

PEER-TO-PEER NETWORKING WITHIN REPUTATION AGGREGATION USING DIFFERENTIAL GOSSIP ALGORITHM

A mini project report submitted to

Jawaharlal Nehru Technological University

In partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

BOTLA SINDHURA :14831A0525

BYKANI RAJU :14831A0527

ABHISHEK JHAWAR :14831A0504

Under the guidance of

Dr.S.SENTHIL KUMAR

(Professor & Head of the Department)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK INSTITUTE OF TECHNOLOGY

Ibrahimpattanam (v), Rangareddy(Dist), Telangana, 501506

April,2018



GURU NANAK INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project entitled **PEER-TO-PEER NETWORKING WITHIN REPUTATION AGGREGATION USING DIFFERENTIAL GOSSIP ALGORITHM** is being submitted by **Ms.BOTLA SINDHURA** bearing Roll No : **14831A0525**, **Mr.BYKANI RAJU** bearing Roll No : **14831A0527** and **Mr.ABHISHEK JHAWAR** bearing Roll No : **14831A0504**, in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Technological University is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this Mini-project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Internal Guide

(Dr.S.Senthil Kumar)

Professor & Head of Department

Project Coordinator

(Mr.N.SriHari Rao)

Associate Professor

(Dr.S.Senthil Kumar)

External Examiner

Head of Department

City Office: B2, 2nd Flr, Above Bata, Vikramপুরi Colony, Karkhana Road, Secunderabad- 500009, Telangana, India.

Ph: +91-40-66323294, Fax: +91-40-27892633

Campus: Ibrahimpatnam, R.R.District, Hyderabad- 501506, Telangana, India. Ph: (0/95) 8414-202120/21

DECLARATION

We hereby declare that mini project entitled “**PEER-TO-PEER NETWORKING WITHIN REPUTATION AGGREGATION USING DIFFERENTIAL GOSSIP ALGORITHM**” is the work done by **BOTLA SINDHURA** bearing Roll No: **14831A0525**, **BYKANI RAJU** bearing Roll No: **14831A0527** and **ABHISHEK JHAWAR** bearing Roll No: **14831A0504**, towards fulfilment of the requirement for the award of the Degree of Computer Science and Engineering to Jawaharlal Nehru Technological University, Hyderabad for A.Y:2017-18, is the result of the work carried out under the guidance of **Dr.S.SENTHIL KUMAR, HEAD OF DEPARTMENT**, Guru Nanak Institute of Technology, Hyderabad.

We further declare that this project report has not been previously submitted before either in part or full award of any degree or any organisation or any universities.

ACKNOWLEDGEMENT

It is indeed our earnest duty to thank all the people who has instrumented us for the completion of our project work. First and foremost,we proudly thank the almighty for the unfathomable blessings showered on us on establishing our project sucessfully.We also thank parents for directing us in glorious path by giving moral courage and pellucid help.

As the outlet,we express our profound gratitude to our Principal **Dr.S.Sreenatha Reddy** an adroitness person who has paved many provisions to complete this project.

We can't find an apt word to express our deep sense of reverence and gratitude to Prof. **Dr.S.Senthil kumar**, Head of the Department of Bachelor of Technology in Computer Science & Engineering and our internal guide for his valuable and timely suggestions whose constant encouragment and cooperation has made successful in completing the project.

We specially thank Project Coordinator **Mr.N.SriHari Rao**, Associate Professor CSE, GNIT for providing seamless support and right suggestions that are given in the development of the project.

Finally, we thank from any heart,all the officals in Depatment of Computer Science & Engineering ,friends and people who spread their valuable time and knowlegde to complete this project.

B.SINDHURA(14831A0525)

B.RAJU(14831A0527)

ABHISHEK(14831A0504)

ABSTRACT

Peer-to-peer (P2P) is a decentralized communications model in which each party has the same capabilities and either party can initiate a communication session. Unlike the client/server model, in which the client makes a service request and the server fulfills the request, the P2P network model allows each node to function as both a client and server. Tendency of nodes to draw resources from the network and not giving anything in return is termed as Free Riding. Usually the nodes will have conflict of interests, thus the selfish behavior of nodes leads to the problem of free riding.

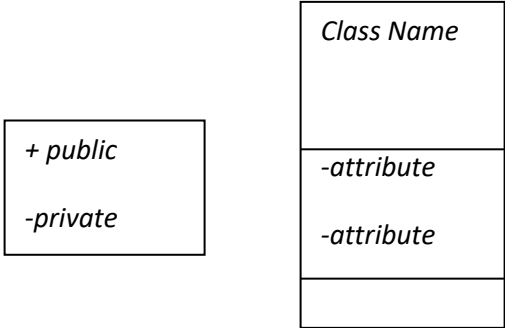
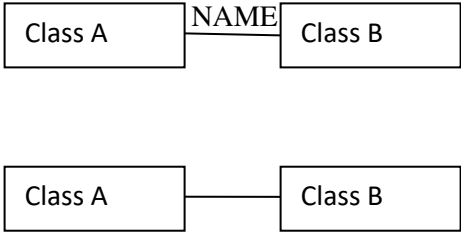
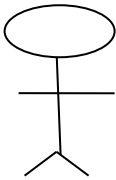
LIST OF TABLES

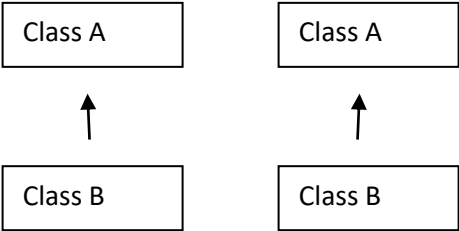
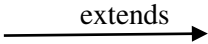

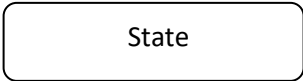

T.NO.	NAME OF THE TABLE	PAGE NO.
5.3	Register Table	50
5.3	Source Table	50

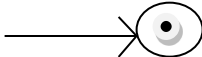
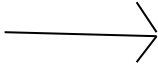
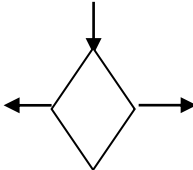
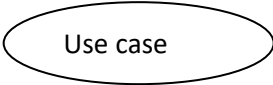
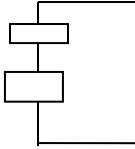
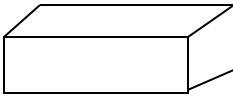
LIST OF FIGURES

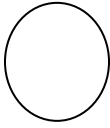
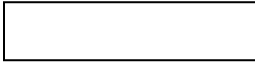


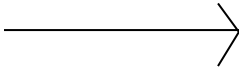
Figure no	Name of the figure	Page no.
3.4.3	Module Diagram	16
4.1.1	Use Case Diagram	23
4.1.2	Class Diagram	24
4.1.3	Object Diagram	25
4.1.4	State Diagram	26
4.1.5	Activity Diagram	27
4.1.6	Sequence Diagram	28
4.1.7	Collaboration Diagram	29
4.1.8	Component Diagram	30
4.1.9	Data Flow Diagrams	31
4.1.10	E-R Diagram	33

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Associations represent static relationships between classes. Roles represent the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.

4.	Aggregation		Interaction between the system and external environment
5.	<i>Relation</i> (uses)	<i>uses</i>	Used for additional process communication.
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the process.
9.	Initial State		Initial state of the object

10.	Final state		F final state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Use case		Interaction between the system and external environment.
14.	Component		Represents physical modules which are a collection of components.
15.	Node		Represents physical modules which are a collection of components.

16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard, sensors, etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message	Message 	Represents the message exchanged.

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	EXPANSION
1.	DB	Data Base
2.	JVM	Java Virtual Machine
3.	JSP	Java Server Page
4.	CB	Collective Behavior
5.	SD	Social Dimension
6.	JRE	Java Runtime Environment
7.	SSD	Sparse Social Dimension
8.	LGP	Line Graph Partition

TABLE OF CONTENTS

Content	Page No.
Certificate.....	i
Declaration.....	ii
Acknowledgement.....	iii
Abstract.....	iv
List of Tables.....	v
List of Figures.....	vi
List of Symbols.....	vii
List of Abbreviations.....	xi
1. INTRODUCTION.....	1
1.1 General.....	1
1.2 Objective.....	1
1.3 Existing System.....	2
1.3.1 Existing System Disadvantages.....	2
1.4 Proposed System.....	3
1.4.1 Proposed System Advantages	3
2. LITERATURE REVIEW.....	4
3. METHODOLOGY.....	10
3.1 General.....	10
3.2 System Requirements.....	10

3.3 Software Description.....	11
3.4 Methodologies.....	13
3.4.1 Modules Name.....	13
3.4.2 Modules Explanation.....	14
3.4.3 Module Diagram	16
3.4.4 Expected Output.....	18
3.5 Technique or Algorithm.....	19
3.6 System Architecture.....	21
4. MODELLING.....	22
4.1 General.....	22
4.1.1 Use Case Diagram.....	24
4.1.2 Class Diagram.....	25
4.1.3 Object Diagram.....	26
4.1.4 State Diagram.....	27
4.1.5 Activity Diagram.....	28
4.1.6 Sequence Diagram.....	29
4.1.7 Collaboration Diagram.....	30
4.1.8 Component Diagram.....	31
4.1.9 Data Flow Diagrams.....	32
4.1.10 E-R Diagram.....	33
5. IMPLEMENTATION & TESTING.....	34
5.1 General.....	34
5.2 Sample Code.....	34
5.3 Database Tables.....	50
5.4 Test Cases.....	51
5.5 Developing Methodologies.....	51

5.6 Types of Testing.....	51
6. RESULT ANALYSIS.....	54
7. CONCLUSION AND FUTURE ENHANCEMENT.....	58
7.1 Conclusion.....	58
7.2 Future Enhancement.....	58
8. REERENCES.....	60

CHAPTER 1

INTRODUCTION

1.1 GENERAL:

Peer-to-peer (P2P) is a decentralized communications model in which each party has the same capabilities and either party can initiate a communication session. Unlike the client/server model, in which the client makes a service request and the server fulfills the request, the P2P network model allows each node to function as both a client and server. Tendency of nodes to draw resources from the network and not giving anything in return is termed as Free Riding. Usually the nodes will have conflict of interests, thus the selfish behavior of nodes leads to the problem of free riding.

1.2 OBJECTIVE:

In our proposed system estimate of reputation is considered to be having two parts, one common component which is same with every node, and the other one is the information received from immediate neighbors based on the neighbors' direct interaction with the node. The differential gossip is fast and requires lesser amount of resources. This mechanism allows computation of independent reputation value by every node, of every other node in the network. The differential gossip trust has been investigated for a power law network formed using preferential attachment (PA) Model. The reputation computed using differential gossip trust shows good amount of immunity to the collusion.

1.3 EXISTING SYSTEM:

Existing peer-to-peer system, there is no Server. Each node acts as a server as well as a client. Free riding has emerged as a big challenge for peer-to-peer systems. Tendency of nodes to draw resources from the network and not giving anything in return is termed as Free Riding. Aggregation of trust generally consumes a lot of time and memory especially with large number of nodes.

1.3.1 EXISTING SYSTEM DISADVANTAGES:

- Generally consumes a lot of time and memory especially with large number of nodes.
- When reputation management systems are deployed, rogue peers will start using collusion to bypass it.

1.4 PROPOSED SYSTEM:

In our proposed system we have been proposed we have proposed an algorithm that discourages the free riding and reduces the collusion significantly in networks with power law degree distribution. The proposed algorithm does not require any central server or repository. The communication and computation cost for trust aggregation is low, and the aggregation is normally completed in reasonable time.

1.4.1 PROPOSED SYSTEM ADVANTAGES:

- Generally simple, lightweight and robust for Errors.
- Discourages the free riding and reduces the collusion significantly in networks with power law degree distribution.

CHAPTER 2

LITERATURE REVIEW

TITLE : Overcoming Free-Riding Behavior in Peer-to-Peer Systems.
AUTHOR : MICHAL FELDMAN and JOHN CHUANG
YEAR : 2005

DESCRIPTION

The peer-to-peer (P2P) communications model has emerged a widely deployed alternative to the traditional client-server model for many distributed systems. In atypical P2P system, each node is owned and operated by an independent entity, and the nodes collectively form a self-organizing, self-maintaining network with no central authority. As a result, P2P system performance is highly dependent on the amount of voluntary resource contribution from the individual nodes.

TITLE : Non-Cooperation in Competitive P2P Networks
AUTHOR : Beverly Yang, Tyson Condie, Sepandar Kamvar, Hector Garcia-Molina
YEAR : July 28, 2010

DESCRIPTION

Large-scale competitive P2P networks are threatened by the noncooperation problem, where peers do not forward queries to potential competitors. While non-cooperation is not a problem in current P2Pfree file-sharing networks, it is likely to be a problem in such applications as pay-per-transaction file-sharing, P2P auctions, and P2P service discovery networks, where peers are in competition with each other to provide services. Here, we show how non-cooperation causes unacceptable degradation in quality of results, and present an economic protocol to address this problem. This protocol, called

the RTR protocol, is based on the buying and selling of the right-to-respond (RTR) to each query in the network. Through simulations we show how the RTR protocol not only overcomes non-cooperation by providing proper incentives to peers, but also results in a network that is even more effective and efficient through intelligent, incentive-compatible routing of messages.

