METHODIST

Estd:2008

COLLEGE OF ENGINEERING AND TECHNOLOGY

(Affiliated to Osmania University & Approved by AICTE, New Delhi)

LABORATORY MANUAL

DISTRIBUTED SYSTEMS LAB

BE VII Semester (CBCS): 2020-21

NAME: _____

ROLL NO:_____

BRANCH:_____ **SEM:**_____

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERNG

Empower youth- Architects of Future World

Estd:2008

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

VISION

To produce ethical, socially conscious and innovative professionals who would contribute to sustainable technological development of the society.

MISSION

To impart quality engineering education with latest technological developments and interdisciplinary skills to make students succeed in professional practice.

To encourage research culture among faculty and students by establishing state of art laboratories and exposing them to modern industrial and organizational practices.

To inculcate humane qualities like environmental consciousness, leadership, social values, professional ethics and engage in independent and lifelong learning for sustainable contribution to the society.

METHODIST

Estd:2008

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING

LABORATORY MANUAL

DISTRIBUTED SYSTEMS LAB

Prepared
By
Dr. Vuppu Padmakar,

Associate Professor.

Estd:2008

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION & MISSION

VISION

To become a leader in providing Computer Science & Engineering education with emphasis on knowledge and innovation.

MISSION

- To offer flexible programs of study with collaborations to suit industry needs.
- To provide quality education and training through novel pedagogical practices.
- To expedite high performance of excellence in teaching, research and innovations.
- To impart moral, ethical values and education with social responsibility.

Estd:2008

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES

After 3-5 years of graduation, the graduates will be able to

PEO1: Apply technical concepts, Analyze, Synthesize data to Design and create novel products and solutions for the real life problems.

PEO2: Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.

PEO3: Promote collaborative learning and spirit of team work through multidisciplinary projects

PEO4: Engage in life-long learning and develop entrepreneurial skills.

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES

Engineering graduates will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the Engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

At the end of 4 years, Computer Science and Engineering graduates at MCET will be able to:

PSO1: Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.

PSO2: Develop software applications with open-ended programming environments**.**

PSO3: Design and develop solutions by following standard software engineering principles and implement by using suitable programming languages and platforms

Course Code	Course Title						Core / Elective
PC 752 CS	**Distributed Systems Lab**						**Core**
Prerequisite	Contact Hours per Week				CIE	SEE	Credits
	L	T	D	P			
-	**-**	**-**	**-**	**2**	**25**	**50**	**1**

Course Objectives

- To implement client and server programs using sockets
- To learn about working of NFS
- To use Map, reduce model for distributed processing
- To develop mobile applications

Course Outcomes

After completing this course, the student will be able to

1. Write programs that communicate data between two hosts
2. Configure NFS
3. Use distributed data processing frameworks and mobile application tool kits

List of Experiments to be performed:

1. Implementation FTP Client
2. Implementation of Name Server
3. Implementation of Chat Server
4. Understanding of working of NFS (Includes exercises on Configuration of NFS)
5. Implementation of Bulletin Board.
6. Implement a word count application which counts the number of occurrences of each word a large collection of documents Using Map Reduce model.
7. Develop an application (small game-like scrabble, Tic-tac-Toe) using Android SDK.

METHODIST

Estd:2008

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Outcomes (CO's):

SUBJECT NAME: DISTRIBUTED SYSTEMS LAB **CODE: PC752CS**

SEMESTER: VII

CO No.	Course Outcome	Taxonomy Level
PC752CS.1	Write programs that communicate data between two hosts	Creating
PC752CS.2	Configure Network File Systems	Understanding
PC752CS.3	Use distributed data processing frameworks and mobile application tool kits	Applying
PC752CS.4	Trace Communication protocols in distributed systems	Analyze
PC752CS.5	Develop an application using a technology from distributed system	Creating
PC752CS.6	Design of algorithm distributed system	Creating

METHODIST

Estd:2008

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the program / experiment details.
3. Student should enter into the laboratory with:
    a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b. Laboratory Record updated up to the last session experiments.
    c. Formal dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviours with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out. If anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

METHODIST

Estd:2008

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CODE OF CONDUCT FOR THE LABORATORY

- All students must observe the dress code while in the laboratory
- Footwear is NOT allowed
- Foods, drinks and smoking are NOT allowed
- All bags must be left at the indicated place
- The lab timetable must be strictly followed
- Be PUNCTUAL for your laboratory session
- All programs must be completed within the given time
- Noise must be kept to a minimum
- Workspace must be kept clean and tidy at all time
- All students are liable for any damage to system due to their own negligence
- Students are strictly PROHIBITED from taking out any items from the laboratory
- Report immediately to the lab programmer if any damages to equipment

BEFORE LEAVING LAB:

- Arrange all the equipment and chairs properly.
- Turn off / shut down the systems before leaving.
- Please check the laboratory notice board regularly for updates.

Lab In – charge

Estd:2008

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LIST OF EXPERIMENTS

SI. No.	Name of the Experiment	Date of Experiment	Date of Submission	Page No	Faculty Signature
1.	Implementation FTP Client			3	
2	Implementation of Name Server			11	
3	Implementation of Chat Server			14	
4	Understanding of Working of NFS (includes exercises Configuration of NFS)			20	
5.	Implementation of Bulletin Board.			21	
6.	Implement a word count application which counts the number of occurrences of each word a large collection of documents Using Map Reduce model.			32	
7.	Develop an application (small game like scrabble, Tic-tac-Toe Using Android SDK)			37	

ADDITIONAL EXPERIMENTS

SI. No.	Name of the Experiment	Date of Experiment	Date of Submission	Page No	Faculty Signature
8	Implementing Publish/Subscribe paradigm using Web Services, ESB and JMS			48	
9	Implementing Stateful grid services using Globus WS-Core 4.0.3			53	

Introduction

Distributed Computing is a field of computer science that studies distributed systems. A distributed syste is a model in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. Thee significant characteristics of distributed systems are : concurrency of components, lack of a global clock, and independent failure of components. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

A computer program that runs in a distributed system is called a distributed program, and distributed programmming is the process of writing such programs. There are many alternatives for the message passing mechanism, including pure HTTP, RPC-like connectors and message queues.

A goal and challenge pursued by some computer scientists and practitioners in distributed systems is location transparency; however, this goal has fallen out of favour in industry, as distributed systems are different from conventional non-distributed systems, and the differences, such as network partitions, partial system failures, and partial upgrades, cannot simply be “prepared over” by attempts at “transparency”

APPLICATIONS OF DISTRIBUTED SYSTEM

There are two main reasons for using distributed systems and distributed computing. First, the very nature of the application may require the use of a communication network that connects several computers. For example, data is produced in one physical location and it is needed in another location.

Second, there are many cases in which the use of a single computer would be possible in principle, but the use of a distributed system is beneficial for practical reasons. For example, it may be more cost-efficient to obtain the desired level of performance by using a cluster of several low-end computers, in comparison with a single high-end computer. A distributed system can be more reliable than a non-distributed system, as there is no single point of failure. Moreover, a distributed system may be easier to expand and manage than a monolithic uniprocessor system.

Examples of distributed systems and applications of distributed computing include the following

Telecommunication networks:

- ✓ Telephone networks and cellular networks
- ✓ Computer networks such as the Internet.
- ✓ Wireless sensor networks.
- ✓ Routing algorithms

Network applications:

- ✓ World Wide Web and peer-to-peer networks
- ✓ Massively multiplayer online games and virtual reality communities
- ✓ Distributed databases and distributed database management systems.
- ✓ Network files systems.
- ✓ Distributed information processing systems such as banking systems and airline reservation systems

Real-time process control:

- ✓ Aircraft control systems
- ✓ Industrial control systems

Parallel computation:

- ✓ Scientific computing, including cluster computing and grid computing and various volunteer computing projects; see the list of distributed computing projects.

PROGRAM-1: Implementation FTP Client

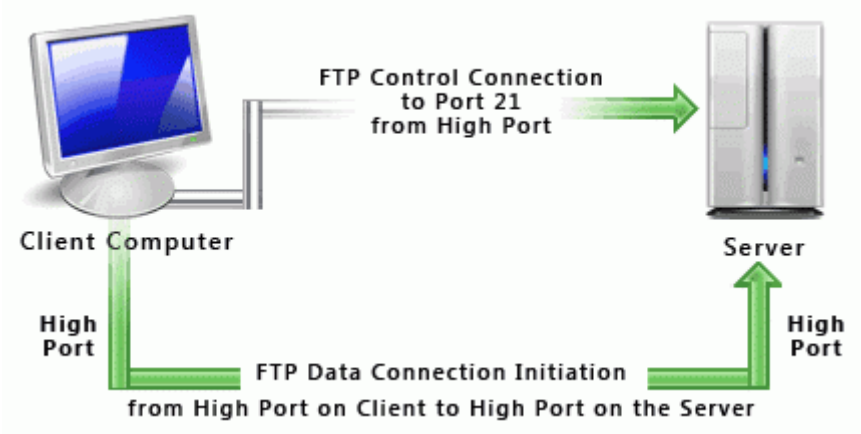
Description: The File Transfer Protocol

Aim: **To develop a client server application which implements File Transfer protocol. Let the client side request for files and the server side reads it and sends to the client**.

