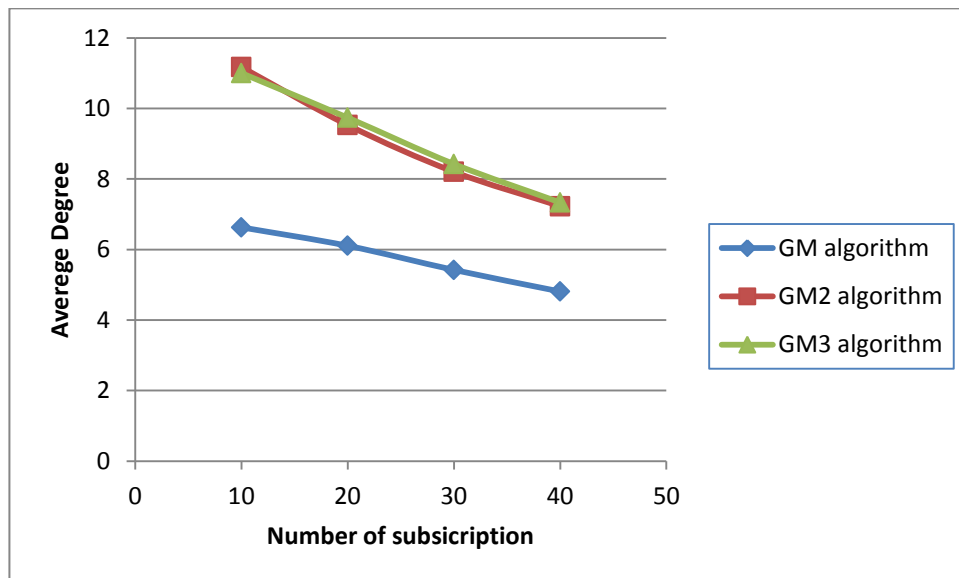
Figure 9 compares the GM, GM2 and the GM3 algorithms depending on the maximum degree. The rise of the size of the subscription increases the correlation of the node. In this case, the overlay network's maximum degree for the GM, GM2 and the GM3 algorithms decrease. They are mainly affected by the rise of the subscription rise. GM3 becomes better than a GM by 13% and GM2 by 34% when compared to other algorithms Figure 9.

Figure 10 compares the GM, GM2 and GM3 algorithms depending on their average degree. The degree of the average overlay network decreases when the algorithms spot edges that have a higher correlation. This is applicable for all the three algorithms. GM beats them by 36% when compared to the GM2 and GM3 algorithm

as illustrated in Figure 10.



**Figure 9** Maximum node degree for different subscription size (Number of nodes (200) and number of topics is 100).



**Figure 10** Average node degree for different subscription size (Number of nodes (200) and number of topics is 100).

## CHAPTER 6

### CONCLUSIONS

In this thesis, we study a new optimization problem (Fault Tolerant Overlay Networks Design) that constructs a practical and scalable overlay network for publish/subscribe communication with many topics. We presented a topic 2-connected overlay network design algorithm GM3 which is heuristic for the Fault Tolerant Overlay Networks Design problem.

We presented a polynomial-time overlay design algorithms, GM and GM2, we design our algorithm for the Fault-Tolerant Overlay Networks Design problem, namely the GM3 algorithm, especially for highly correlated pub/sub workloads.

Our experimental results validate our formal analysis for our algorithm (GM3), the obtain result show that the maximum degree resulting from our algorithm GM3 is best than which obtained by GM and GM2 algorithms. When we compare the results of GM, GM2 and GM3 algorithms, GM3 improves GM by 38% and GM2 by 56%.

When we compare GM, GM2 and GM3 algorithms according to average node degree, average node degree produce by GM is better than that generated by the GM2 and GM3 by 40% .

In subscription experiment, first we compare GM, GM2 and GM3 algorithms according to the maximum degree. When the size of the subscription increase , the overlay network's maximum degree decrease for the GM, GM2 and the GM3 algorithms. GM3 is better than a GM by 13% and GM2 by 34%. Second comparison on the average degree of the overlay network. GM beats GM2 and GM3 algorithms by 36%.