TITLE : Trust Based Incentive in P2P Network

AUTHOR : Yang Bin Tang, Huai Min Wang, Wen Dou

YEAR : 2002

DESCRIPTION

In the world of Internet technologies, Peer-to-Peer (P2P) computing is currently receiving considerable interest. However, as experience with P2P networks such as Gnutella shows, the selfish behaviors of peers may lead to serious problems of P2P network, such as Free-riding and Tragedy of Commons. In order to solve these problems, there are increasingly considerations on incentive design in the study of P2P systems. In this paper, we give a brief survey of researches on trust based incentive in P2P network. By investigating the reputation systems in P2P networks, we outline some key issues within the design of trust based incentive in P2P networks. After that we introduce some other approaches addressing the incentive of P2P networks and conclude the paper with a summary and discussion on future research.

TITLE : Free Riding on Gnutella

AUTHOR : Eytan Adar and Bernardo A. Huberman

YEAR : June 13, 2006

DESCRIPTION

An extensive analysis of user traffic on Gnutella shows a significant amount of free riding in the system. By sampling messages on the Gnutella network over a 24-hour period, we established that nearly 70% of Gnutella users share no files, and nearly 50% of all responses are returned by the top 1% of sharing hosts. Furthermore, we found out that free riding is distributed evenly between domains, so that no one group contributes significantly more than others, and that peers that volunteer to share files are not necessarily those who have desirable ones. We argue that free riding leads to degradation of the system performance and adds vulnerability to the system. If this trend continues copyright issues might become moot compared to the possible collapse of such systems.

TITLE : Free Riding on Gnutella Revisited: the Bell Tolls

AUTHOR : Daniel Hughes, Geoff Coulson, James Walkerdine

YEAR : April 2, 2006

DESCRIPTION

Since the release of Napster [1] in 1999, peer-to-peer computing has rapidly been growing in popularity. Napster is an example of a *semi-centralized* peer-to-peer system [2]. However, due to legal and scalability problems, semi-centralized peer-to-peer file sharing systems have largely been replaced by fully decentralized file-sharing networks, such as Gnutella [3]. Peer-to-peer file-sharing networks like Gnutella embody a social dilemma wherein the behavior of rational users is at odds with the common good. The dilemma for each individual is whether to contribute to the common good by sharing files, or to maximize their personal experience by ‘freeriding’ (i.e. Downloading files while not contributing any to the network). As there is no personal benefit gained by uploading files (in fact it is inconvenient), it is ‘rational’ for users to free ride. However, significant numbers of free riders degrade the utility of the entire system. This is known as ‘the tragedy of the digital commons’

TITLE : Analysis of Large-Scale Peer-to-Peer Network Topology

AUTHOR : Chao Xie

YEAR : April 2, 2008

DESCRIPTION

Modeling peer-to-peer (P2P) networks is a challenge for P2P researchers. It is the key to provide insight into the nature of the underlying system and building a successful model able to generate realistic topologies for simulation purpose. In this paper, we provide a detailed analysis of large-scale P2P network topology, using Gnutella as a case study. First, we re-examine the power-law distributions of the Gnutella network discovered by previous researchers. Our results show that the current Gnutella network deviates from the early power-laws, suggesting that the Gnutella network topology may have evolved a lot over time. Second, we identify important trends with regard to the evolution of the Gnutella network between September 2005 and February 2006. Third, we provide a novel two-layered approach to study the topology of the Gnutella network. Due to the limitations of the power-laws, we divide the Gnutella network into two layers, namely the mesh and the forest, to model the hybrid and highly dynamic architecture of the current Gnutella network. We give detailed analysis of the topology of the mesh and present four power-laws concerning the mesh topology. Moreover, we examine the topology properties of the forest and provide one empirical law concerning the tree size. Using the two-layered approach and laws proposed, we can generate realistic topologies easily.

TITLE : Small-World File-Sharing Communities

AUTHOR : Adriana Iamnitchi, Matei Ripeanu, Ian Foster

YEAR : 2009

DESCRIPTION

Web caches, content distribution networks, peer-to-peer file sharing networks, distributed file systems, and data grids all have in common that they involve community of users who generate requests for shared data. In each case, overall system performance

can be improved significantly if we can first identify and then exploit interesting structure within a community's access patterns. To this end, we propose a novel perspective on file sharing based on the study of the relationships that form among users based on the files in which they are interested. We propose a new structure that captures common user interests in data—the data-sharing graph— and justify its utility with studies on three data-distribution systems: a high-energy physics collaboration, the Web, and the Kazaa peer-to-peer network. We find small-world patterns in the data-sharing graphs of all three communities. We analyze these graphs and propose some probable causes for these emergent small-world patterns. The significance of small world patterns is twofold: it provides a rigorous support to intuition and, perhaps most importantly, it suggests ways to design mechanisms that exploit these naturally emerging patterns.

TITLE : Emergence of Scaling in Random Networks

AUTHOR : Albert-László Barabási and Réka Albert

YEAR : 2011

DESCRIPTION

Systems as diverse as genetic networks or the world wide web are best described as networks with complex topology. A common property of many large networks is that the vertex connectivity's follow a scale-free power-law distribution. This feature is found to be a consequence of the two generic mechanisms that networks expand continuously by the addition of new vertices, and new vertices attach preferentially to already well connected sites. A model based on these two ingredients reproduces the observed stationary scale free distributions, indicating that the development of large networks is governed by robust self-organizing phenomena that go beyond the particulars of the individual systems.

TITLE : The Eigen Trust Algorithm for Reputation Management in P2P Networks

AUTHOR : Sepandar D. Kamvar, Mario T. Schlosser, Hector Garcia Molina

YEAR : May 2, 2004

DESCRIPTION

Peer-to-peer file-sharing networks are currently receiving much attention as a means of sharing and distributing information. However, as recent experience shows, the anonymous, open nature of these networks offers an almost ideal environment for the spread of self-replicating inauthentic files. We describe an algorithm to decrease the number of downloads of inauthentic files in a peer-to-peer file-sharing network that assign each peer a unique global trust value, based on the peer's history of uploads. We present a distributed and secure method to compute global trust values, based on Power iteration. By having peers use these global trust values to choose the peers from whom they download, the network effectively identifies malicious peers and isolates them from the network. In simulations, this reputation system, called Eigen Trust, has been shown to significantly decrease the number of inauthentic files on the network, even under a variety of conditions where malicious peers cooperate in an attempt to deliberately subvert the system.

TITLE : Trusted P2P Transactions with Fuzzy Reputation Aggregation

AUTHOR : Shanshan Song, Kai Hwang, and Runfang Zhou

YEAR : 2013

DESCRIPTION

The Internet has enabled e-commerce and e-auctions for online commodity exchanges among strangers worldwide. Many companies now offer e-auction platforms to facilitate such transactions, including eBay, Amazon, uBid, and Yahoo. A growing number of online commercial transactions occur in a peer-to-peer (P2P) environment, which doesn't require a central authority to mediate such exchanges. Rather, participating peers can sign in and out asynchronously at will and perform their transactions point-to-point or point-to-multipoint anonymously. Thus, a business or enterprise must have an effective reputation system to help users locate trustworthy partners and exchange commodities securely with confidence.

CHAPTER 3

PROJECT DESCRIPTION

3.1 GENERAL

Peer-to-peer (P2P) is a decentralized communications model in which each party has the same capabilities and either party can initiate a communication session. Unlike the client/server model, in which the client makes a service request and the server fulfills the request, the P2P network model allows each node to function as both a client and server. Tendency of nodes to draw resources from the network and not giving anything in return is termed as Free Riding. Usually the nodes will have conflict of interests, thus the selfish behavior of nodes leads to the problem of free riding.

3.2 SYSTEM REQUIREMENTS

Hardware Requirements

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

PROCESSOR	:	PENTIUM IV 2.6 GHz, Intel Core 2 Duo.
RAM	:	512 MB DD RAM
MONITOR	:	15" COLOR
HARD DISK	:	40 GB

Software Requirements

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for

creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's and tracking the team's progress throughout the development activity.

Front End	:	JAVA(Swing)
Back End	:	MY SQL 5.5
Operating System	:	Windows 07
IDE	:	Eclipse

3.3 SOFTWARE DESCRIPTION

The Java Framework

Java is a programming language originally developed by James Gosling at Sun Micro systems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applications. The java framework is a new platform independent that simplifies application development internet. Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific super computers, cell phones to the Internet, Java is everywhere!

Objectives Of Java

To see places of Java in Action in our daily life, explore java.com.

Why Software Developers Choose Java

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the

planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

- Write software on one platform and run it on virtually any other platform
- Create programs to run within a Web browser and Web services
- Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
- Combine applications or services using the Java language to create highly customized applications or services
- Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

Some Ways Software Developers Learn Java

- Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Center, and signing up for Web, virtual, or instructor-led courses.

Object Oriented

To be an Object Oriented language, any language must follow at least the four characteristics.

1. Inheritance: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding additional features as needed.

2. Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3. Polymorphism: As the name suggests one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.

4. Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

Collections

The Java Collections API's provide Java developers with a set of classes and interfaces that makes it easier to handle collections of objects. In a sense Collection's works a bit like arrays, except their size can change dynamically, and they have more advanced behavior than arrays. In this project we are using Array List for collecting the user input and saving values.

Thread

In this project threading concept is very important. A thread is a sequential path of code execution within a program. And each thread has its own local variables, program counter and lifetime. Like creation of a single thread, we can also create more than one thread (multithreads) in a program using class Thread or implementing interface Runnable to make our project efficient and dynamic. In our project we are using request process with the help of multi threading concepts.

Swings

Swing, which is an extension library to the AWT, includes new and improved components that enhance the look and functionality of GUIs. Swing can be used to build Standalone swing gui apps as well as Servlets and Applets. It employs a model/view design architecture. Swing is more portable and more flexible than AWT.

3.4 METHODOLOGIES

Following modules involves.

3.4.1 MODULES

- SERVER MONITORING
- PEER TO PEER CONSTRUCTION
- DIFFERENTIAL GOSSIP ALGORITHM

- REPUTATION AGGREGATION
- DATAMANUPULATION

3.4.2 MODULES EXPLANATION

Server Monitoring

In this module we design the windows for the project. These windows are used to send a message from one peer to another. We use the Swing package available in Java to design the User Interface. Swing is a widget toolkit for Java. It is part of Sun Microsystems' Java Foundation Classes an API for providing a graphical user interface for Java programs. In this module mainly we are focusing on the all the peer to locate the where located save monitoring construction. So easily find out the peer details and located. So message or file share easily neighbor peer find and send data manipulating the process.

Peer To Peer Construction:

Peer to Peer construction is an important role for the user to file sharing and communication with anyone. In this module has created for the security purpose for the peer to peer system. However, establishing trust in an unknown entity is difficult in such a malicious environment. Metrics are needed to represent in computational Models. Classifying peers as either trustworthy or untrustworthy is not sufficient in most cases. In the presence of an authority, a central server is a preferred way to store and manage trust information. Since there is no central server in most peer to peer systems. Then peers only trust information managed.

Differential Gossip Algorithm

In this algorithm, every node makes different number of pushes in a single gossiping step depending upon the ratio of its own degree to the average neighbor degree. If every node also pushes its degree to all the neighboring nodes, then each node can estimate the average degree of all its neighbors. Here, we make three assumptions, i) every node has a unique identification number known to every other node. So, if some

node pushes some information about another node, receiving node knows that this information is about which particular node; ii) time is discrete; and iii) every node knows about the starting time of gossip process.

Reputation Aggregation:

As the unstructured peer-to-peer networks are very similar to human social networks, the former can be mimicked as the latter. In human network, when we need the reputation value of somebody, we rely on personal experience with him. If we don't have any prior personal experience, we rely on the combination of two things: First, the general perception about him which we receive from gossip flowing around. Second, the weighted average of information given by our friends if they have any direct interaction with him where weights to information are given according to degree of friendship. Based on this, we propose a new algorithm as follows

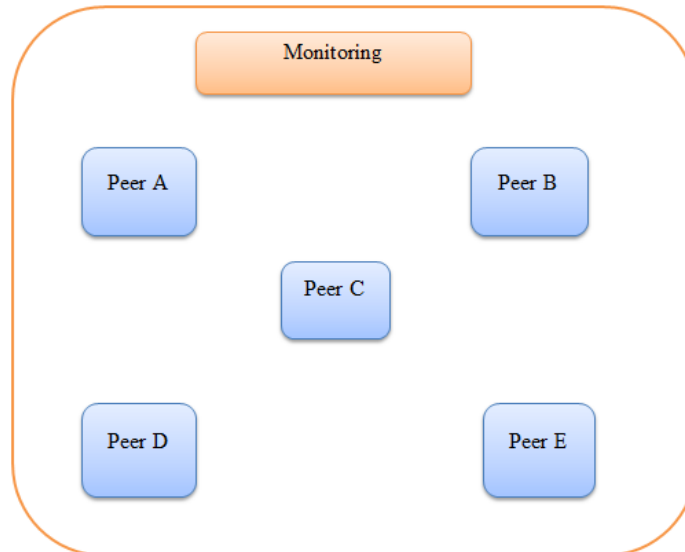
In this algorithm the nodes gather opinion of their neighbors and combine it with the opinion, obtained from general gossip after weighing the neighbors' opinion or direct feedback according to the confidence in the neighbors'. In general, it can be said that a node gives weight to every node in the network. The nodes that have not interacted with it are given weight as 1 whereas nodes that have interacted are given weight according to the confidence in them (always 1).