(FTP) is a standard network protocol used to transfer computer files from one host to another host over a TCP-based network, such as the internet.

FTP is built on client-server architecture and used separate control and data connections between the client and the server. FTP users may authenticate themselves using a clear-text sing-in-protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password and encrypts the content, FTP is often secured with SSL/TLS. SSH File Transfer Protocol is sometimes also used instead, but is technologically different.

The first FTP client applications were command line applications developed before operating systems had graphical user interfaces, and are still shipped with most Windows, UNIX, and Linux operating systems. Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware and FTP has been incorporated into productivity applications, such as Web page editors.



FTP Client:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
class One extends JFrame implements ActionListener
{
/* ctrl space */
```

```
public JButton b,b1;
public JLabel l;
public  JLabel l1,lmsg1,lmsg2;
One()
{
b=new JButton("Upload");
l=new JLabel("Uplaod a file : ");
lmsg1=new JLabel("");

b1=new JButton("Download");
l1=new JLabel("Downlaod a file");
lmsg2=new JLabel("");

setLayout(new GridLayout(2,3,10,10));
add(l);add(b);add(lmsg1);add(l1);add(b1);add(lmsg2);
b.addActionListener(this);
b1.addActionListener(this);
setVisible(true);
setSize(600,500);
}
public  void actionPerformed(ActionEvent e)
{
// TODO Auto-generated method stub
try {

/* String s=e.getActionCommand();
if(s.equals("Upload"))*/

if (b.getModel().isArmed())
{

Socket s=new Socket("localhost",1010);
System.out.println("Client connected to server");
JFileChooser j=new JFileChooser();
int val;
```

```
val=j.showOpenDialog(One.this);
String filename=j.getSelectedFile().getName();
String path=j.getSelectedFile().getPath();

PrintStream out=new PrintStream(s.getOutputStream());
out.println("Upload");
out.println(filename);
FileInputStream fis=new FileInputStream(path);
int n=fis.read();
while (n!=-1)
{
out.print((char)n);n=fis.read();
}
fis.close(); out.close();lmsg1.setText(filename+"is uploaded");
//s.close();
repaint();
}

if (b1.getModel().isArmed())
{
Socket s=new Socket("localhost",1010);
System.out.println("Client connected to server");
String remoteadd=s.getRemoteSocketAddress().toString();
System.out.println(remoteadd);
JFileChooser j1=new JFileChooser(remoteadd);
int val;
val=j1.showOpenDialog(One.this);
String filename=j1.getSelectedFile().getName();
String filepath=j1.getSelectedFile().getPath();

System.out.println("File name:"+filename);
PrintStream out=new PrintStream(s.getOutputStream());
out.println("Download");
out.println(filepath);
```

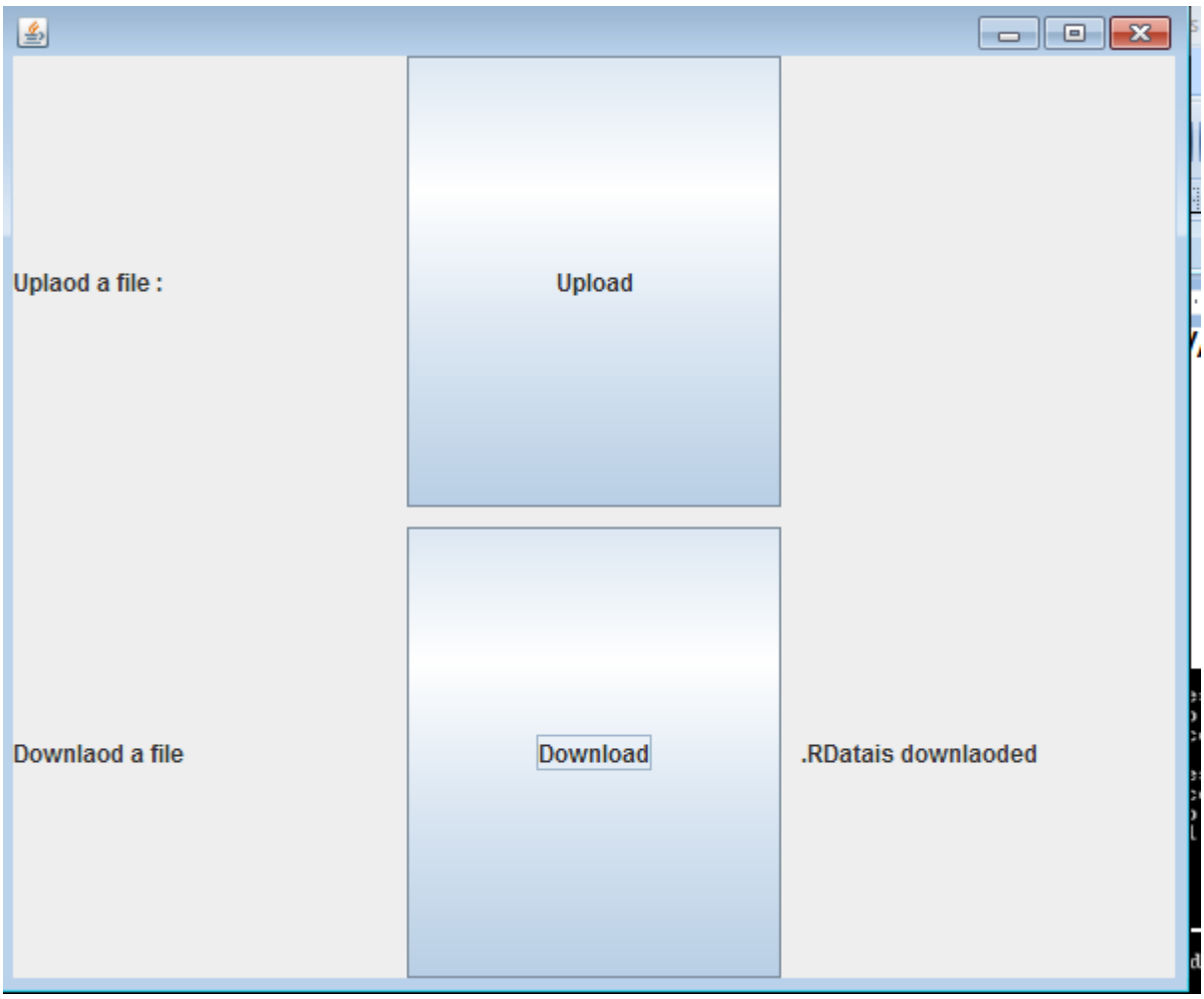
```
FileOutputStream fout=new FileOutputStream(filename);
DataInputStream fromserver=new DataInputStream(s.getInputStream());
int ch;
while ((ch=fromserver.read())!=-1)
                        {
fout.write((char) ch);
}
fout.close();//s.close();
lmsg2.setText(filename+"is downlaoded");
repaint();
}
}
catch (Exception ee)
{
// TODO: handle exception
System.out.println(ee);
}
}

}
public class FTPClient
{
public static void main(String[] args)
{
new One();
}
}
```

FTP Server:

```
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
public class FTPServer {
public static void main(String[] args)
{
try {
while (true)
{
ServerSocket ss=new ServerSocket(1010);
Socket sl=ss.accept();
System.out.println("Server scoket is created....");
System.out.println(" test1");
DataInputStream fromserver=new DataInputStream(sl.getInputStream());
System.out.println(" test2");
String option=fromserver.readLine();
if (option.equalsIgnoreCase("upload"))
{
System.out.println("upload test");
String filefromclient=fromserver.readLine();
File clientfile=new File(filefromclient);

FileOutputStream fout=new FileOutputStream(clientfile);
int ch;
while ((ch=fromserver.read())!=-1)
{
fout.write((char)ch);
}
fout.close();
```

```
}
if (option.equalsIgnoreCase("download"))
{
System.out.println("download test");
String filefromclient=fromserver.readLine();
File clientfile=new File(filefromclient);