There are several things important lines in the work to be design efficient distributed algorithms for the MinMax-2TCO problem, and to look at this problem under the line of a dynamic configuration of the node set  $V$  and the interest assignment  $I$ , our designed algorithms are capable of achieving more reliable topic 2-connectivity by compromising the maximum node degree and average node degrees insignificantly.

## REFERENCES

- 1- **Eugster P. T., (2003)**, "*The Many Faces of Publish/Subscribe*", ACM Computing Surveys (CSUR), vol.35, no.2, pp. 114-131.
- 2- **Anceaume E., (2006)**, "*A Semantic Overlay for Self-Peer-to-Peer Publish/Subscribe*", Distributed Computing Systems ICDCS, 26th IEEE International Conference, pp. 22-22.
- 3- **Baehni S., (2004)**, "*Data-Aware Multicast*", IEEE, DSN, International Conference, pp. 233-242.
- 4- **Baldoni R., (2007)**, "*TERA: Topic-Based Event Routing for Peer-To-Peer Architectures*", 1st international conference on Distributed event-based systems (DEBS), ACM, vol.6, pp. 2-13.
- 5- **Banerjee S., (2002)**, "*Scalable Application Layer Multicast*", ACM, vol.32, no.4, pp. 205-217.
- 6- **Bhola S., (2002)**, "*Exactly-Once Delivery in A Content-Based Publish-Subscribe System*", IEEE, Dependable Systems and Networks, DSN, International Conference, pp. 7-16.
- 7- **Carzaniga A., (2004)**, "*A Routing Scheme for Content-Based Networking*", INFOCOM, Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, vol.2, pp. 918-928.
- 8- **Castro M., (2002)**, "*Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure*", Selected Areas in Communications, IEEE Journal, vol.20, no.8, pp. 1489-1499.

- 9- **Chand R., (2005)**, “*Semantic Peer-To-Peer Overlays for Publish/Subscribe Networks*”, *Euro-Par 2005 Parallel Processing*, Springer Berlin Heidelberg, vol.3648, pp. 1194-1204.
  
- 10- **Guerraoui R., (2004)**, “*Gossip: A Gossip-Based Structured Overlay Network for Efficient Content-Based Filtering*”, No. LPD-REPORT-005).
  
- 11- **Levis R., (1999)**, “*Advanced Messaging Applications with MSMQ and MQSeries*”, QUE.
  
- 12- **Voulgaris S., (2006)**, “*Sub-2-Sub: Self-Organizing Content-Based Publish Subscribe for Dynamic Large Scale Collaborative Networks*”, IPTPS.
  
- 13- **Liu H., (2005)**, “*Client Behavior and Feed Characteristics of RSS, A Publish-Subscribe System for Web Micronews*”, The 5th ACM SIGCOMM conference on Internet Measurement, pp. 3-3.
  
- 14- **Chockler G., (2007)**, “*Constructing Scalable Overlays for Pub-Sub with Many Topics: Problems, Algorithms, and Evaluation*”, the twenty-sixth annual ACM symposium on Principles of distributed computing, PODC, pp. 109-118.
  
- 15- **Chockler G., (2007)**, “*SpiderCast: A Scalable Interest-Aware Overlay for Topic-Based Pub/Sub Communication*”, 1st International Conference on Distributed Event-Based Systems (DEBS). ACM, vol.6, pp. 14-25.
  
- 16- **Hughes D., (2005)**, “*Free Riding on Gnutella Revisited: The Bell Tolls?*”, Distributed Systems Online, IEEE, vol.6, no.6, pp. 1.
  
- 17- **Bo CW., (2003)**, “*Peer-to-Peer Overlay Networks: A Survey*”, Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong, Available from: [citeseer.ist.psu.edu/706822.html](http://citeseer.ist.psu.edu/706822.html), pp. 9.
  
- 18- **Cohen E., (2002)**, “*Replication Strategies in Unstructured Peer-to-Peer Networks*”, ACM SIGCOMM Computer Communication Review, vol.32, no.4, pp. 177-190.