Data Manipulation

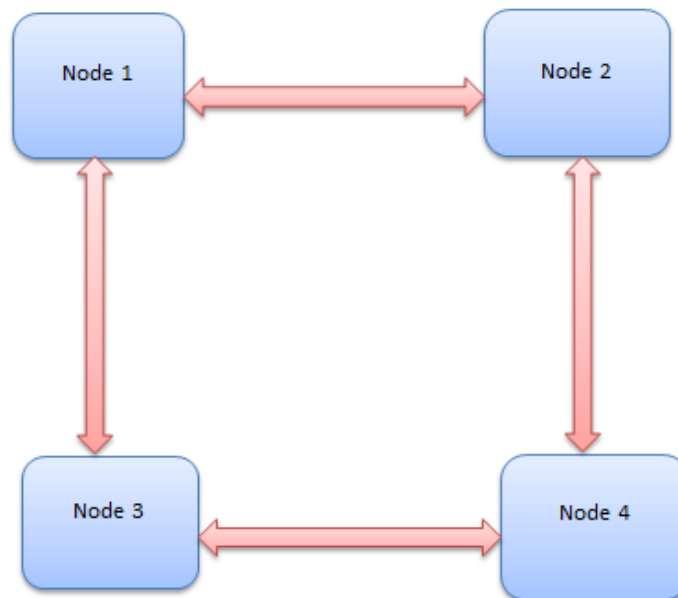
The Process of interaction between the two nodes.it means sending and receiving the data with the immunity of collusion.

3.4.3 MODULES DIAGRAM:

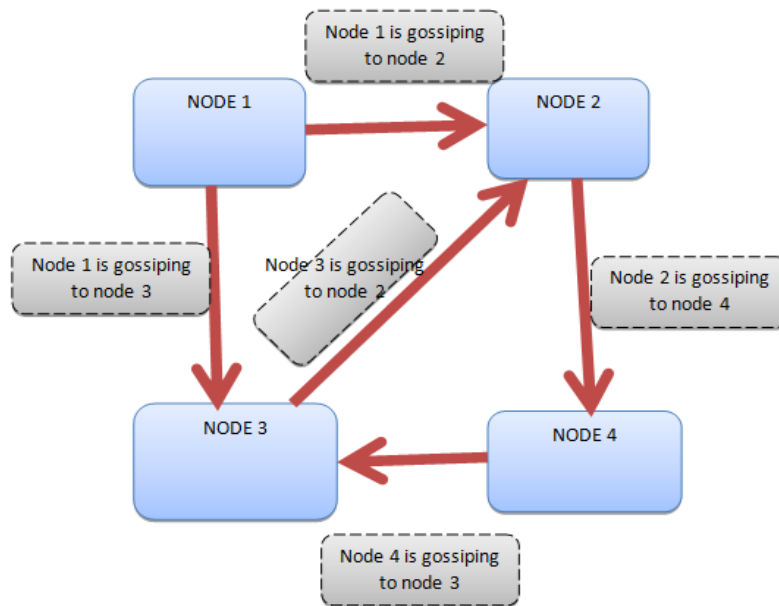
Server Monitoring



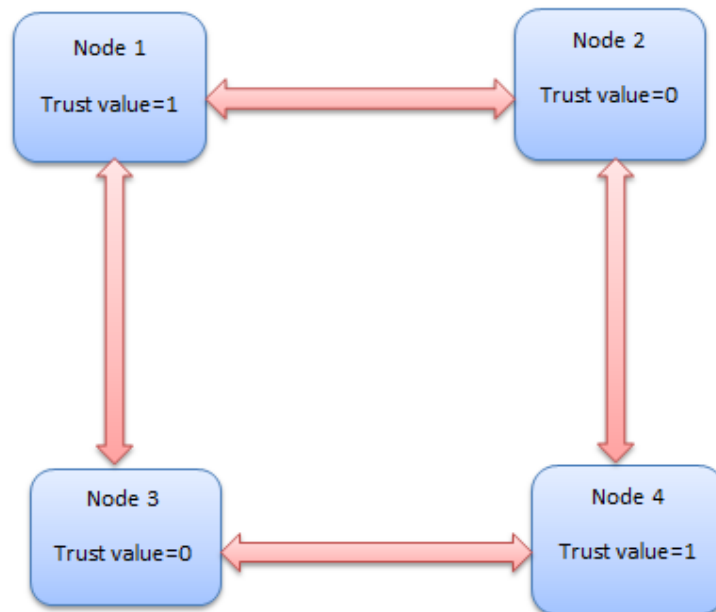
Peer To Peer Construction:



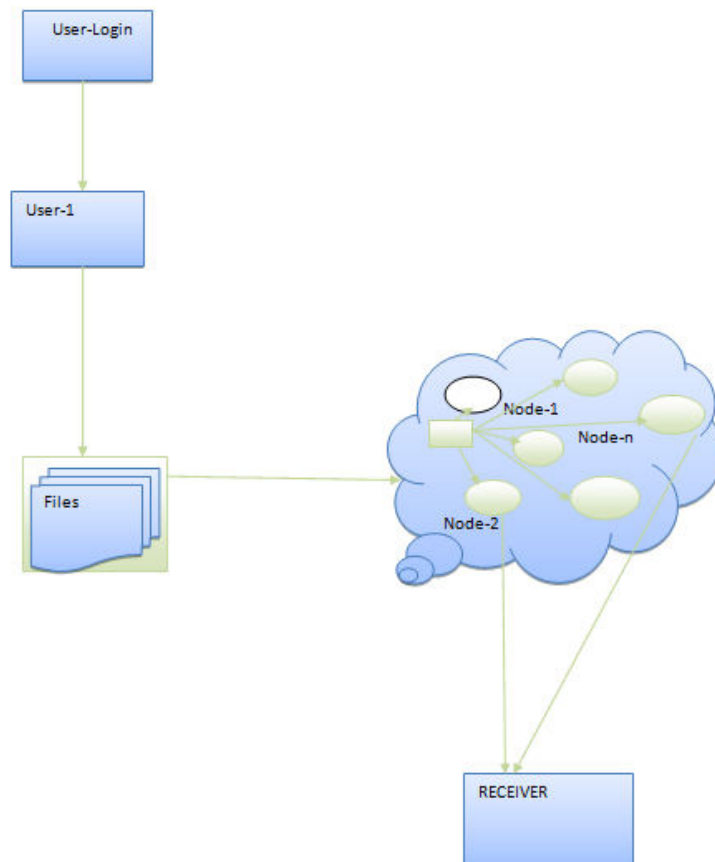
Differential Gossip Algorithm:



Reputation Aggregation



Data manipulation



3.4.4 EXPECTED OUTPUT

1. Server Monitoring

Given Input : Server Running

Output : node is connected to Server

2. Peer To Peer Construction

Input : User and Customer Login name and Password and also retrieving the database's information

Output : all the nodes are connected together

3. Differential Gossip Algorithm

Given Input : sending the data from source to destination

Output : based on reputation value.

4. Reputation Aggregation

Given Input : sending the request

Output : finding the reputation value

5. Data Manipulation

Given Input : sending the data

Output : receiving the data

3.5 TECHNIQUE OR ALGORITHM

Gossip Algorithms are used for spreading information in the large decentralized networks. These algorithms are random in nature as in these nodes randomly choose their communication partner in each information diffusion step

ALGORITHM: Differential Gossip Algorithm.

Global Reputation Aggregation for Single Node

Require: t_{ij} (The reputation estimated by node i for node j only on the basis of direct interaction)

for $1 \leq i \leq N$, gossip error tolerance ϵ ,

Ensure: Global Reputation of node j (R_j)

if i has some reputation value about j **then**

Assume weight $g_{ij} = 1$, and $y_{ij} = t_{ij}$

else

Assume weight $g_{ij} = 0$, and $y_{ij} = 0$

end if

Push self degree to neighboring node

Take the average of neighbor's degrees

Calculate the ratio of its degree and average of neighbor

degree ($k_i \leftarrow \text{degree of } i \div \text{average neighbor degree}$)

Round off k_i to nearest integer for $k_i \geq 1$ else take $k_i = 1$

$m \leftarrow 1$ { Initialize Gossip Step }

$u \leftarrow y_{ij} \div g_{ij}$ for nodes having $g_{ij} \neq 0$; otherwise $u \leftarrow -10$.

repeat

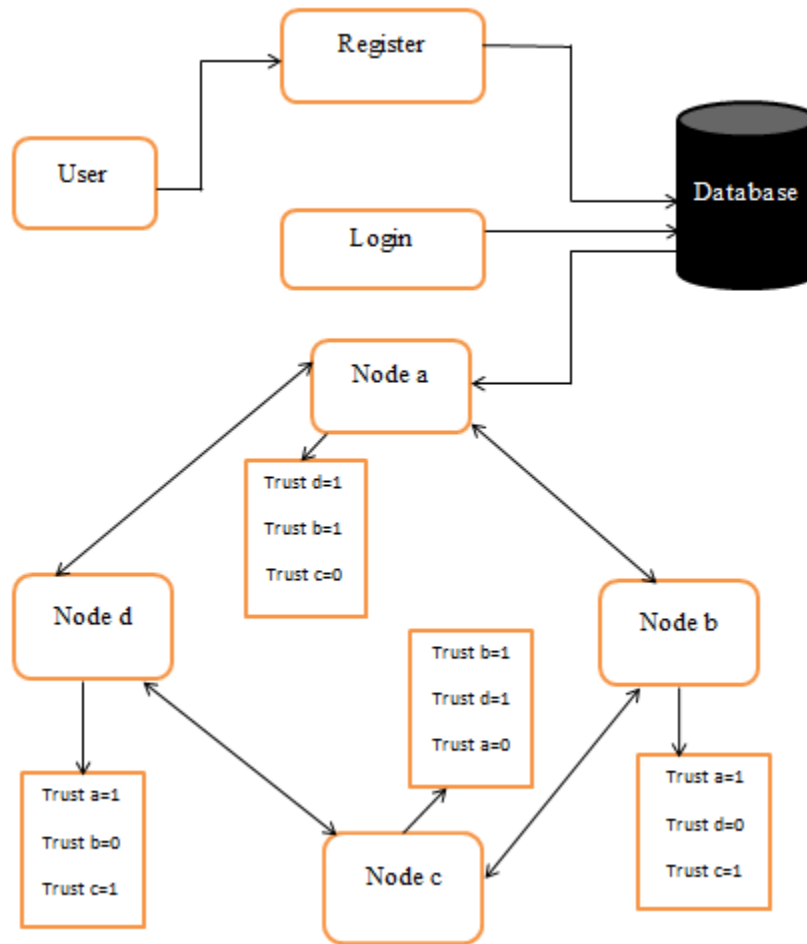
for all the node i **do**

choose k_i random nodes in its neighborhood

send gossip pair $((1 \div k_i + 1)y_{ij}, (1 \div k_i + 1)g_{ij})$ to all k_i nodes and also to itself

end for

3.6 SYSTEM ARCHITECTURE:



In the above diagram tells about the flow of objects between the classes. The main object of this diagram after user login interface. If the node A needs resource from node B, it means it will send the request to node B. The node B checks the reputation table before it interacts with node A, meaning it will check reputation; otherwise, it will push the differential gossip and find the average value of the node B. Finally, it will send the data.