FileInputStream fis=new FileInputStream(clientfile);
PrintStream out=new PrintStream(sl.getOutputStream());
int n=fis.read();
while (n!=-1)
{
out.print((char)n);
n=fis.read();
}
fis.close();
out.close();

} //while
}
}
catch (Exception e)
{
System.out.println(e);
// TODO: handle exception
}
}
}
```

Expected Expected Output




```
C:\Users\LAB4-57\Desktop>java FTPClient
Client connected to server
java.net.ConnectException: Connection refused: connect

C:\Users\LAB4-57\Desktop>java FTPClient
java.net.ConnectException: Connection refused: connect
Client connected to server
localhost/127.0.0.1:1010
File name:.RData
```

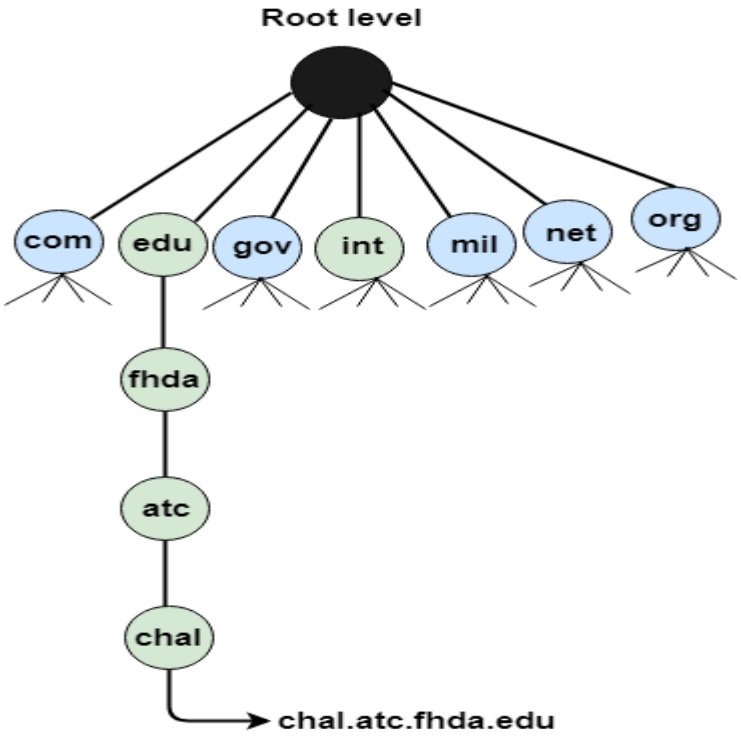
```
C:\Users\LAB4-57>cd desktop

C:\Users\LAB4-57\Desktop>javac FTPServer.java
Note: FTPServer.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\LAB4-57\Desktop>java FTPServer
Server scoket is created....
 test1
 test2
upload test
java.net.BindException: Address already in use: JVM_Bind

C:\Users\LAB4-57\Desktop>java FTPServer
Server scoket is created....
 test1
 test2
download test
java.net.BindException: Address already in use: JVM_Bind

C:\Users\LAB4-57\Desktop>java FTPServer
```

Result : Thus the implementation FTP Client was successfully done.

Program-2: Implementation of Name Server

Aim: Develop a client server application which implements Name Server. Let the client like a web browser sends a request containing a hostname, then a piece of software such as name server resolver sends a request to the name server to obtain the IP address of a hostname.

Description: Name server is a client / server network communication protocol. Name server clients send request to the server while name servers send response to the client. Client request contain a name which is converted into in IP address known as a forward name server lookups while requests containing an IP address which is converted into a name known as reverse name server lookups. Name server implements a distributed database to store the name of all the hosts available on the internet. If a client like a web browser sends a request containing a hostname, then a piece of software such as name server resolver sends a request to the name server to obtain the IP address of a hostname. If name server does not contain the IP address associated with a hostname then it forwards the request to another name server. It IP address has arrived at the resolver, which in turn completes the request over the internet protocol.

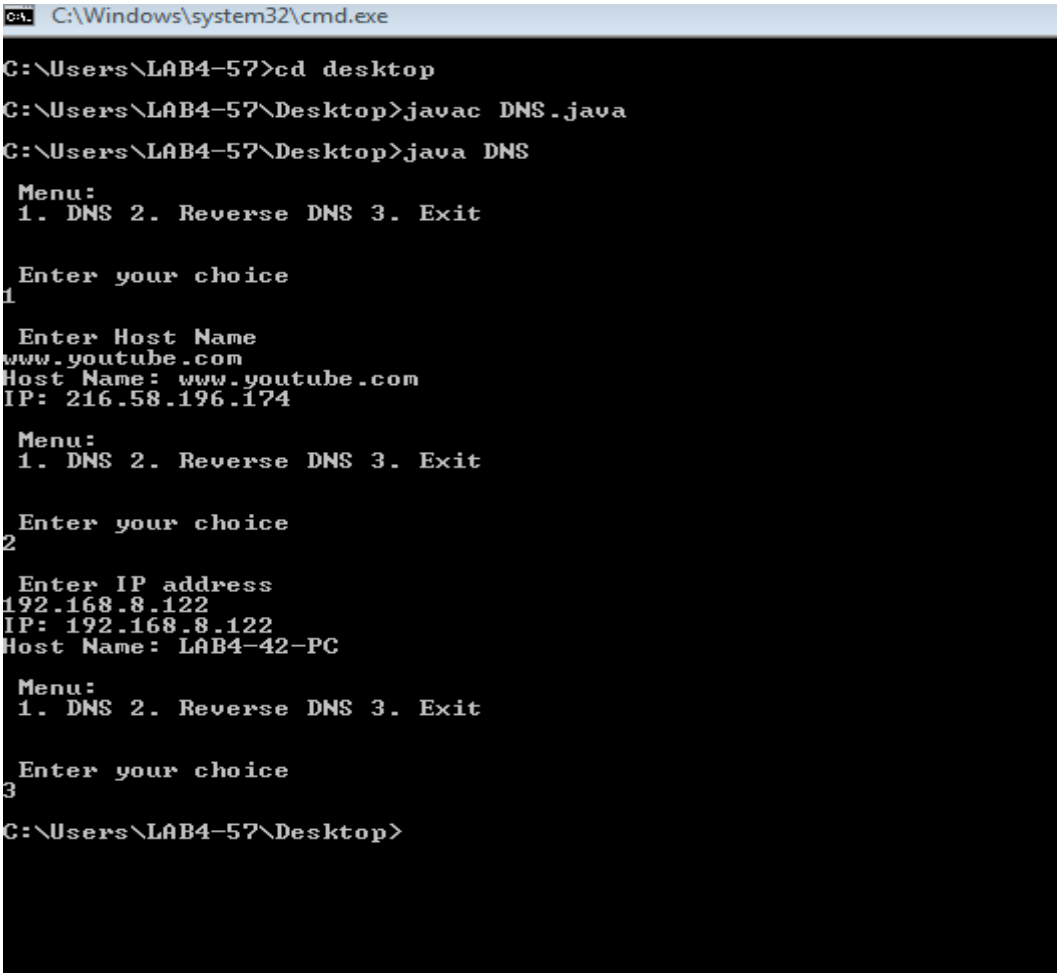


Program:

```
        import java.net.*;
        import java.io.*;
        import java.util.*;
        public class DNS
        {
public static void main(String[] args)
{
int n;
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
do
{
System.out.println("\n Menu: \n 1. DNS 2. Reverse DNS 3. Exit \n");
System.out.println("\n Enter your choice");
        n = Integer.parseInt(System.console().readLine());
if(n==1)
{
try
{
System.out.println("\n Enter Host Name ");
String hname=in.readLine();
InetAddress address;
address = InetAddress.getByName(hname);
System.out.println("Host Name: " + address.getHostName());
System.out.println("IP: " + address.getHostAddress());
}
catch(IOException ioe)
{
ioe.printStackTrace();
}
}
if(n==2)
{
try
{
```