- 19- Stoica I., (2001),** “*Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*”, ACM SIGCOMM Computer Communication Review, vol.31, no.4, pp. 149-160.
- 20- Ratnasamy S., (2001),** “*A Scalable Content Addressable Network*”, ACM SIGCOMM, vol.31, no.4, pp. 161-172.
- 21- Maymounkov P., (2002),** “*Kademlia: A Peer-to-Peer Information System Based on The XOR Metric*”, 1st International Workshop on Peer-to-Peer Systems, Springer Berlin Heidelberg, pp. 53-65.
- 22- Hauswirth M., (2005),** “*Peer-to-Peer: Grundlagen und Architektur*”, Datenbank-Spektrum, vol.5, no.13, pp. 5–13.
- 23- <http://freenetproject.org>,** (Data Download Date: 05.04.2015).
- 24- <http://www.softwax.com>,** (Data Download Date: 01.05.2015).
- 25- <http://www.napster.com>,** (Data Download Date: 01.05.2015).
- 26- <http://www.kazaa.com/us/index.htm>,** (Data Download Date: 02.05.2015).
- 27- <http://www.jxta.org>,** (Data Download Date: 04.05.2015).
- 28- <http://www.intel.com>,** (Data Download Date: 05.05.2015).
- 29- [http://www.ipoque.com/en/p2p\\_filter.html](http://www.ipoque.com/en/p2p_filter.html),** (Data Download Date: 01.05.2015).
- 30- <http://www.boincstats>,** (Data Download Date: 01.05.2015).
- 31- Mauthe A., (2003),** “*Peer-to-Peer Computing: Systems, Concepts and Characteristics*”, Praxis der Informationsverarbeitung & Kommunikation, vol.26, no.2, pp. 60-64.



- 32-Ledlie J., (2002),** “*Self-Organization in Peer-to-Peer Systems*”, The 10th workshop on ACM SIGOPS European workshop, pp. 125-132.
- 33-Milojicic DS., (2002),** “*Peer to Peer Computing*”, HP Laboratories Palo Alto, vol.1.
- 34- <http://setiathome.ssl.berkeley.edu/>,** (Data Download Date: 01.05.2015).
- 35- Carzaniga A., (2000),** “*Achieving Scalability and Expressiveness in An Internet-Scale Event Notification Service*”, The nineteenth annual ACM symposium on Principles of distributed computing, pp. 219-227.
- 36-Strom R., (1998),** “*Gryphon: An Information Flow Based Approach to Message Brokering*”, in International Symposium on Software Reliability Engineering, Citeseer.
- 37-Pietzuch P., (2002),** “*Hermes: A Distributed Event-Based Middleware Architecture*”, Distributed Computing Systems Workshops, 22nd International Conference, IEEE, pp. 611-618.
- 38-Ramasubramanian V., (2006),** “*Corona: A High Performance Publish-Subscribe System for The World Wide Web*”, NSDI, Vol.6, pp. 2-2.
- 39- Castro M., (2002),** “*Scribe: A Large Scale and Decentralized Application-Level Multicast Infrastructure*”, Selected Areas in Communications, IEEE Journal, vol.20, no.8, pp. 1489-1499.
- 40-Zhuang S., (2001),** “*Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination*”, ACM, The 11th international workshop on Network and operating systems support for digital audio and video, pp. 11-20.
- 41-Patel J., (2009),** “*Rappel: Exploiting Interest and Network Locality to Improve Fairness in Publish-Subscribe Systems*”, Computer Networks, vol.53, no.13, pp. 2304-2320.
- 42-Matos M., (2010),** “*Stan: Exploiting Shared Interests without Disclosing Them in Gossip-Based Publish/Subscribe*”, IPTPS, pp. 9.