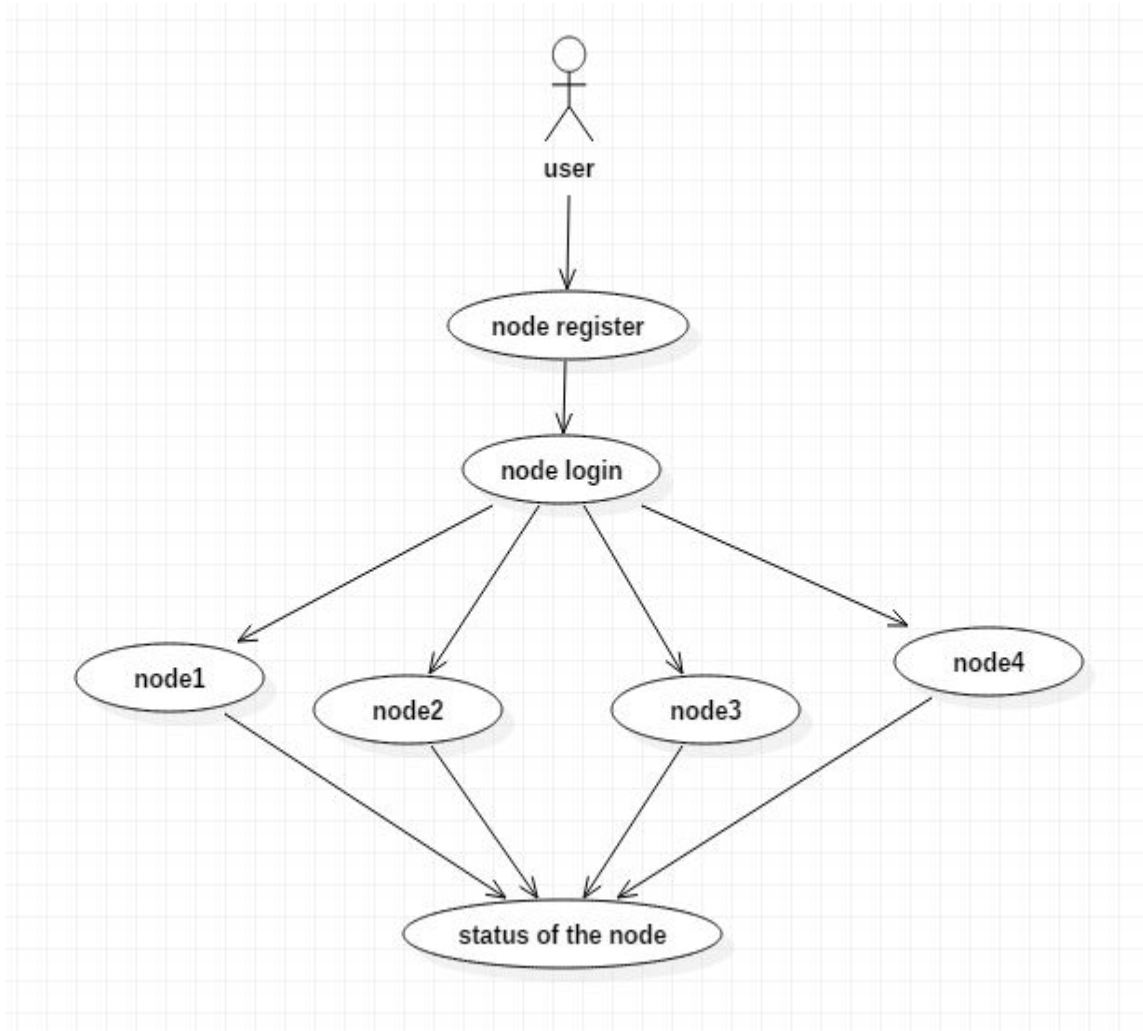
CHAPTER 4

MODELLING

4.1 GENERAL

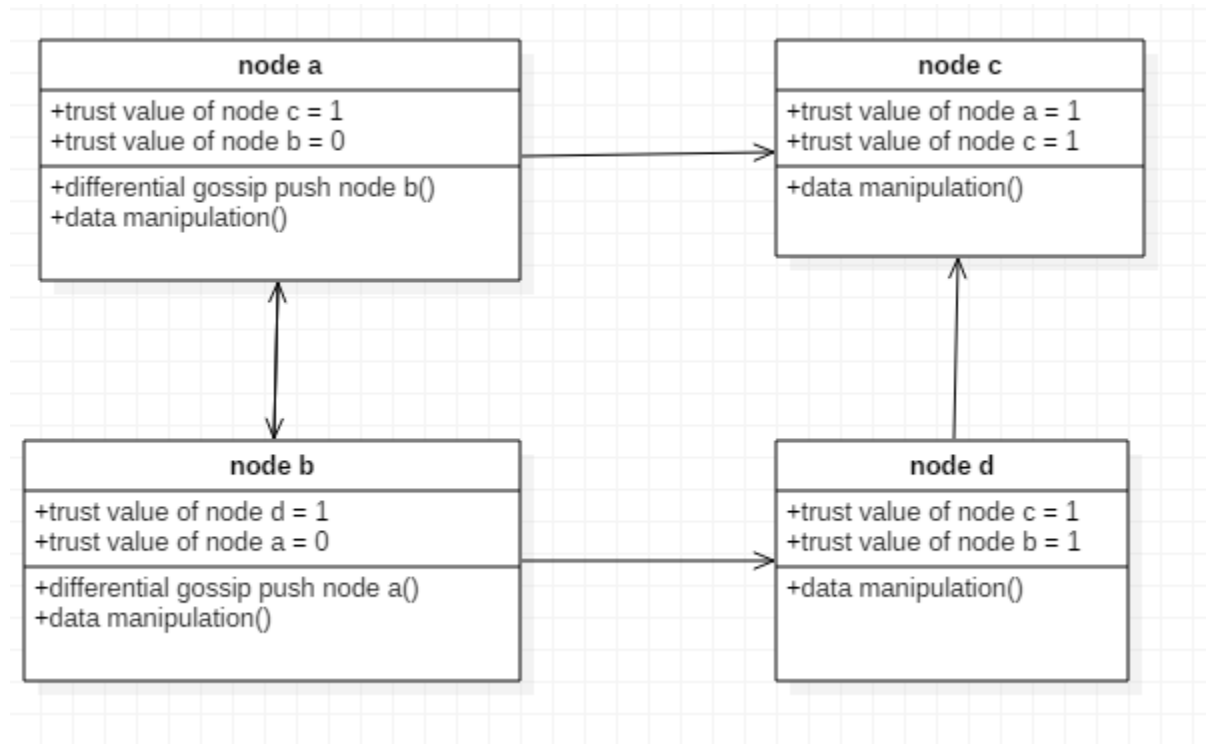
Design Engineering deals with the various UML [Unified Modeling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

4.1.1 USE CASE DIAGRAM:



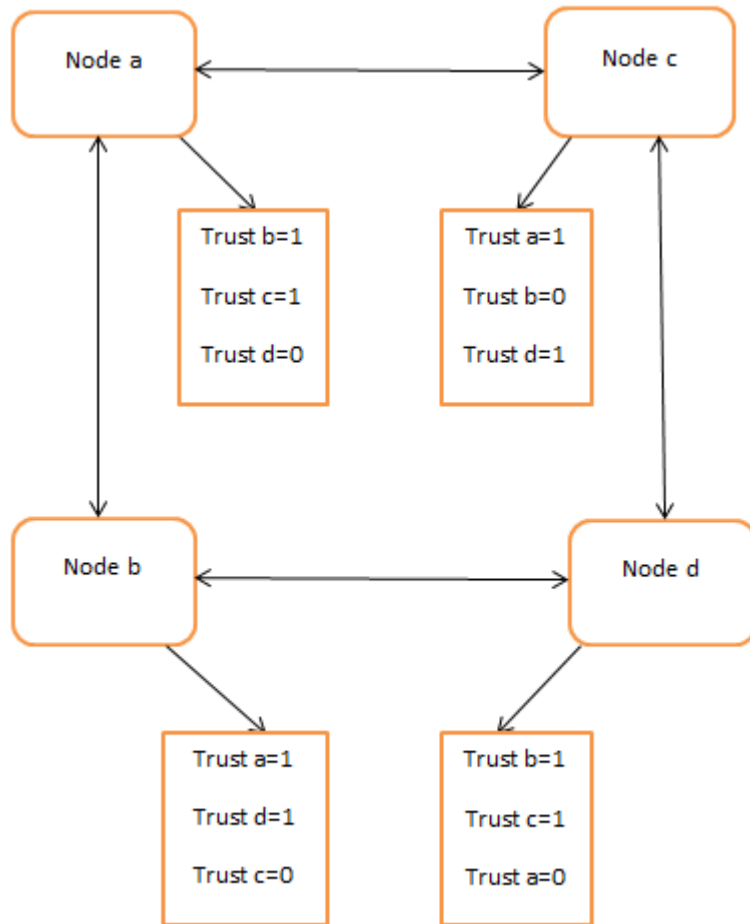
This is a type of behavioral diagram defined by and created from a Use-case analysis this diagram shows the user need to register and login to the peer to peer network. If the node A need resource from node B means if will send the request to node B. The node B check the reputation table if before it interact with node A means it will check reputation otherwise it will push the differential gossip and find the average value of the node B. finally it will send the data.

4.1.2 CLASS DIAGRAM:



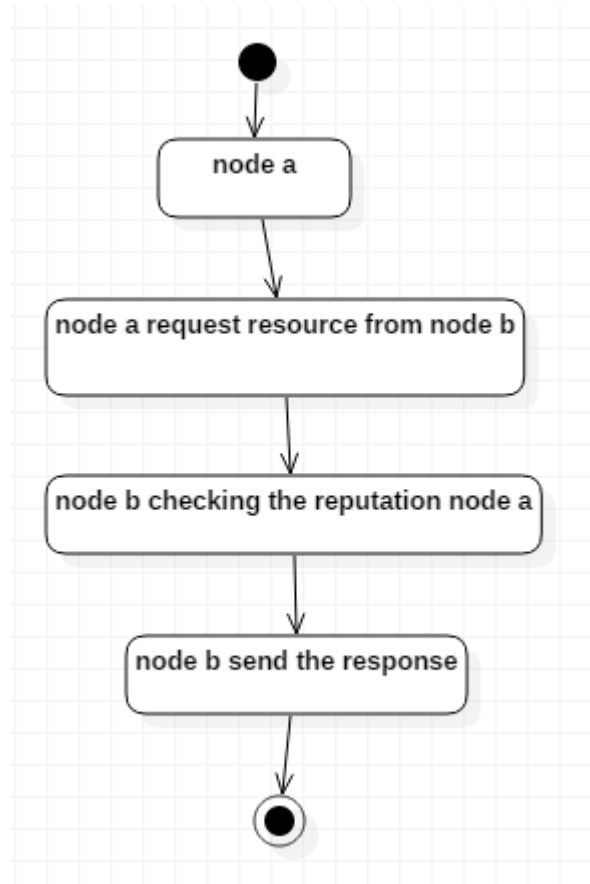
This type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. If the node A need resource from node B means if will send the request to node B. The node B check the reputation table if before it interact with node A means it will check reputation otherwise it will push the differential gossip and find the average value of the node B. finally it will send the data.

4.1.3 OBJECT DIAGRAM:



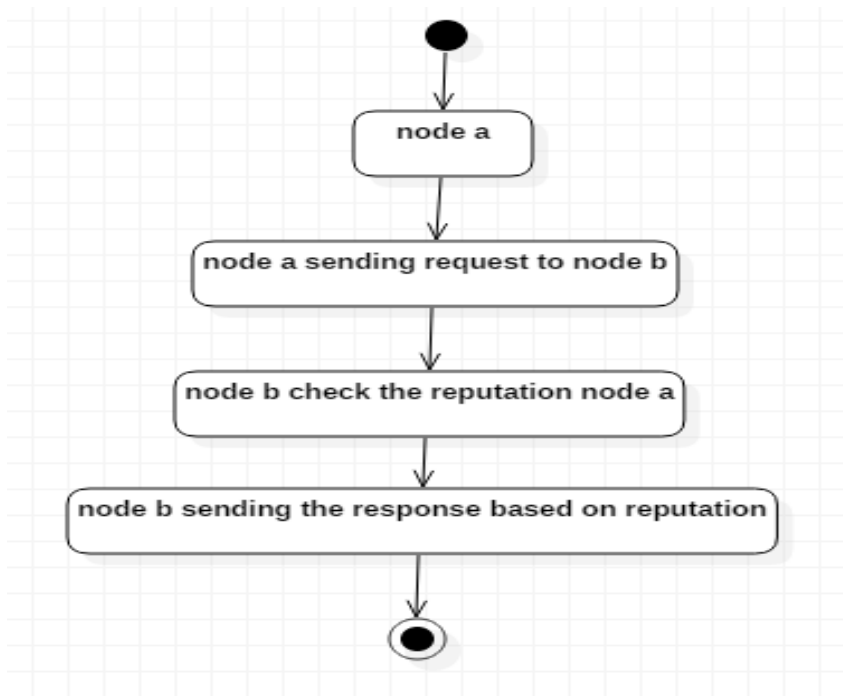
Object diagram we are telling about the flow of objects how the process is running. In the above diagram tells about the flow of objects between the classes. The main object of this diagram is if the node A needs resource from node B, it means it will send the request to node B. Node B checks the reputation table before it interacts with node A, meaning it will check reputation; otherwise, it will push the differential gossip and find the average value of the node B. Finally, it will send the data.