```
System.out.println("\n Enter IP address");
String ipstr = in.readLine();
InetAddress ia = InetAddress.getByName(ipstr);
System.out.println("IP: "+ipstr);
System.out.println("Host Name: " +ia.getHostName());
}
catch(IOException ioe)
{
ioe.printStackTrace();
}
}
}while(!(n==3));
}}
```

Expected Output:



Result : Thus the implementation of Name Server was successfully done

Program-3: Implementation of Chat Server

Aim: To develop a client server application this implements Chat Server. Let the client side request for message and the server side displays it and sends to the client.

Description: A client / server program into a fully functioning chat client / server. A simple server that will accept a single client connection and display everything the client says on the screen. If the client user's types "OK" the client and the server will both quit. A server as before, but this time it will remain open for additional connection once a client has quit. The server can handle at most one connection at a time. A server as before but his time it can handle multiple clients simultaneously. The output from all connected clients will appear on the server's screen. A server as before, but his time it sends all text received from any of the connected clients to all clients. This means that the server has to receive and send the client has to send as well as receive.

Program:

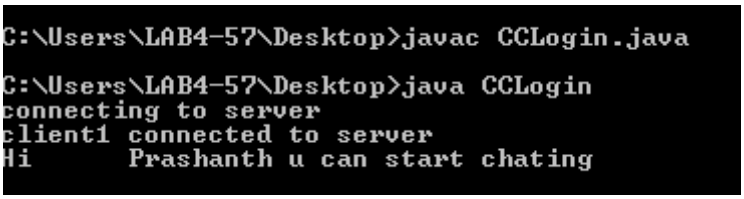
CCLogin.java

```
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import java.awt.GridLayout;

public class CCLogin implements ActionListener
{
JFrame frame1; JTextField tf,tf1; JButton button;
JLabel heading; JLabel label,label1;

public static void main(String[] paramArrayOfString)
{
new  CCLogin();
}
```

```
public  CCLogin()
{
this.frame1 = new JFrame("Login Page");
this.tf = new JTextField(10);
this.button = new JButton("Login");

this.heading = new JLabel("Chat Server");
this.heading.setFont(new Font("Impact", 1, 40));
this.label = new JLabel("Enter you Login Name");
this.label.setFont(new Font("Serif", 0, 24));

JPanel localJPanel = new JPanel();
this.button.addActionListener(this);
localJPanel.add(this.heading); localJPanel.add(this.label);
localJPanel.add(this.tf);
localJPanel.add(this.button);
this.heading.setBounds(30, 20, 280, 50);
this.label.setBounds(20, 100, 250, 60);
this.tf.setBounds(50, 150, 150, 30);
this.button.setBounds(70, 190, 90, 30);
this.frame1.add(localJPanel);
localJPanel.setLayout(null);
this.frame1.setSize(300,300);
this.frame1.setVisible(true);
this.frame1.setDefaultCloseOperation(3);
}

public void actionPerformed(ActionEvent paramActionEvent)
{
String str = "";
try
{
str = this.tf.getText();
this.frame1.dispose();
Client1 c1= new Client1(str);
```

```
c1.main(null);
}
catch(Exception localIOException)
{
}
}
}
```

EXPECTED OUTPUT:

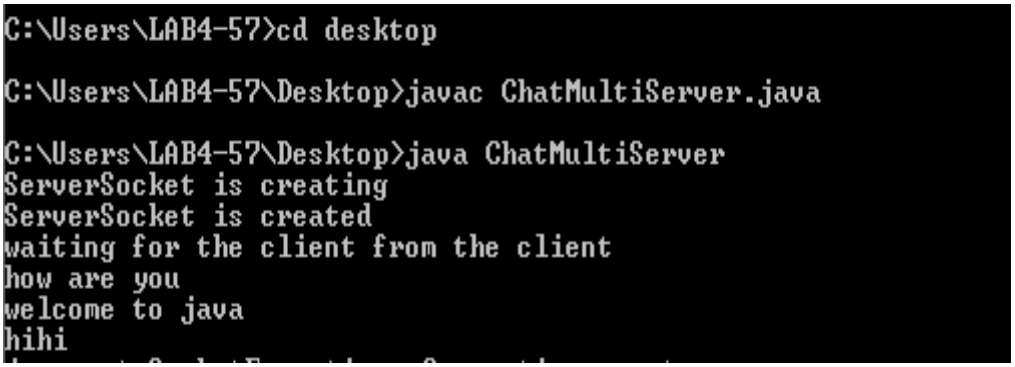


ChatMultiServer:

```
import java.net.*;
import java.io.*;
class A implements Runnable
{
Thread t;
Socket s;
A(Socket x)
{
s=x;
t=new Thread(this);
t.start();
}
public void run()
{
try
{
/* Reading data from client */
InputStream is=s.getInputStream();
byte data[]=new byte[50];
is.read(data);
String mfc=new String(data);
```

```
mfc=mfc.trim();
System.out.println(mfc);

/* Sending message to the server */
//System.out.println("Hi"+name+"u can start chating");
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
String n=br.readLine();
OutputStream os=s.getOutputStream();
os.write(n.getBytes());
}
catch(Exception e)
{
e.printStackTrace();
}
}
}

class ChatMultiServer
{
static int c=0;
public static void main(String args[]) throws Exception
{
System.out.println("ServerSocket is creating");
ServerSocket ss=new ServerSocket(1010);
System.out.println("ServerSocket is created");
System.out.println("waiting for the client from the client");

while(true)
{
Socket s=ss.accept();
new A(s);
}
}
}
```

EXPECTED OUTPUT:

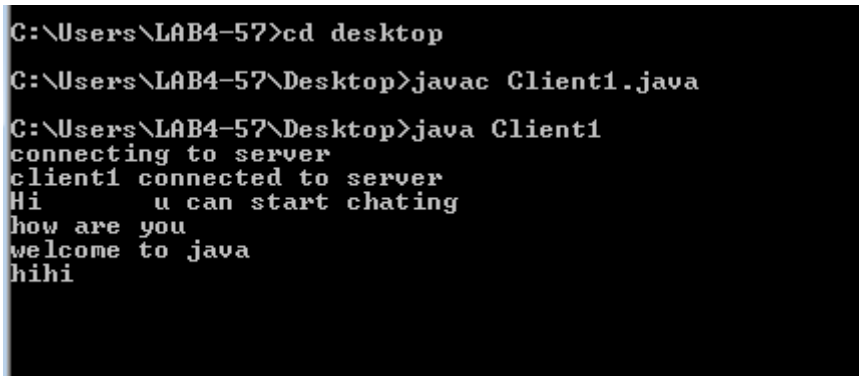


Client1.java

```
import java.net.*;
import java.io.*;
class Client1
{
static String name="";
public Client1(String n)
{
name=n;
}
public static void main(String args[]) throws Exception
{
System.out.println("connecting to server");
System.out.println("client1 connected to server");
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

/* Sending message to the server */
System.out.println("Hi\t"+name+" u can start chating");
while(true)
{
Socket s=new Socket("localhost",1010);
String n=br.readLine();
OutputStream os=s.getOutputStream();
os.write(n.getBytes());
```

```
/* Reading data from client */
InputStream is=s.getInputStream();
byte data[]=new byte[50];
is.read(data);
String mfc=new String(data);
mfc=mfc.trim();
System.out.println(mfc);


}
}
}
```

EXPECTED OUTPUT:



Result : Thus the Chat Server was successfully implemented

Program-4: Understanding of Working of NFS (includes exercises Configuration of NFS)

Aim: To understanding Network File System, distributed file system protocol allows a user on a client computer to access files over a network in the same implement the protocol.

Description: To access data stored on another machine (i.e., Server) the server would implement NFS daemon processes to make data available to clients. The server administrator determines what to make available and ensures it can recognize validated clients. From the client's side the machine requests access to exported data, typically by issuing a mount command. If successful the client machine can then view and interact with the file systems within the decided parameters.

Program:

Study of Network File Systems

1. Create a Folder nfs/abc.txt
2. Know the ipaddress

   Applications->System Settings->Network—edit (ipaddress, subnetmask)

   (or) In terminal type ifconfig
3. Enable the desired services
   1. System Services->Server Settings->Services
      - Network (Enable)
      - Nfs (Enable)
      - Iptables (Disable) (we do not firewalls)
   2. System Settings ->Security Level (Firewall options-disable, Selinux-disable)

Creation of Network File System Server

1. System Settings->Server Settings->NFS

   + Add (All are making security levels low)
2. Open Terminal

   Type: service nfs restart

   Creation of NFS Client

   Open terminal

   Type: df

   Type: mount –t nfs 135.135.5.120:/usr/nfs /root/abc

   cd abc

   ls : abc.txt

   Unmount: umount –t nfs 135.135.5.120:/usr/nfs

   Note: service network restart (if n/w is disabled use this)

Program-5: Implementation of Bulletin Board

Aim & Description: To develop a bulletin board system is a computer or an application dedicated to the sharing or exchange of messages or other files on a network. Originally an electronic version of the type of bulletin board found on the wall in any kitchens and work places, the bulletin board was used to post simple messages between users. The bulletin board become primary kind of online community before the World Wide Web arrived.

A bulletin board may be accessible from a dialup modem telnet or the internet. Because it originated before the graphical user interface become prevalent, the bulletin board system interface was text based. Although recent web based version have a graphical interactive user interface, the text only interface preferred by BBS purists can often be accessed by telnet.

Init.py