- 43- Chockler G., (2007),** “*Spider Cast: A Scalable Interest-Aware Overlay for Topic-Based Pub/Sub Communication*”, ACM, Inaugural international conference on Distributed event-based systems, pp. 14-25.
- 44- Wong B., (2008),** “ *Quasar: A Probabilistic Publish-Subscribe System for Social Networks*”, IPTPS, pp. 2.
- 45- Tock Y., (2005),** “*Hierarchical Clustering of Message Flows in A Multicast Data Dissemination System*”, IASTED PDCS, pp. 320-326.
- 46- Zhao B., (2001),** “*Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing*”, Computer, vol.74, pp. 11-20.
- 47- Girdzijauskas S., (2010),** “*Magnet: Practical Subscription Clustering for Internet-Scale Publish/Subscribe*”, ACM, The Fourth ACM International Conference on Distributed Event-Based Systems, pp. 172-183.
- 48- Girdzijauskas S., (2008),** “*Gravity: An Interest-Aware Publish/Subscribe System Based on Structured Overlays*”, DEBS, vol. 8.
- 49- Alveirinho J., (2010),** “*Flexible and Efficient Resource Location in Large-Scale Systems*”, ACM, The 4th International Workshop on Large Scale Distributed Systems and Middleware, pp. 55-60.
- 50- Rahimian F., (2011),** “*Vitis: A Gossip-Based Hybrid Overlay for Internet-Scale Publish/Subscribe Enabling Rendezvous Routing in Unstructured Overlay Networks*”, IEEE, Parallel & Distributed Processing Symposium (IPDPS), pp. 746-757.
- 51- Gupta A., (2004),** “*Meghdoot: Content-Based Publish/Subscribe over P2P Networks*”, The 5th ACM/IFIP/ USENIX international conference on Middleware, Springer-Verlag New York, Inc, pp. 254-273.
- 52- Ratnasamy S., (2001),** “ *A Scalable Content-Addressable Network*”, ACM. Chicago, vol.31, no.4, pp. 161-172.

- 53-Voulgaris S., (2006),** *“Sub-2-Sub: Self-Organizing Content-Based Publish and Subscribe for Dynamic and Large Scale Collaborative Networks”*, IPTPS, The fifth International Workshop on Peer-to-Peer Systems, Citeseer, pp.16 .
- 54-Zhang R.,(2005),** *“Hyper: A Hybrid Approach to Efficient Content-Based Publish/Subscribe”*, IEEE, Distributed Computing Systems, ICDCS, The 25th IEEE International Conference, pp. 427-436.
- 55-Zhu Y., (2005),** *“Ferry: An Architecture for Content-Based Publish/Subscribe Services on P2P Networks”*, IEEE, Parallel Processing, ICPP, International Conference, pp. 427-434.
- 56-Yang X., (2007),** *“Scalable Content-Based Publish/Subscribe Services over Structured Peer-to-Peer Networks”*, IEEE, Parallel, Distributed and Network-Based Processing, PDP'07, The 15th EUROMICRO International Conference , pp. 171-178.
- 57-Pujol-Ahullo J., (2009),** *“Towards A Lightweight Content-Based Publish/Subscribe Services for Peer-to-Peer Systems”*, International Journal of Grid and Utility Computing, vol.1, no.3, pp. 239-251.
- 58-Fox E., (1991),** *“Order-Preserving Minimal Perfect Hash Functions and Information Retrieval”*, ACM Transactions on Information Systems (TOIS), vol.9, no.3, pp. 281 -308.
- 59-Czech Z., (1992),** *“An Optimal Algorithm for Generating Minimal Perfect Hash Functions”*, Information Processing Letters, vol.43, no.5, pp. 257-264.
- 60-Chockler G., (2007),** *“Constructing Scalable Overlays for Pub-Sub with Many Topics”*, ACM, The Twenty-Sixth Annual ACM Symposium On Principles Of Distributed Computing, pp. 109-118.

## APPENDICES A

### CURRICULUM VITAE

#### PERSONAL INFORMATION

**Surname, Name:** AL-AGELE, Raad

**Date and Place of Birth:** 28 November 1981, Waset

**Marital Status:** Married

**Phone:** +90 5346198116

**Email:** [raad1aziz@yahoo.com](mailto:raad1aziz@yahoo.com)



#### EDUCATION

| Degree      | Institution                                               | Year of Graduation |
|-------------|-----------------------------------------------------------|--------------------|
| M.Sc.       | Çankaya University<br>Mathematics and Computer<br>Science | 2015               |
| B.Sc.       | Al-Rafidain University<br>College, Computer Science       | 2005               |
| High School | Al-Suwayra High School                                    | 2000               |

#### WORK EXPERIENCE

| Year          | Place                          | Enrollment        |
|---------------|--------------------------------|-------------------|
| 2006- Present | Technical Institute Al-Suwayra | Technical Trainer |

#### FOREIN LANGUAGES

English.

#### HOBBIES

Books, Football, Travel, Swimming.