4.1.4 STATE DIAGRAM:



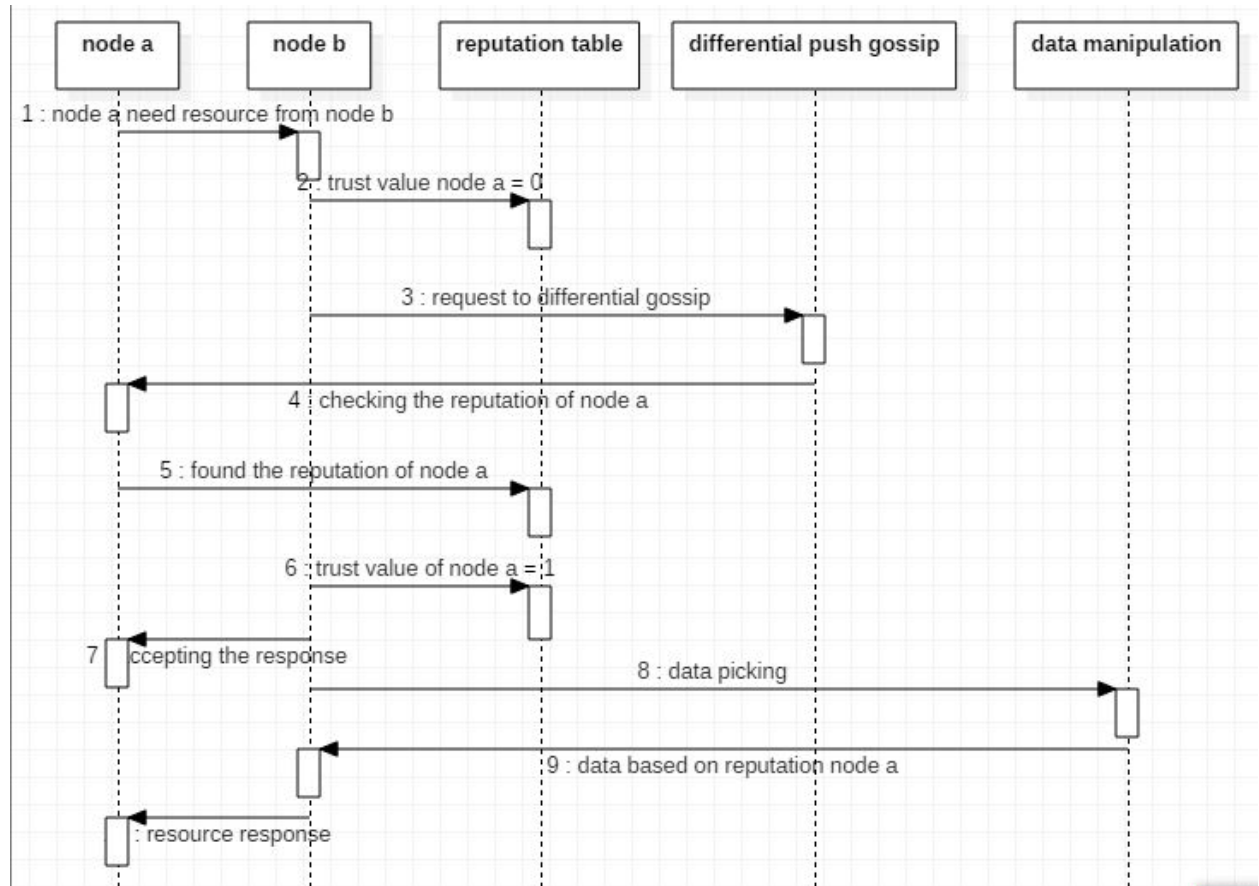
State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics. In our state diagram If the node A need resource from node B means if will send the request to node B. The node B check the reputation table if before it interact with node A means it will check reputation otherwise it will push the differential gossip and find the average value of the node B. finally it will send the data.

4.1.5 ACTIVITY DIAGRAM:



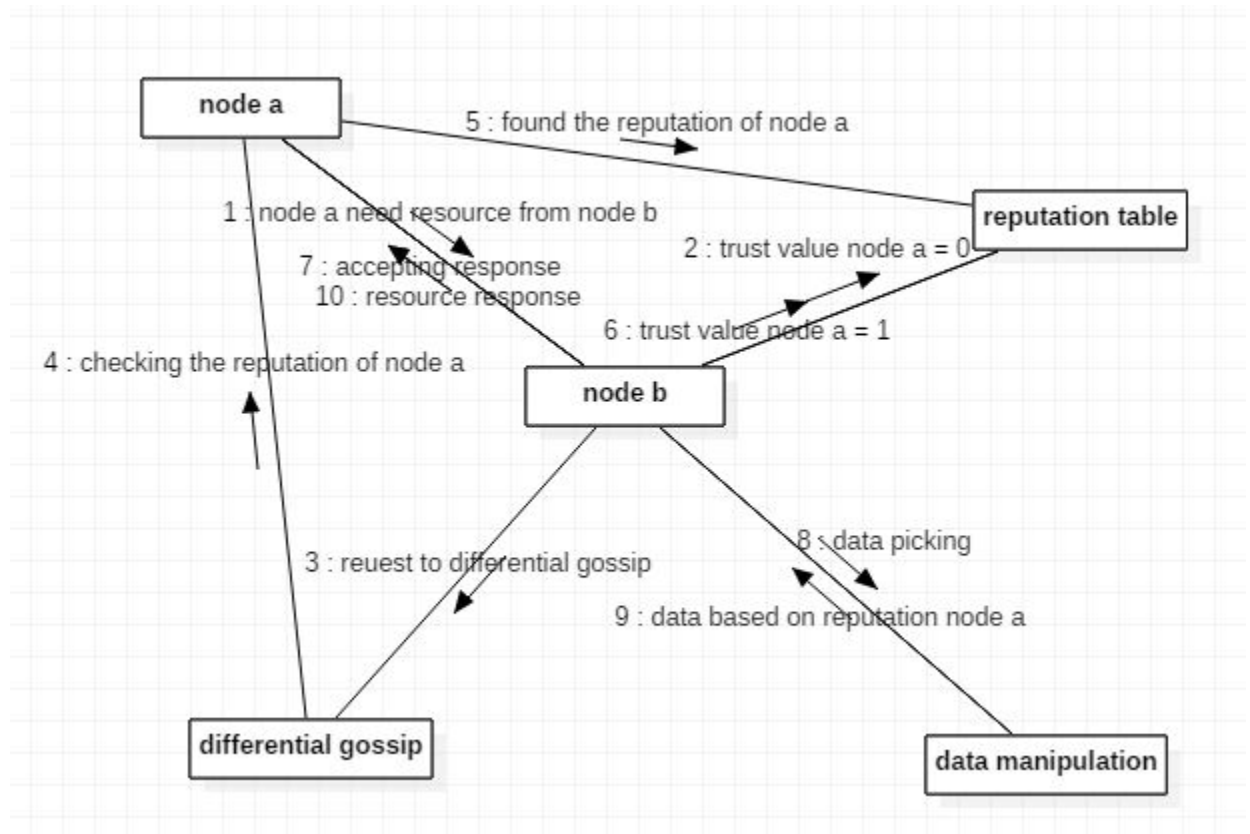
In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system diagram. After user login interface, if the node A needs resource from node B, it will send the request to node B. Node B checks the reputation table before it interacts with node A, meaning it will check reputation; otherwise, it will push the differential gossip and find the average value of the node B. Finally, it will send the data.

4.1.6 SEQUENCE DIAGRAM:



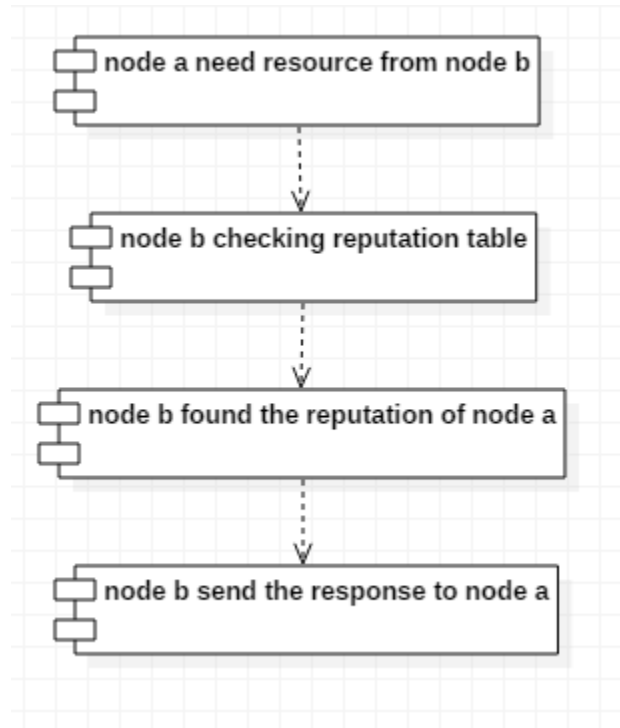
In our sequence diagram specifying processes operate with one another and in order user login file is after user login interface node A need resource from node B means if will send the request to node B. The node B check the reputation table if before it interact with node A means it will check reputation otherwise it will push the differential gossip and find the average value of the node B. finally it will send the data.

4.1.7 COLLABORATION DIAGRAM:



A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system. At the same time After user login interface node A need resource from node B means if will send the request to node B. The node B check the reputation table if before it interact with node A means it will check reputation otherwise it will push the differential gossip and find the average value of the node B. finally it will send the data.

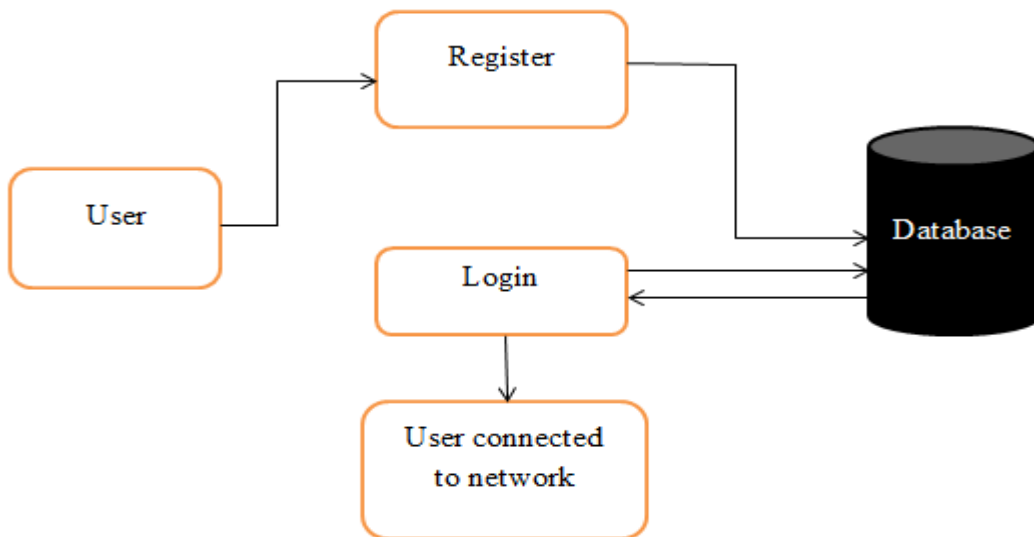
4.1.8 COMPONENT DIAGRAM:



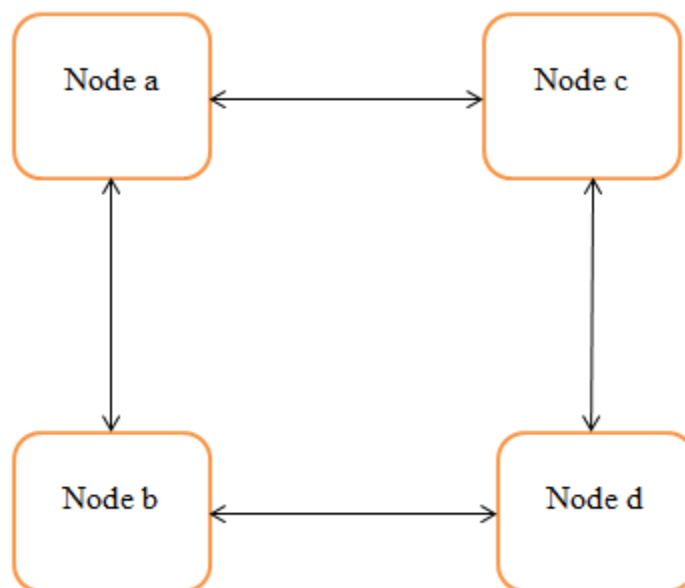
In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and they are used to illustrate the structure of arbitrarily complex systems. If the node A needs resource from node B, it means it will send the request to node B. The node B checks the reputation table before it interacts with node A, meaning it will check the reputation; otherwise, it will push the differential gossip and find the average value of the node B. Finally, it will send the data.

4.1.9 DATA FLOW DIAGRAM:

Level-0:

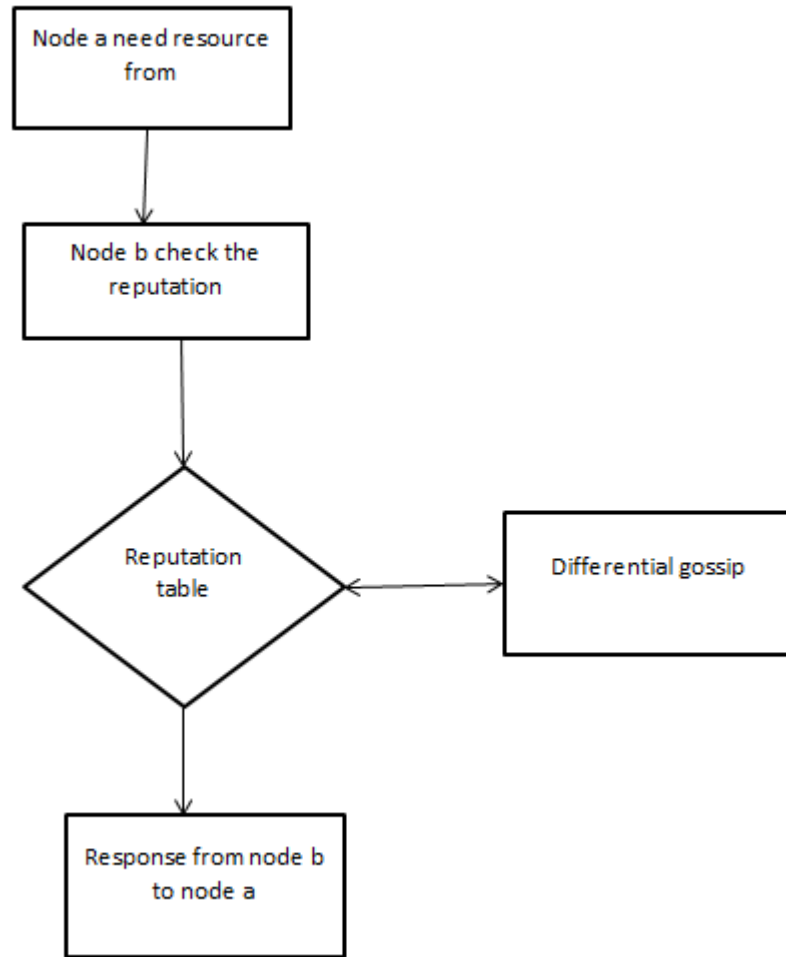


Level-1:



It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel. In the DFDs the level zero process is based on the login validations. After user login interface if the node A need resource from node B means it will send the request to node B. The node B check the reputation table if before it interact with node A means it will check reputation otherwise it will push the differential gossip and find the average value of the node B. finally it will send the data.