```
all= ['index', 'join', 'login', 'logout', 'list', 'write']
```

Index.py

```
from flask import render_template

from board.board_blueprint import board

@board.route('/')

def index():

return render_template('index.html', title='Index Page')
```

join.py

```
import pymysql

from flask import render_template, request, current_app

from board.board_blueprint import board

@board.route('/join')

def join():

return render_template('join.html')

@board.route('/join_process', methods=['POST'])

def join_process():
```

```
id = request.form['id']

password = request.form['password_1']

email = request.form['email']

print('id:[%s] password:[%s] email:[%s]' % (id, password, email))

db_address = current_app.config['DB_ADDRESS']

db_port = current_app.config['DB_PORT']

db_id = current_app.config['DB_ID']

db_password = current_app.config['DB_PASSWORD']

db_name = current_app.config['DB_NAME']

conn = pymysql.connect(host=db_address,

port=int(db_port),

user=db_id,

password=db_password,

db=db_name,

charset='utf8')

try:

cursor = conn.cursor()

sql = "INSERT INTO users(id, password, email) VALUES('%s', '%s', '%s')" % (id, password,
email)

cursor.execute(sql)

conn.commit()

finally:

conn.close()

return render_template('join.html', title='Member Join')
```

list.py

```
import pymysql

from flask import render_template, request, current_app

from board.board_blueprint import board

@board.route('/list', methods=['GET'])

def list():

page = request.args.get('page')

db_address = current_app.config['DB_ADDRESS']

db_port = current_app.config['DB_PORT']

db_id = current_app.config['DB_ID']

db_password = current_app.config['DB_PASSWORD']

db_name = current_app.config['DB_NAME']

conn = pymysql.connect(host=db_address,

port=int(db_port),

user=db_id,

password=db_password,

db=db_name,

charset='utf8')

try:

cursor = conn.cursor()

sql = "SELECT `no`, `content`, `writer`, `read` FROM board ORDER BY `write_time` DESC"

cursor.execute(sql)

rows = cursor.fetchall()

for row_data in rows:
```

```
print('no:[%s] content:[%s] writer:[%s] read:[%s]' % (row_data[0], row_data[1], row_data[2],
row_data[3]))

finally:

conn.close()

return render_template('list.html', rows=rows, title='Article List')
```

login.py

```
import pymysql

from flask import render_template, request, current_app, session, redirect, url_for

from functools import wraps

from board.board_blueprint import board

from board.board_logger import Log

def login_required(f):

@wraps(f)

def decorated_function(*args, **kwargs):

try:

session_key = request.cookies.get(current_app.config['SESSION_COOKIE_NAME'])

print('session_key:[%s]' % session_key)

is_login = False

if session.sid == session_key and session.__contains__('usn'):

is_login = True

if not is_login:

return redirect(url_for('.login_form', next=request.url))

return f(*args, **kwargs)

except Exception as e:
```

```
Log.error('Login error : %s' % str(e))

return decorated_function

@board.route('/login')

def login_page():

next_url = request.args.get('next', '')

id = request.args.get('id', '')

password = request.args.get('password', '')

return render_template('login.html')

@board.route('/login', methods=['POST'])

def login_process():

next_url = request.args.get('next')

id = request.form['id']

password = request.form['password']

db_address = current_app.config['DB_ADDRESS']

db_port = current_app.config['DB_PORT']

db_id = current_app.config['DB_ID']

db_password = current_app.config['DB_PASSWORD']

db_name = current_app.config['DB_NAME']

conn = pymysql.connect(host=db_address,

port=int(db_port),

user=db_id,

password=db_password,

db=db_name,

charset='utf8')
```

```
try:

cursor = conn.cursor()

sql = "SELECT `usn`, `id`, `email`, `update_time` FROM users WHERE `id`='%s' AND
`password`='%s' LIMIT 1" % (id, password)

print(sql)

cursor.execute(sql)

rows = cursor.fetchall()

if rows:

for row_data in rows:

session.parmanent = True

usn = row_data[0]

id = row_data[1]

email = row_data[2]

update_time = row_data[3]

print('usn:[%s] id:[%s] email:[%s] update_time:[%s]' % (usn, id, email, update_time))

session['usn'] = usn

session['user'] = id

session['email'] = email

if next_url != '' and next_url != None:

return redirect(url_for(next_url))

else:

return redirect(url_for('.list'))

else:

print('Cannot found user')
```

```
finally:

conn.close()

return render_template('login.html', next_url=next_url, title='Member Login')
```

logout.py

```
from flask import session, redirect, url_for

from board.board_blueprint import board

@board.route('/logout')

def logout():

session.clear()

return redirect(url_for('.list'))
```

write.py

```
import pymysql

from flask import render_template, request, redirect, url_for, current_app

from board.board_blueprint import board

@board.route('/write', methods=['GET', 'POST'])

def write():

if request.method == 'POST':

writer = request.form['writer']

content = request.form['content']

if writer and content:

print('POST writer:[%s] content:[%s]' % (writer, content))

else:

return render_template('write.html')

db_address = current_app.config['DB_ADDRESS']
```

```
db_port = current_app.config['DB_PORT']

db_id = current_app.config['DB_ID']

db_password = current_app.config['DB_PASSWORD']

db_name = current_app.config['DB_NAME']

conn = pymysql.connect(host=db_address,

port=int(db_port),

user=db_id,

password=db_password,

db=db_name,

charset='utf8')

try:

cursor = conn.cursor()

sql = "INSERT INTO board(writer, content) VALUES(%s, '%s')" % (writer, content)

print(sql)

cursor.execute(sql)

conn.commit()

finally:

conn.close()

return redirect(url_for('.list'))

return render_template('write.html', title='Article Write')
```

Board_blueprint.py

```
from flask import Blueprint

board = Blueprint('bikeparking', __name__, template_folder='../templates', static_folder='../static')
```

board_config.py

```
class FlaskBoardConfig(object):

DB_ADDRESS = ''

DB_PORT = ''

DB_ID = ''

DB_PASSWORD = ''

DB_NAME = ''
```

Board_logger.py

```
import logging

from logging import getLogger, handlers, Formatter

class Log:

__log_level_map = {

'debug' : logging.DEBUG,

'info' : logging.INFO,

'warn' : logging.WARN,

'error' : logging.ERROR,

'critical' : logging.CRITICAL

}

__my_logger = None

@staticmethod

def init(logger_name='flaskboardlog', log_level='debug', log_filepath='board/logs/log.log'):

Log.__my_logger = getLogger(logger_name);

Log.__my_logger.setLevel(Log.__log_level_map.get(log_level, 'warn'))

formatter = Formatter('%(asctime)s - %(levelname)s - %(message)s')

console_handler = logging.StreamHandler()
```

```
console_handler.setFormatter(formatter)

Log.__my_logger.addHandler(console_handler)

file_handler = \

handlers.TimedRotatingFileHandler(log_filepath, when='D', interval=1)

file_handler.setFormatter(formatter)

Log.__my_logger.addHandler(file_handler)

@staticmethod

def debug(msg):

Log.__my_logger.debug(msg)

@staticmethod

def info(msg):

Log.__my_logger.info(msg)

@staticmethod

def warn(msg):

Log.__my_logger.warn(msg)

@staticmethod

def error(msg):

Log.__my_logger.error(msg)

@staticmethod

def critical(msg):

Log.__my_logger.critical(msg)
```

Expected Output:

Result : Thus the implementation of Bulletin Board was successfully done

Program-6: Implement a word count application which counts the number of occurrences of each words a large collection of documents Using Map Reduce model.

Aim: To develop to implement a word count application which counts the number of occurrences of each words a large collection of documents Using Map Reduce model.

Description: In Hadoop, MapReduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of task can be joined together to compute final results.

MapReduce consists of 2 steps:

Map Function – it takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair)

Example - (Map function in word count)

Input	Set of data	Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN
Output	Convert into another set of data (Key, Value)	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)

- **Reduce Function** –Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

  Example – (Reduce function in word count)

Input **(output of Map function)**	Set of Tuples	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)
Output	Converts into smaller set of tuples	(BUS,7), (CAR,7), (TRAIN,4)

Work Flow of the program:

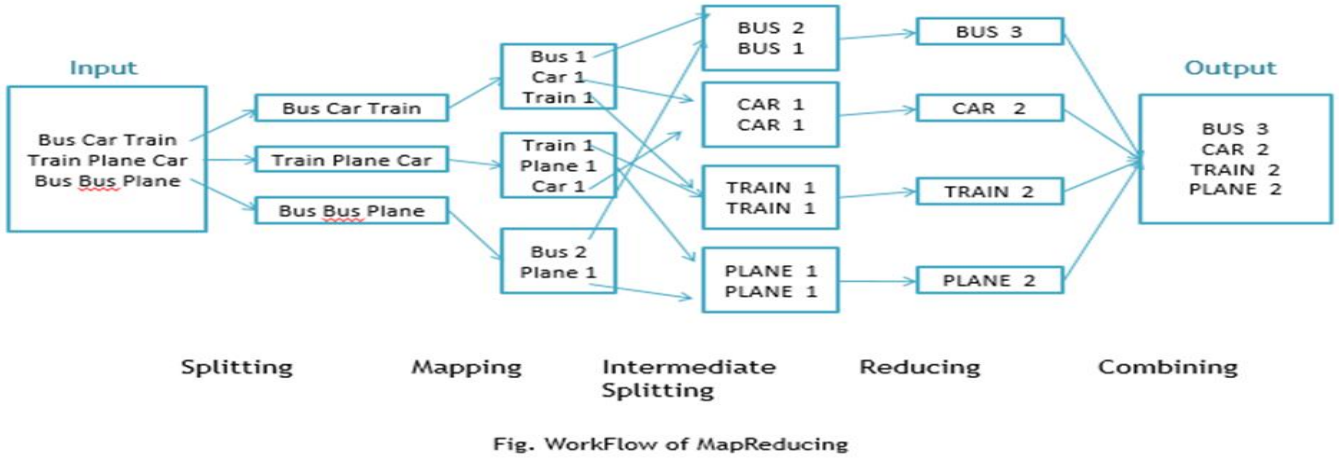


Fig. WorkFlow of MapReducing

Workflow of Map Reduce consists of 5 steps:

Splitting – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').

Mapping – as explained above.

Intermediate splitting – the entire process in parallel on different clusters. In order to group them in "Reduce Phase" the similar KEY data should be on the same cluster.

Reduce – it is nothing but mostly group by phase.

Combining – The last phase where all the data (individual result set from each cluster) is combined together to form a result.

We need to write the splitting parameter, Map function logic, and Reduce function logic. The rest of the remaining steps will execute automatically.

Make sure that Hadoop is installed on your system with the Java SDK.

Steps

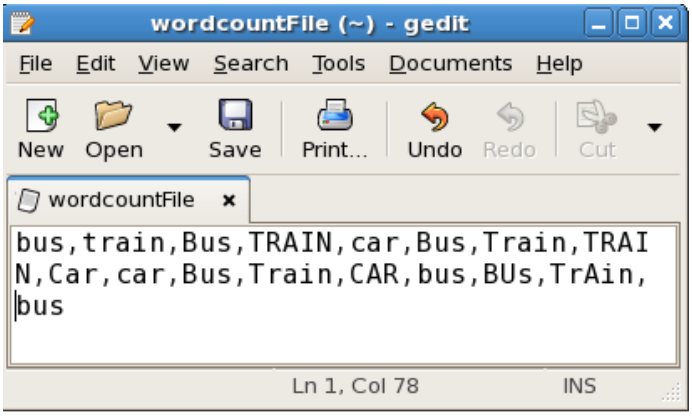
1. Open Eclipse> File > New > Java Project >(Name it – MRProgramsDemo) > Finish.
2. Right Click > New > Package (Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - WordCount).
4. Add Following Reference Libraries:
    1. Right Click on Project > Build Path> Add External
        1. /usr/lib/hadoop-0.20/***hadoop-core.jar***
        2. Usr/lib/hadoop-0.20/lib/***Commons-cli-1.2.jar***

```
package PackageDemo;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new Configuration();
String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
Path input=new Path(files[0]);
Path output=new Path(files[1]);
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.class);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{
public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException
{
```

```
String line = value.toString();
String[] words=line.split(",");
for(String word: words )
{
Text outputKey = new Text(word.toUpperCase().trim());
IntWritable outputValue = new IntWritable(1);
con.write(outputKey, outputValue);
}
}
}
public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException,
InterruptedException
{
int sum = 0;
for(IntWritable value : values)
{
sum += value.get();
}
con.write(word, new IntWritable(sum));
}
}
}
```

Expected Output

1. Take a text file and move it into HDFS format:



To move this into Hadoop directly, open the terminal and enter the following commands:

[training@localhost ~]$ hadoop fs -put wordcountFile wordCountFile

2. Run the jar file:

(Hadoop jar jarfilename.jar packageName.ClassName PathToInputTextFile PathToOutputDirectry)

[training@localhost ~]$ hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile MRDir1

3. Open the result:

[training@localhost ~]$ hadoop fs -ls MRDir1

Found 3 items

-rw-r--r-- 1 training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/_SUCCESS

drwxr-xr-x - training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/_logs

-rw-r--r-- 1 training supergroup 20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000

[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000

BUS 7

CAR 4

TRAIN 6

Result: A word count application which counts the number of occurrences of each word a large collection of documents Using Map Reduce model was successfully developed.

Program-7: Develop an application (small game like scrabble, Tic-tac-Toe Using Android SDK)

Aim & Description: Creating the Board, First step is to create the Board for the Tic-Tac-Toe game. The Board class will store the elements of the grid in an array and will contain a Boolean indicating if the game is ended or no.

The play method will let you to set the mark of the currentPlayer on the grid at a given (x, y) position. A changePlayer method will be used to change the current player for the next play. Besides, a computer method is defined to let the user to randomly place a mark on the grid. Finally, we define a checkEnd method to check if the game is ended. The game is ended if there is a winner or a draw: all the cases of the grids are filled and no one wins the game.

This gives us the following code for the *Board* class:

```
package com.ssaurel.tictactoe;
  import java.util.Random;
  public class Board {
  private static final Random RANDOM = new Random();
  private char[] elts;
  private char currentPlayer;
  private boolean ended;
  public Board() {
  elts = new char[9];
  newGame();
  }
  public boolean isEnded() {
  return ended;
  }
  public char play(int x, int y) {
  if (!ended  &&  elts[3 * y + x] == ' ') {
  elts[3 * y + x] = currentPlayer;
  changePlayer();
  }
  return checkEnd();
  }
  public void changePlayer() {
  currentPlayer = (currentPlayer == 'X' ? 'O' : 'X');
```

```
}
public char getElt(int x, int y) {
return elts[3 * y + x];
}
public void newGame() {
for (int i = 0; i  < elts.length; i++) {
elts[i] = ' ';
}
currentPlayer = 'X';
ended = false;
}
public char checkEnd() {
for (int i = 0; i < 3; i++) {
if (getElt(i, 0) != ' ' &&
getElt(i, 0) == getElt(i, 1)  &&
getElt(i, 1) == getElt(i, 2)) {
ended = true;
return getElt(i, 0);
}
if (getElt(0, i) != ' ' &&
getElt(0, i) == getElt(1, i)  &&
getElt(1, i) == getElt(2, i)) {
ended = true;
return getElt(0, i);
}
}
if (getElt(0, 0) != ' '  &&
getElt(0, 0) == getElt(1, 1)  &&
getElt(1, 1) == getElt(2, 2)) {
ended = true;
return getElt(0, 0);
}
if (getElt(2, 0) != ' '  &&
getElt(2, 0) == getElt(1, 1)  &&
getElt(1, 1) == getElt(0, 2)) {
```

```
ended = true;
return getElt(2, 0);
}
for (int i = 0; i < 9; i++) {
if (elts[i] == ' ')
return ' ';
}
return 'T';
}
public char computer() {
if (!ended) {
int position = -1;
do {
position = RANDOM.nextInt(9);
} while (elts[position] != ' ');
elts[position] = currentPlayer;
changePlayer();
}
return checkEnd();
}
}
```

Rendering the Board on the Screen

Next step is to create a BoardView class to render our Board on the screen. Our BoardView will extend the View class and we will draw the Board and its elements on the Canvas object associated. It is a good way to discover how to draw simple shapes on a Canvas of a specific View too.

Furthermore, we must manage the touch events of the users on the *Board* to let it to play to our Tic-Tac-Toe game. For that, we override the on TouchEvent method from the *View* parent class. In that method, we convert a point touched on the screen to a case on our grid. Then, we make the play on the *Board* object. After that, we need to call the gameEnded method of the parent activity if the game is ended to display the win dialog to the user. If not, we make the play for the computer. Like you can see, the heart of the logic game will be located in this method.

This gives us the following code for the *BoardView* object :