4.1.10 E-R DIAGRAM:



Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database. In our ER diagram we If the node A need resource from node B means if will send the request to node B. The node B check the reputation table if before it interact with node A means it will check reputation otherwise it will push the differential gossip and find the average value of the node B. finally it will send the data.

CHAPTER 5

IMPLEMENTATION

5.1 GENERAL

In this we implement the coding part using eclipse. Below are the coding's that are used to generate the domain module for Cloud Computing. Here the proposed techniques are used in the coding part to Cloud to Cloud Interaction.

5.2 SAMPLE CODE

File

```
package bean;

public class beanclass {

    private String Username, Password, Cpassword, UserId, Gender, DOB, Phone, Address;

    public void setUsername(String username) {

        Username = username;

    }

    public String getPassword() {

        return Password;

    }

    public void setPassword(String password) {

        Password = password;

    }

    public String getCpassword() {

        return Cpassword;

    }

    public void setCpassword(String cpassword) {
```

```

Cpassword = cpassword;
}

public String getUserId() {
return UserId;
}

public void setUserId(String userId) {
UserId = userId;
}

public String getGender() {
return Gender;
}

public void setGender(String gender) {
Gender = gender;
}

public String getDOB() {
return DOB;
}

public void setDOB(String dOB) {
DOB = dOB;
}

public String getPhone() {
return Phone;
}

public void setPhone(String phone) {
Phone = phone;
}

public String getAddress() {

```

```

return Address;

}

public void setAddress(String address) {

Address = address;

}

}

```

Peer-to-Peer

```

package com.PEER;

import java.awt.Component;
import java.awt.EventQueue;
import javax.print.attribute.standard.MediaSize.Other;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.ItemEvent;

public class PEERA extends JFrame implements Runnable {
    ArrayList<String> listj= new ArrayList<String>();

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {

```

```

PEERA frame = new PEERA();
frame.setVisible(true);
Thread t=new Thread(frame);
t.start();
} catch (Exception e) {
e.printStackTrace();
}
public PEERA() {
setTitle("Peer_A");
setBounds(100, 100, 528, 470);
contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
JLabel label = new JLabel("");
label.setBounds(10, 35, 228, 179);
Border border = BorderFactory.createLineBorder(Color.red);
label.setBorder(border);
ImageIcon background = new ImageIcon("images/computer.jpg");
label.setIcon(background);
FileNameExtensionFilter filter = new FileNameExtensionFilter("TEXT FILES", "txt",
"text");
fileChooser.setFileFilter(filter);
int retval=fileChooser.showDialog(contentPane, "Add");
if (retval == JFileChooser.APPROVE_OPTION){
files=fileChooser.getSelectedFiles();
textField_1.setText(String.valueOf(files.length)+" Files Selected");
}
button.setBounds(12, 23, 81, 23);
panel.add(button);
textField_1 = new JTextField();
textField_1.setColumns(10);
textField_1.setBounds(92, 23, 131, 23);

```

```

panel.add(textField_1);
JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    System.out.println("File="+f);
    filename = f.getName();
    System.out.println("selected files="+filename1);
    System.out.println("FILENAME"+filename);
    path=f.getPath();
    byte b[]=null;
    try
    {
        FileInputStream fis=new FileInputStream(path);
        b = new byte[fis.available()];
        System.out.println("sdgfb==" +b);
        fis.read(b);
        String reading=new String(b);
        filecontent=filecontent+reading;
        System.out.println("Filecontent &*&*&*&"+filecontent);
        save("");
    }
    catch (Exception e1)
    {
        btnSave.setBounds(22, 59, 71, 23);
        jTextField_2 = new JTextField();
        textField_2.setBounds(12, 93, 81, 20);
        JOptionPane.showMessageDialog(rootPane, "Select View all Files");
        Connection con=DBcon.getConnection();
        try {
            ps=con.prepareStatement("SELECT * FROM `peertopeer`.`peera`");
            ResultSet i=ps.executeQuery();
            while(i.next()){

```

```

Path=i.getString(1).toString();
listj.add(i.getString(1));
System.out.println("i am list==" + list);
System.out.println("path ===" + Pathname);
list_1.add(Path);
add.put(Path, Content);
}
System.out.println("Hash Map"+ add);
System.out.println("content==" +Content);
} catch (Exception e1) {
e1.printStackTrace();
}
JOptionPane.showMessageDialog(rootPane, "Check Any other Content File");
}
try {
ps=con.prepareStatement("SELECT CONTENT FROM `peertopeer`.`peera` where
while(i.next()){
Content=i.getString(1).toString();
System.out.println("sdfsdf==" +Content);
}
} catch (Exception e1) {
e1.printStackTrace();
}

public void actionPerformed(ActionEvent e) {
String x=JOptionPane.showInputDialog("X axis");
System.out.println("Alagu sundar"+x);
String y=JOptionPane.showInputDialog("Y axis");
System.out.println("Alagu sundar"+y);
String w=JOptionPane.showInputDialog("Width ");
System.out.println("Alagu sundar"+w);
String h=JOptionPane.showInputDialog("Height");

```

```

System.out.println("Alagu sundar"+h);
ImageIcon icon = new ImageIcon("images/request1.jpg");
JOptionPane.showMessageDialog(null,"Trust Value Change For 1 Successfully","Hello",
JOptionPane.INFORMATION_MESSAGE,icon);
ArrayList al= new ArrayList<>();
al.add("PEERA");
al.add(x);
al.add(y);
al.add(w);
al.add(h);
System.out.println(al);
try {
Socket socket=new Socket("localhost", port);
System.out.println("connect");
ObjectOutputStream outputStream=new ObjectOutputStream(socket.getOutputStream());
outputStream.writeObject(al);
while (itr.hasNext()) {
String trust=itr.next().toString();
text.setText(trust);
}
} catch ( IOException e2) {
e2.printStackTrace();
} catch (ClassNotFoundException e1) {
e1.printStackTrace();
}
package com.PEER;

import java.awt.Component;

import java.awt.EventQueue;
import javax.print.attribute.standard.MediaSize.Other;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;

```

```

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.ItemEvent;

public class PEERA extends JFrame implements Runnable {
    ArrayList<String> listj= new ArrayList<String>();
    HashMap<String , String> add=new HashMap<String,String>();
    ArrayList files1=new ArrayList<>();
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            try {
                PEERA frame = new PEERB();
                frame.setVisible(true);
                Thread t=new Thread(frame);
                t.start();
            } catch (Exception e) {
                e.printStackTrace();
            }
        })
    }
    public PEERB() {
        setTitle("Peer_B");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 528, 470);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        JLabel label = new JLabel("");
    }
}

```



```

label.setBounds(10, 35, 228, 179);
Border border = BorderFactory.createLineBorder(Color.red);
label.setBorder(border);
ImageIcon background = new ImageIcon("images/computer.jpg");
FileNameExtensionFilter filter = new FileNameExtensionFilter("TEXT FILES", "txt",
"text");
fileChooser.setFileFilter(filter);
int retval=fileChooser.showDialog(contentPane, "Add");
if (retval == JFileChooser.APPROVE_OPTION){
files=fileChooser.getSelectedFiles();
textField_1.setText(String.valueOf(files.length)+" Files Selected");
}
button.setBounds(12, 23, 81, 23);
panel.add(button);
textField_1 = new JTextField();
textField_1.setColumns(10);
textField_1.setBounds(92, 23, 131, 23);
panel.add(textField_1);
JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
ImageIcon icon = new ImageIcon("images/ezgif.com-resize.gif");
JOptionPane.showMessageDialog(null,"Loading", "Hello",
JOptionPane.INFORMATION_MESSAGE,icon);
System.out.println("selected files="+filename1);
System.out.println("FILENAME"+filename);
path=f.getPath();
//text.setText(path);
byte b[]=null;
try
{

```

```

FileInputStream fis=new FileInputStream(path);
b = new byte[fis.available()];
System.out.println("sdgfb==" +b);
fis.read(b);
String reading=new String(b);
filecontent=filecontent+reading;
System.out.println("Filecontent &*&*&*&"+filecontent);
save("");
}
catch (Exception e1)
{
e1.printStackTrace();
}
JOptionPane.showMessageDialog(rootPane, "Saved Successfully");
}
btnSave.setBounds(22, 59, 71, 23);
panel.add(btnSave);
textField_2 = new JTextField();
textField_2.setBounds(12, 93, 81, 20);
panel.add(textField_2);
textField_2.setColumns(10);
public void actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(rootPane, "Select View all Files");
Connection con=DBcon.getsqConnection();
try {
ps=con.prepareStatement("SELECT * FROM `peertopeer`.`peerb`");
ResultSet i=ps.executeQuery();
while(i.next()){
Path=i.getString(1).toString();
listj.add(i.getString(1));
System.out.println("i am list==" + list);

```

```

Content=i.getString(2);
Pathname=i.getString(3);
System.out.println("path ===" + Pathname);
list_1.add(Path);
add.put(Path, Content);
} catch (Exception e1) {
e1.printStackTrace();
}
JOptionPane.showMessageDialog(rootPane, "Check Any other Content File");
}
try {
ps=con.prepareStatement("SELECT CONTENT FROM `peertopeer`.`peerb` where
PATH='"+selectedPacket+"'");
System.out.println("sdfsdf==" + Content);
} catch (Exception e1) {
e1.printStackTrace();
}
public void actionPerformed(ActionEvent e) {
String x=JOptionPane.showInputDialog("X axis");
System.out.println("Alagu sundar"+x);
String y=JOptionPane.showInputDialog("Y axis");
System.out.println("Alagu sundar"+y);
String w=JOptionPane.showInputDialog("Width ");
System.out.println("Alagu sundar"+w);
String h=JOptionPane.showInputDialog("Height");
System.out.println("Alagu sundar"+h);
ImageIcon icon = new ImageIcon("images/request1.jpg");
JOptionPane.showMessageDialog(null,"Trust Value Change For 1 Successfully","Hello",
JOptionPane.INFORMATION_MESSAGE,icon);
al.add("PEERB");
al.add(x);

```

```

al.add(y);
al.add(w);
al.add(h);
try {
Socket socket=new Socket("localhost", port);
System.out.println("connect");
ObjectOutputStream outputStream=new ObjectOutputStream(socket.getOutputStream());
files1=(ArrayList) inputStream.readObject();
System.out.println(files1);
Iterator itr=files1.iterator();
} catch ( IOException e2) {
e2.printStackTrace();
} catch (ClassNotFoundException e1) {
e1.printStackTrace();
}
}

```

Differential Gossip Algorithm

```

package com.PEER;
import javax.swing.border.EmptyBorder;
import javax.swing.BorderFactory;
import javax.swing.DefaultComboBoxModel;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.ListSelectionModel;
import javax.swing.border.EtchedBorder;

```

```

import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.awt.List;
import java.awt.Font;
import javax.swing.JTextField;
public class globalresource extends JFrame {
    private JPanel contentPane;
    ArrayList files=new ArrayList<>();
    List list_1 = new List();
    String selectedPacket;
    private JTextField selecttext;
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    globalresource frame = new globalresource();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
    public globalresource() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 538, 416);
    }
}

```

```

contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane);
JLabel label = new JLabel("");
label.setBounds(10, 21, 241, 182);
Border border = BorderFactory.createLineBorder(Color.red);
label.setBorder(border);
ImageIcon background = new ImageIcon("images/global.jpg");
contentPane.setLayout(null);
label.setIcon(background);
contentPane.add(label);
JLabel lblSeletAPeer = new JLabel("Selet a Peer To Response");
lblSeletAPeer.setBounds(317, 21, 151, 14);
contentPane.add(lblSeletAPeer);
final JComboBox comboBox_1 = new JComboBox();
comboBox_1.setBounds(317, 48, 151, 20);
comboBox_1.setModel(new DefaultComboBoxModel(new String[] { "Select", "PeerA",
"PeerB", "PeerC", "PeerD", "PeerE" }));
contentPane.add(comboBox_1);
JScrollPane scrollPane = new JScrollPane();
scrollPane.setBounds(317, 81, 151, 98);
contentPane.add(scrollPane);
list_1.addItemListener(new ItemListener() {
public void itemStateChanged(ItemEvent e) {
selectedPacket = list_1.getSelectedItem();
selectcon=add.get(selectedPacket);
selecttext.setText(selectedPacket);
}
});
scrollPane.setViewportViewView(list_1);
JPanel panel = new JPanel();

```

```

panel.setBounds(74, 313, 414, 35);
panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));
contentPane.add(panel);
JButton button = new JButton("Show");
button.setBounds(126, 7, 77, 25);
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try{
            String sendTo=comboBox_1.getSelectedItem().toString().trim();
            System.out.println("infomeen is==="+sendTo);
            switch(sendTo){
                case "PeerA": port=100;break;
                case "PeerB": port=101;break;
                case "PeerC": port=102;break;
                case "PeerD": port=103;break;
                case "PeerE": port=104;break;
                default:break;
            }
            ArrayList list=new ArrayList<>();
            list.add("show");
            //list.add(infoAbout);
            Socket socket=new Socket("localhost", port);
            System.out.println("connect");
            ObjectOutputStream outputStream=new ObjectOutputStream(socket.getOutputStream());
            outputStream.writeObject(list);
            ObjectInputStream inputStream=new ObjectInputStream(socket.getInputStream());
            add=(HashMap<String, String>) inputStream.readObject();
            for(Map.Entry m:add.entrySet())
            {
                s= (String) m.getKey();
                System.out.println("fgdfg==" +s);
            }
        }
    }
});

```

```

list_1.add(s);
}
if(add!=null){
}
else
JOptionPane.showMessageDialog(rootPane, "The did not have any resources");
}catch(Exception ex){
ex.printStackTrace();
}
};
panel.setLayout(null);
panel.add(button);
JButton button_1 = new JButton("Download");
button_1.setBounds(229, 7, 111, 25);
button_1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
try{
JFileChooser chooser=new JFileChooser();
chooser.setDialogTitle("Download");
chooser.setAcceptAllFileFilterUsed(false);
FileNameExtensionFilter filter = new FileNameExtensionFilter("TEXT FILES", "txt",
"text");
chooser.setFileFilter(filter);
int retval=chooser.showSaveDialog(contentPane);
FileOutputStream fos=new FileOutputStream(chooser.getSelectedFile());
fos.write(selectcon.getBytes());
}
catch(Exception e1){
e1.printStackTrace();
}
}
}

```


5.3 DATABASE TABLES

Register Table

Field Name	Data Type	Size	Allow Nulls
User name	Varchar	45	No
Password	Varchar	45	No
Mobile no.	Varchar	45	Yes
Email id	Varchar	45	Yes
Address	Varchar	45	Yes

Source Table

Field Name	Data Type	Size	Allow Nulls
To	Varchar	45	No
Message	Varchar	45	No
User name	Varchar	45	Yes
Upload location	Varchar	45	Yes

5.4 TEST CASES

Peer Case ID	Peer Aggregation	Peer Steps	Peer Data	Expected Result	Actual Result	Pass or Fail
Peer A	Aggregated to Peer D	Enter ID, Enter Aggregation	Check Aggregation	Peer gets successfully aggregated to Peer D	Peer gets successfully aggregated to Peer D	Pass
Peer B	Aggregated to Peer A	Enter ID, Enter Aggregation	Check Aggregation	Peer gets unsuccessfully aggregated to Peer A	Peer gets unsuccessfully aggregated to Peer D	Pass
Peer C	Aggregated to Peer D	Enter ID, Enter Aggregation	Check Aggregation	Peer gets unsuccessfully aggregated to Peer D	Peer gets successfully aggregated to Peer D	Fail
Peer D	Aggregated to Peer B	Enter ID, Enter Aggregation	Check Aggregation	Peer gets unsuccessfully aggregated to Peer B	Peer gets unsuccessfully aggregated to Peer B	Pass

5.5 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

5.6 TYPES OF TESTS

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an

individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Performance Testing

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization:

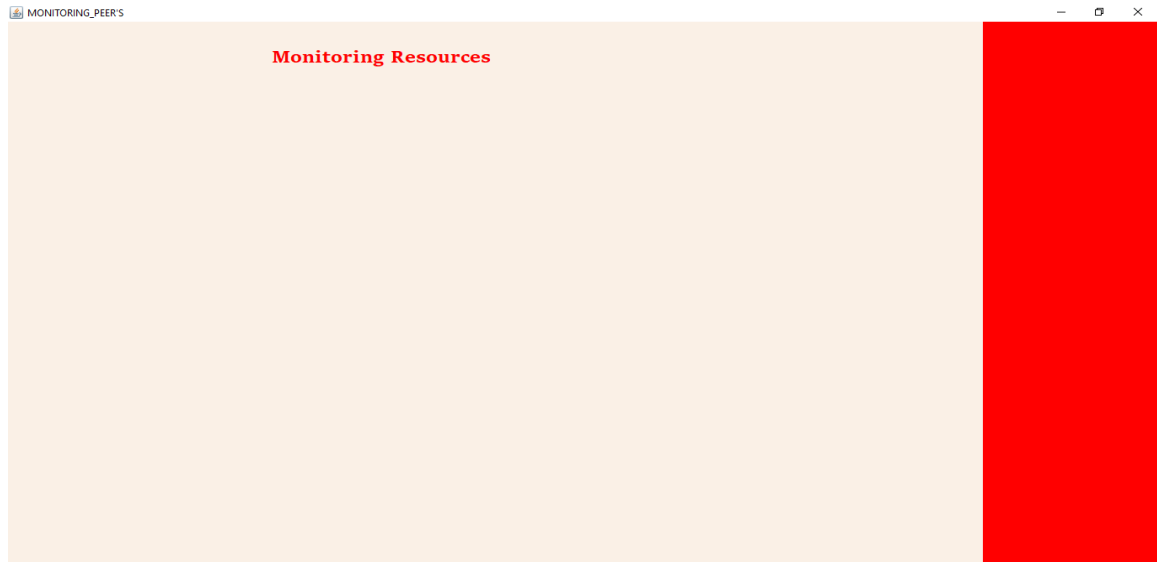
- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

Build the test plan

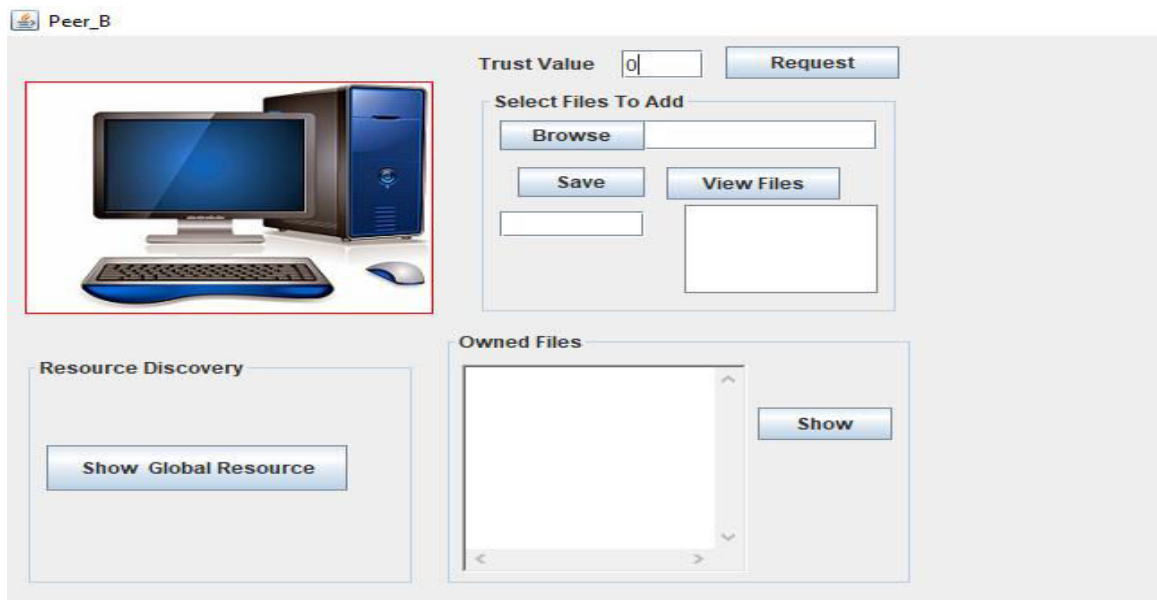
Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identify the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

CHAPTER 6

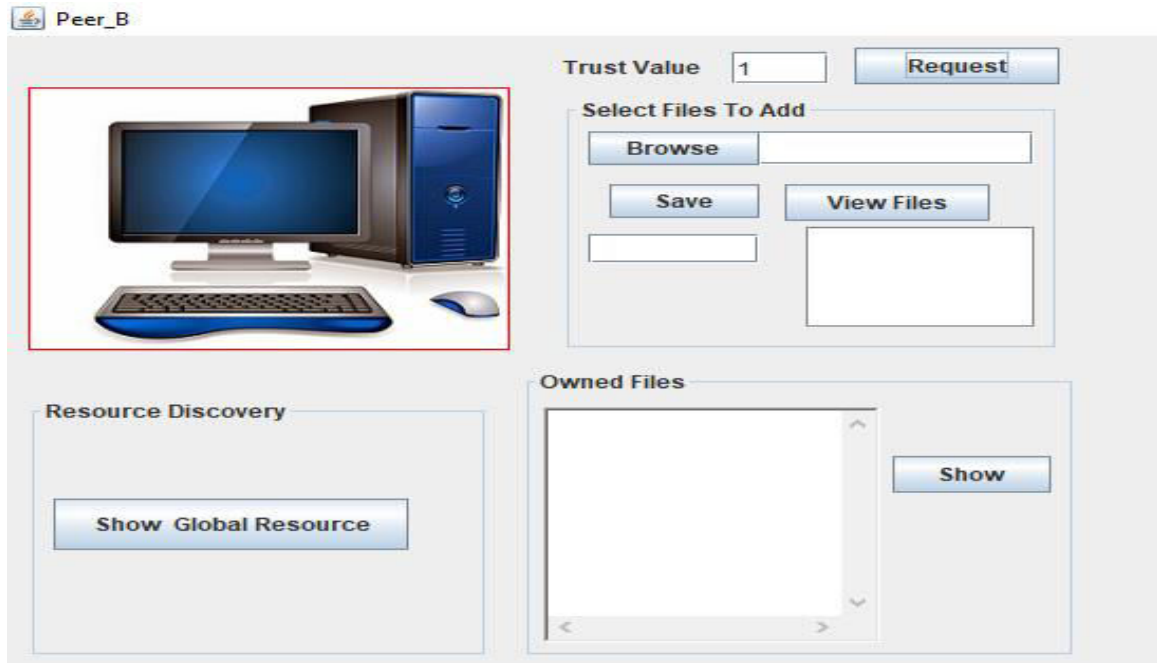
RESULT ANALYSIS



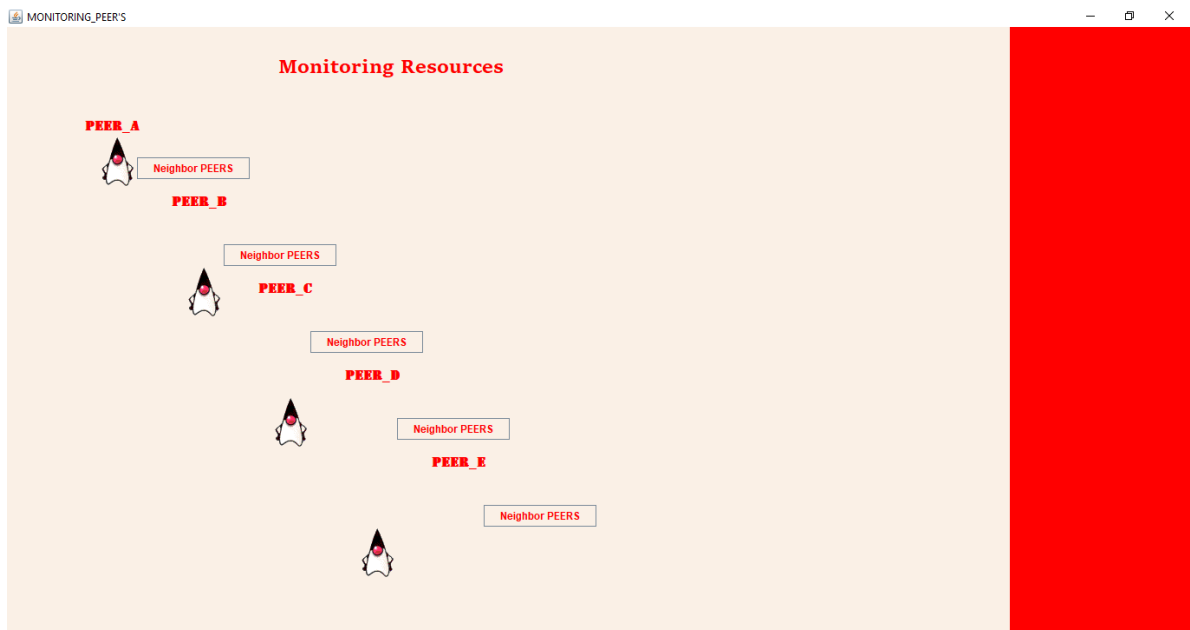
Monitoring resources give information about how many peers are available to communicate. The above figure shows that there are no peers are available at present.