```
package com.ssaurel.tictactoe;
 import android.content.Context;
```

```
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.support.annotation.Nullable;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
public class BoardView extends View {
private static final int LINE_THICK = 5;
private static final int ELT_MARGIN = 20;
private static final int ELT_STROKE_WIDTH = 15;
private int width, height, eltW, eltH;
private Paint gridPaint, oPaint, xPaint;
private GameEngine gameEngine;
private MainActivity activity;
public BoardView(Context context) {
super(context);
}
public BoardView(Context context, @Nullable AttributeSet attrs) {
super(context, attrs);
gridPaint = new Paint();
oPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
oPaint.setColor(Color.RED);
oPaint.setStyle(Paint.Style.STROKE);
oPaint.setStrokeWidth(ELT_STROKE_WIDTH);
xPaint = new Paint(oPaint);
xPaint.setColor(Color.BLUE);
}
public void setMainActivity(MainActivity a) {
activity = a;
}
public void setGameEngine(GameEngine g) {
gameEngine = g;
}
@Override
```

```
protected void onMeasure(int widthMeasureSpec, int

height = View.MeasureSpec.getSize(heightMeasureSpec);
width = View.MeasureSpec.getSize(widthMeasureSpec);
eltW = (width - LINE_THICK) / 3;
eltH = (height - LINE_THICK) / 3;
setMeasuredDimension(width, height);
}
@Override
protected void onDraw(Canvas canvas) {
drawGrid(canvas);
drawBoard(canvas);
}
@Override
public boolean onTouchEvent(MotionEvent event) {
if (!gameEngine.isEnded()  &&  event.getAction() ==
N) {
int x = (int) (event.getX() / eltW);
int y = (int) (event.getY() / eltH);
char win = gameEngine.play(x, y);
invalidate();
if (win != ' ') {
activity.gameEnded(win);
} else {
// computer plays ...
win = gameEngine.computer();
invalidate();
if (win != ' ') {
activity.gameEnded(win);
}
}
}
return super.onTouchEvent(event);
}
private void drawBoard(Canvas canvas) {
```

```
for (int i = 0; i < 3; i++) {
for (int j = 0; j < 3; j++) {
drawElt(canvas, gameEngine.elt(i, j), i, j);
}
}
}
private void drawGrid(Canvas canvas) {
for (int i = 0; i < 2; i++) {
// vertical lines
float left = eltW * (i + 1);
float right = left + LINE_THICK;
float top = 0;
float bottom = height;
canvas.drawRect(left, top, right, bottom, gridPaint);
// horizontal lines
float left2 = 0;
float right2 = width;
float top2 = eltH * (i + 1);
float bottom2 = top2 + LINE_THICK;
canvas.drawRect(left2, top2, right2, bottom2, gridPaint);
}
}
private void drawElt(Canvas canvas, char c, int x, int y) {
if (c == 'O') {
float cx = (eltW * x) + eltW / 2;
float cy = (eltH * y) + eltH / 2;
canvas.drawCircle(cx, cy, Math.min(eltW, eltH) / 2 - ELT_MARGIN *

} else if (c == 'X') {
float startX = (eltW * x) + ELT_MARGIN;
float startY = (eltH * y) + ELT_MARGIN;
float endX = startX + eltW - ELT_MARGIN * 2;
float endY = startY + eltH - ELT_MARGIN;
canvas.drawLine(startX, startY, endX, endY, xPaint);
float startX2 = (eltW * (x + 1)) - ELT_MARGIN;
```

```
float startY2 = (eltH * y) + ELT_MARGIN;
float endX2 = startX2 - eltW + ELT_MARGIN * 2;
float endY2 = startY2 + eltH - ELT_MARGIN;
canvas.drawLine(startX2, startY2, endX2, endY2, xPaint);
}
}
}
```

Creating the UI for our Game

The biggest part of the user interface of our Tic-Tac-Toe game is managed in the BoardView class. So, we just need to set our BoardView component into a RelativeLayout parent *View* in our layout file:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.ssaurel.tictactoe.MainActivity">
<com.ssaurel.tictactoe.BoardView
android:id="@+id/board"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_centerInParent="true"/>
</RelativeLayout>
```

Starting a new Game

To start a new game, the user will have to click on a load item in the action bar of our application. So, we add the item in a main.xml menu file under /res/menu:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto">
<item
android:id="@+id/action_new_game"
android:orderInCategory="50"
```

```
android:title="New Game"
app:showAsAction="always" />
</menu>
```

Assemble all the pieces of the puzzle

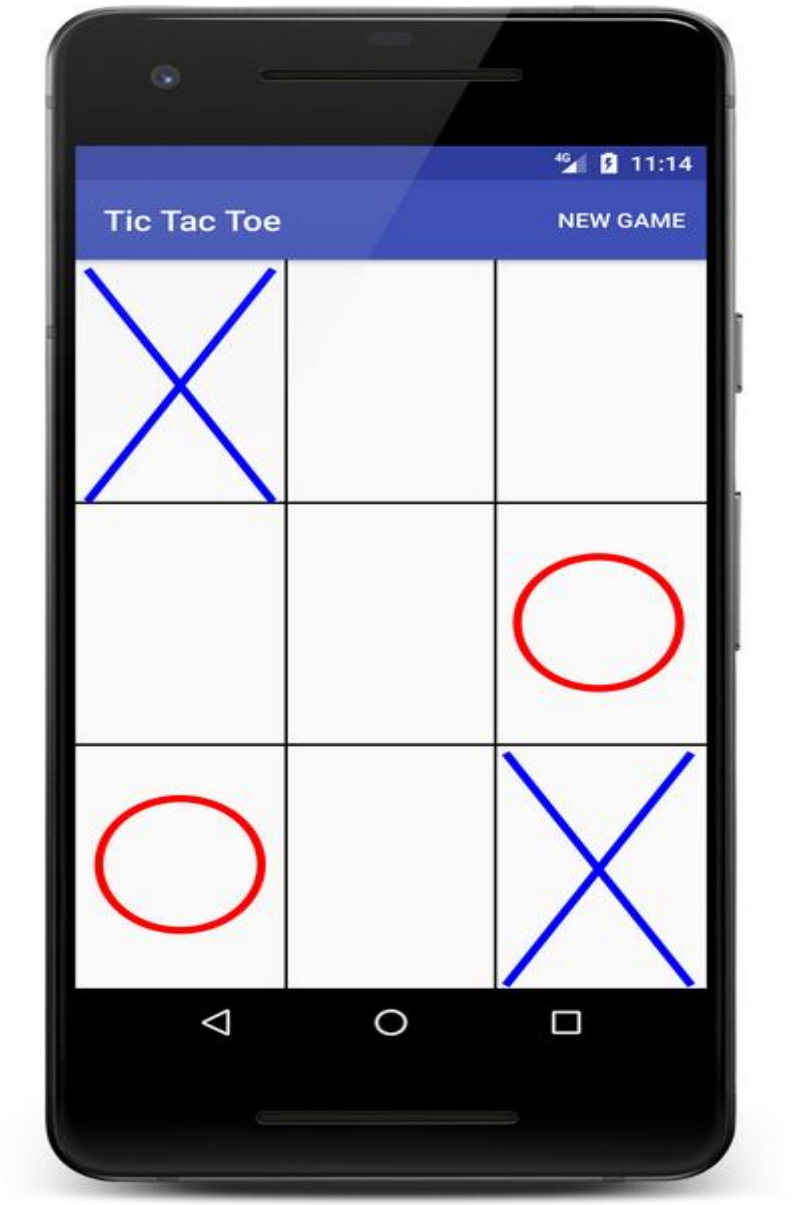
Last step is to assemble all the components created previously in the MainActivity class. In the onCreate method, we create the Board object and then we pass it in parameter of the BoardView got from the main layout of the application. Then, we connect the new game item of the action bar with the newGame method of the Board object to create a new game when the user will click on it. Finally, we define the gameEnded method which was called in the BoardView object.

This gives us the following code for our MainActivity :

```
package com.ssaurel.tictactoe;
  import android.content.DialogInterface;
  import android.os.Bundle;
  import android.support.v7.app.AlertDialog;
  import android.support.v7.app.AppCompatActivity;
  import android.view.Menu;
  import android.view.MenuItem;
  import static com.ssaurel.tictactoe.R.id.board;
  public class MainActivity extends AppCompatActivity {
  private BoardView boardView;
  private GameEngine gameEngine;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  boardView = (BoardView) findViewById(board);
  gameEngine = new GameEngine();
  boardView.setGameEngine(gameEngine);
  boardView.setMainActivity(this);
  }
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
```