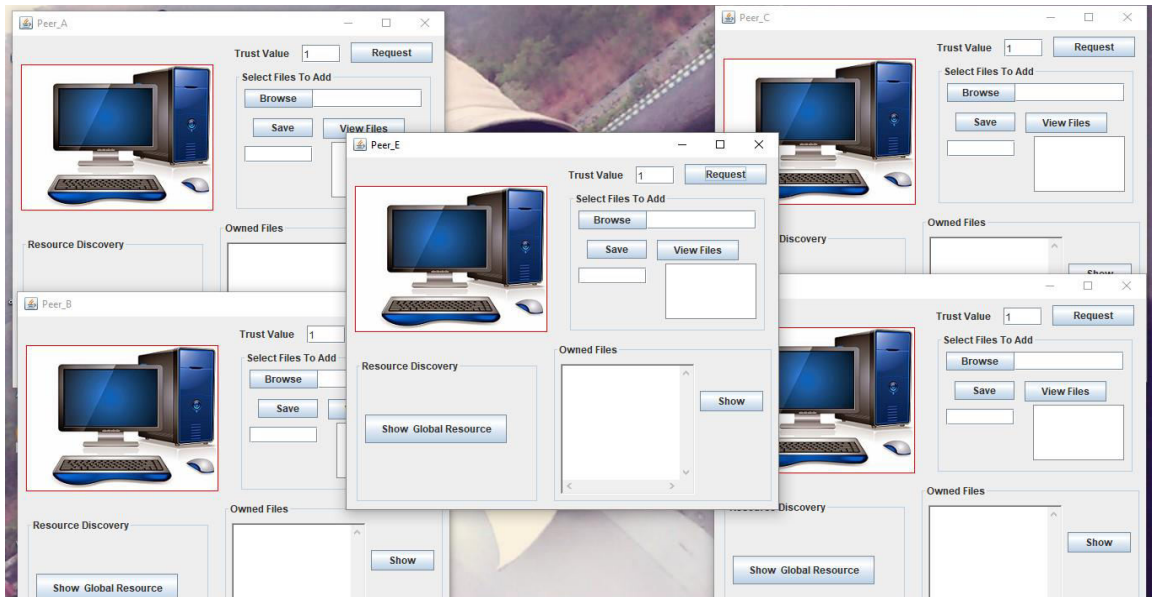
This shows that a peer (eg. PeerB) is selected but not yet confirmed in the monitoring resources as its trust value is zero.



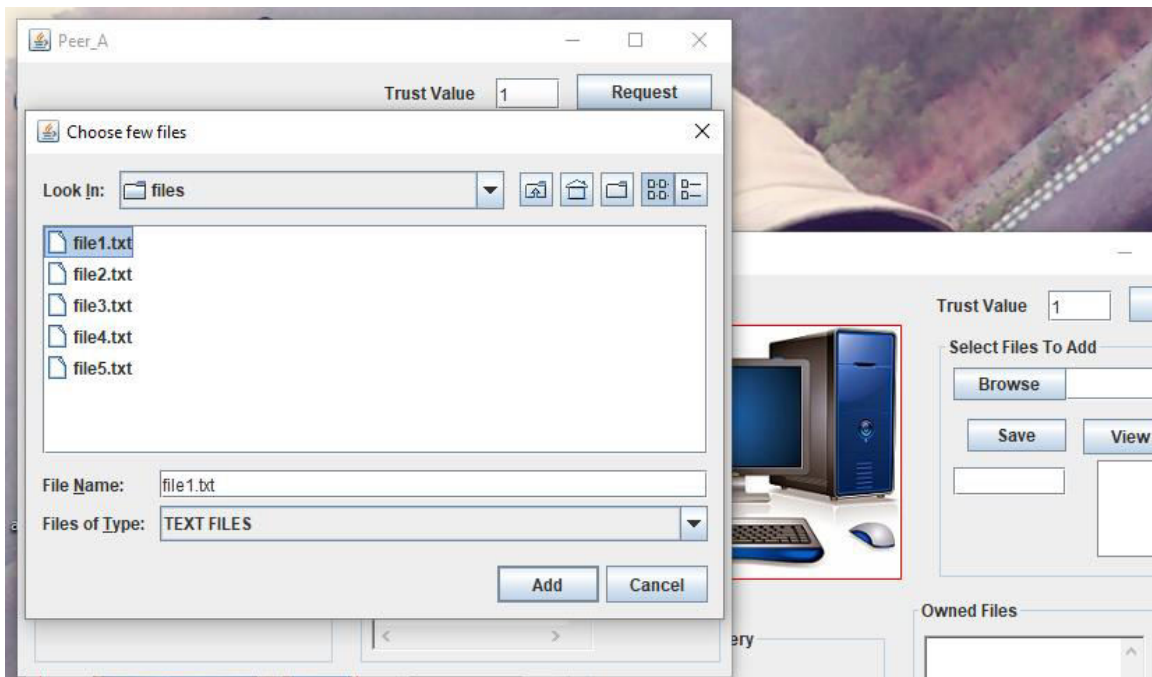
This shows that trust value of peer is changed to one and its will be available in the monitoring resources. In the same way required peers can be placed in the monitoring resources by converting their trust value to one through the request.



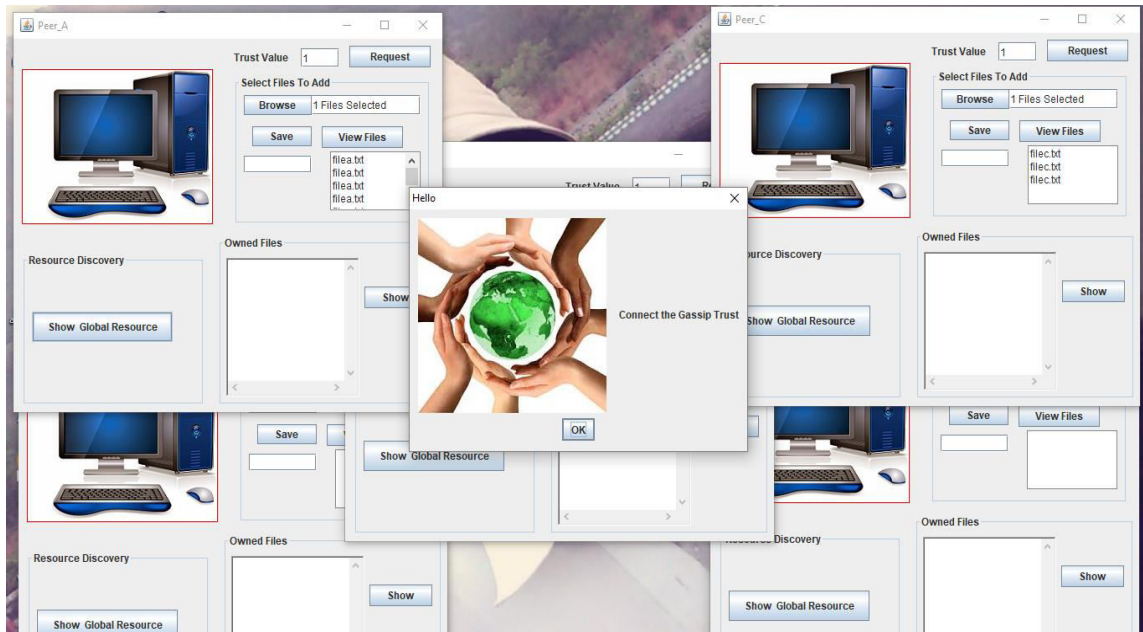
Here the available peers and their neighboring peers can be seen through the monitoring resources.



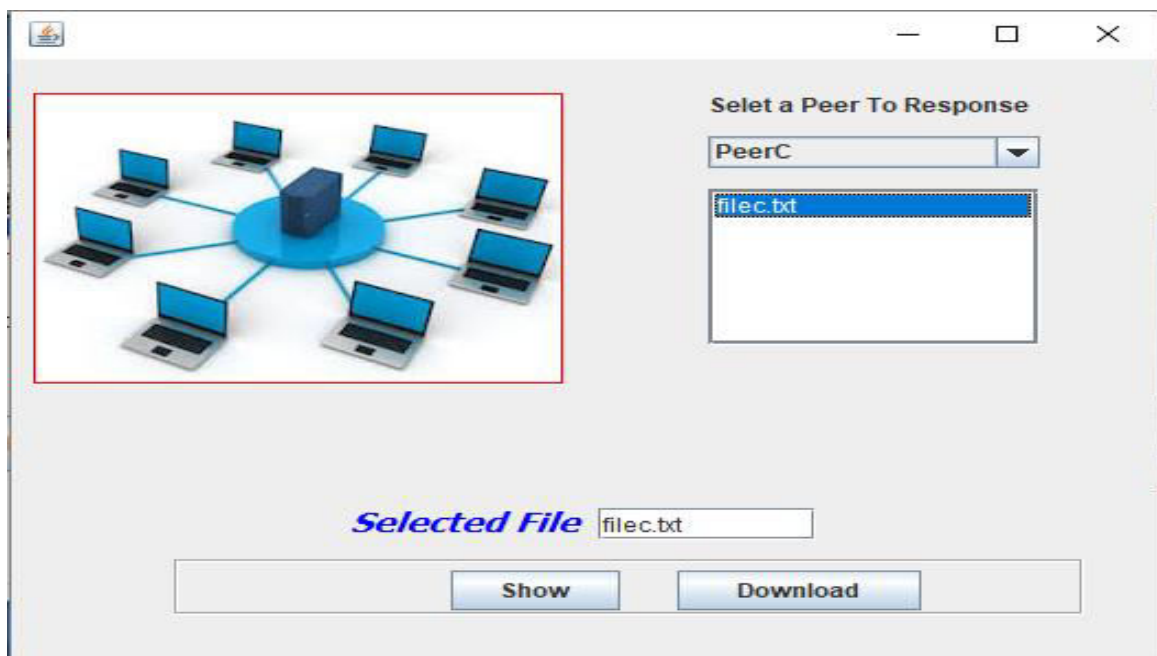
All the required peers are selected to generate communication with each other and all the peers are having trust value one.



Here the files available in the selected peer are seen and the required file is selected to share.



The above figure shows that the peer is trying to connect to other peer through the gossip trust.



Here the above figure gives information that peer is connected to gossip trust successfully. The required peer is selected and the file present in it are seen and chosen to download.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

This paper work includes backup storage on friend peers. The user will be able to set the redundancy ratio and split the backup in a way that allows to reconstruct the backup if a given fraction of the friends are online (e.g., backup can be reconstructed if 3 of 5 friends are online). Currently a JavaFX front-end is used, which will be replaced by a web-based GUI that is work in progress. For future work, a desktop integration that allows the user to use Box2Box transparently is foreseen. Besides encryption, future work investigates more security aspects and potential attack scenarios, such as access restriction or colluding peers. An interesting security aspect is to consider friend peers as trusted entities. Relying on these trusted peers can mitigate many attack scenarios.

7.2 FUTURE ENHANCEMENT

Future Concept:

- We have to point out that the Gossip Trust system is not restricted to apply only in uninstructed P2P systems.
- With minor modifications, this reputation system performs even better in a structured P2P systems.

Future Technique:

- Gossip Trust Algorithm

Technique Definition:

A peer providing corrupted services is highly likely to issue dishonest reputation scores. To probe further, we suggest to keep two kinds of reputation scores on each peer node: one to measure the *quality-of-service* (QoS) performance measures reported here and another for *quality-of-feedback* (QoF) by participating peers.

Extravagance:

The Gossip Trust system also leverages the ranking of all nodes in terms of their standing in the global reputation vector. The effects of peer collusion are reduced to the minimum due their low standing in the ranking order.

CHAPTER 8

REFERENCES

- [1] M.Feldman and J.Chuang, “Overcoming free-riding behavior in peer-to-peer systems,” SIGecom Exch., vol. 5,no. 4, pp. 41–50, Jul. 2005. [Online]. Available: <http://doi.acm.org/10.1145/1120717.1120723>
- [2] B.Yang, T.Condie, S.D.Kamvar, and H.Garcia-Molina, “Non-cooperation in competitive p2p networks.” in ICDCS.IEEE Computer Society, 2005, pp. 91–100. [Online]. Available: <http://dblp.uni-trier.de/db/conf/icdcs/icdcs2005.html> YangCKG05
- [3] Y.Tang, H.Wang, and W.Dou, “Trust based incentive in p2pnetwork,” in Proceedings of the E-Commerce Technology for Dynamic E-Business, IEEE International Conference, ser. CEC-EAST '04.Washington, DC, USA: IEEE Computer Society, 2004, pp. 302–305.[Online]. Available: <http://dx.doi.org/10.1109/CEC-EAST.2004.71>
- [4] E.Adar and B.A.Huberman, “Free riding on gnutella, ” First Monday, vol. 5, p. 2000, 2000
- [5] D.Hughes, G.Coulson, and J.Walkerdine, “Free riding on gnutella revisited: The bell tolls?” IEEE Distributed Systems Online, vol. 6, no. 6, pp. 1–, Jun. 2005. [Online]. Available: <http://dx.doi.org/10.1109/MDSO.2005.31>
- [6] Chao Xie, “Analysis of large-scale peer-to-peer network topology” IEEE Internet Computing, vol. 13, no. 2, pp. 92–98, April 2, 2008.
- [7] A.Lamnitchi, M.Ripeanu, and I.Foster, “Small-world file-sharing communities” Oxford University Press, 2009. [Online]. Available: <http://EconPapers.repec.org/RePEc:oxp:obooks:9780195322484>
- [8] Albert.L, Barab and Albert.R, “Emergence of scaling in random networks” IEEE Transactions on Parallel and Distributed Systems, vol. 20, no.1, pp. 83–96, 2011.

- [9] S.D.Kamvar, M.T.Schlosser, H.G.Molina, “The eigen trust algorithm for reputation management in peer-to-peer networks” Knowledge and Data Engineering, IEEE Transactions on, vol. 15, no. 4, pp. 840–854, May 2, 2004
- [10] S.Shanshan, K.Hwang and Runfang Zhou, “Trusted peer-to-peer transactions with fuzzy reputation aggregation” 2013. .[Online]. Available:
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.4223>