```
getMenuInflater().inflate(R.menu.main, menu);
return super.onCreateOptionsMenu(menu);
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
if (item.getItemId() == R.id.action_new_game) {
newGame();
}
return super.onOptionsItemSelected(item);
}
public void gameEnded(char c) {
String msg = (c == 'T') ? "Game Ended. Tie" : "GameEnded. " + c + " win";
new AlertDialog.Builder(this).setTitle("Tic Tac Toe").
setMessage(msg).
setOnDismissListener(new DialogInterface.OnDismissListener() {
@Override
public void onDismiss(DialogInterface dialogInterface) {
newGame();
}
}).show();
}
private void newGame() {
gameEngine.newGame();
boardView.invalidate();
}
}
```

Expected Output:



Playing to the Tic-Tac-Toe Game

The game works great and finally, we win the game against the computer which is logical because our Artificial Intelligence (AI) is really basic:

Result : Thus the Tic-tac-Toe game application was developed successfully.

ADDITIONAL PROGRAMS

PROGRAM-8

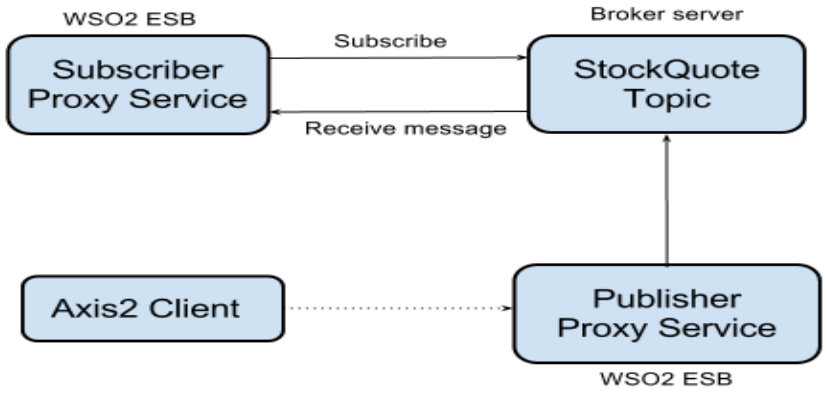
Aim: **Implementing Publish/Subscribe Paradigm using Web Services, ESB and JMS**

Description: JMS supports two models for messaging as follows:

- Queues: point-to-point
- Topics: publish and subscribe

There are many business use cases that can be implemented using the publisher-subscriber pattern. For example, consider a blog with subscribed readers. The blog author posts a blog entry, which the subscribers of that blog can view. In other words, the blog author publishes a message (the blog post content) and the subscribers (the blog readers) receive that message. Popular publisher / subscriber patterns like these can be implemented using JMS topics as described in the following

- Configuring the broker server
- Configuring the publisher
- Configuring the subscribers
- Publishing the topic



Configuring the broker server

We will use ActiveMQ as out broker server. Configure with ActiveMQ to set up ActiveMq for use with WSO2 ESB

Configuring the publisher

1.Open the <ESB_HOME>/repository/conf/JNDI.properties file and specify the JNDI designation of the topic (in this example, SimpleStockQuoteService). For example:

```
# register some queues in JNDI using the form
# queue.[jndiName] = [physicalName]
queue.MyQueue = example.MyQueue
# register some topics in JNDI using the form
# topic.[jndiName] = [physicalName]
```

topic.MyTopic = example.MyTopic

topic.SimpleStockQuoteService = SimpleStockQuoteService

2. Next, add a proxy service named StockQuoteProxy and configure it to publish to the topic SimpleStockQuoteService. You can add the proxy service to the ESB using the management console, either by building the proxy service in the design view or by copying the XML configuration into the source view. Alternatively, you can add an XML file named StockQuoteProxy.xml to <ESB_HOME>/repository/deployment/server/synapse-configs/ default/ proxy-services. A sample XML code segment that defines the proxy service is given below. Notice that the address URI specifies properties for configuring the JMS transport

```
<definitions xmlns="http://ws.apache.org/ns/synapse">
<proxy name="StockQuoteProxy"
transports="http"
startOnLoad="true"
trace="disable">
<target>
<endpoint>
<address
uri="jms:/SimpleStockQuoteService?transport.jms.ConnectionFactoryJNDIName=TopicConnection
Factory&amp;amp;java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory
&amp;amp;java.naming.provider.url=tcp://localhost:61616&amp;amp;transport.jms.DestinationType=topic"/>
</endpoint>
<inSequence>
<property name="OUT_ONLY" value="true"/>
</inSequence>
<outSequence>
<send/>
</outSequence>
</target>
</proxy>
</definitions>
```

Configuring the subscribers

```
<definitions xmlns="http://ws.apache.org/ns/synapse">
<proxy name="SimpleStockQuoteService1"
transports="jms"
startOnLoad="true"
trace="disable">
<description/>
<target>
<inSequence>
<property name="OUT_ONLY" value="true"/>
<log level="custom">
<property name="Subscriber1" value="I am Subscriber1"/>
</log>
<drop/>
</inSequence>
<outSequence>
<send/>
</outSequence>
</target>
<parameter name="transport.jms.ContentType">
<rules>
<jmsProperty>contentType</jmsProperty>
<default>application/xml</default>
</rules>
</parameter>
<parameter name="transport.jms.ConnectionFactory">myTopicConnectionFactory</parameter>
<parameter name="transport.jms.DestinationType">topic</parameter>
<parameter name="transport.jms.Destination">SimpleStockQuoteService</parameter>
</proxy>
<proxy name="SimpleStockQuoteService2"
transports="jms"
```

```
startOnLoad="true"
trace="disable">
<description/>
<target>
<inSequence>
<property name="OUT_ONLY" value="true"/>
<log level="custom">
<property name="Subscriber2" value="I am Subscriber2"/>
</log>
<drop/>
</inSequence>
<outSequence>
<send/>
</outSequence>
</target>
<parameter name="transport.jms.ContentType">
<rules>
<jmsProperty>contentType</jmsProperty>
<default>application/xml</default>
</rules>
</parameter>
<parameter name="transport.jms.ConnectionFactory">myTopicConnectionFactory</parameter>
<parameter name="transport.jms.DestinationType">topic</parameter>
<parameter name="transport.jms.Destination">SimpleStockQuoteService</parameter>
</proxy>
</definitions>
```

Next, you configure two proxy services that subscribe to the JMS topic SimpleStockQuoteService, so that whenever this topic receives a message, it is sent to these subscribing proxy services. Following is the sample configuration for these proxy services.

Publishing to the topic

Start the ESB with one of the following commands:

<ESB_HOME>/bin/wso2server.sh (on Linux)

<MB_HOME>/bin/wso2server.bat (on Windows)

A log message similar to the following will appear:

INFO {org.wso2.andes.server.store.CassandraMessageStore} - Created Topic : SimpleStockQuoteService

INFO {org.wso2.andes.server.store.CassandraMessageStore} -

Registered Subscription tmp_127_0_0_1_44759_1 for Topic SimpleStockQuoteService

To invoke the publisher, use the sample stockquote client service by navigating to <ESB_HOME>/samples/axis2Client and running the following command:
ant stockquote -Daddurl=http://localhost:8280/services/StockQuoteProxy -Dmode=placeorder -Dsymbol=MSFT

The message flow is executed as follows:
When the stockquote client sends the message to the StockQuoteProxy service, the publisher is invoked and sends the message to the JMS topic.
The topic delivers the message to all the subscribers of that topic. In this case, the subscribers are ESB proxy services.

Result : Implementing Publish/Subscribe Paradigm using Web Services, ESB and JMS is successfully executed

PROGRAM-9

Aim: Implementing Stateful grid services using Globus WS-Core-4.0.3

Procedure:

1. Define the web service's interface. This is done with WSDL.
2. Implement the web service in Java.
3. Define the deployment parameters by using WSDD and JNDI.
4. Compile everything and generate a GAR file using Ant.
5. Deploy service using Globus WS-Core-4.0.3 GT4 tool

VIVA Questions:

1. What is a Distributed Systems?
2. Give few examples of distributed systems?
3. What is the Deference between Networked System and Distributed System?
4. Name few characteristics of Distributed Systems?
5. Name Some Case Studies of Distributed Systems which you have studied?
6. If you are said to design a Distributed Systems for your Client which design issues you are going to consider?
7. Explain the TCP and UDP Protocols
8. What is a Distributed Systems?
9. Give few examples of distributed systems?
10. What is the Deference between Networked System and Distributed System?
11. Name few characteristics of Distributed Systems?
12. Name Some Case Studies of Distributed Systems which you have studied?
13. If you are said to design a Distributed Systems for your Client which design issues you are going to consider?
14. Explain the TCP and UDP Protocols
15. What are Di challenges faced by Distributed Systems?
16. Name Popular System Models in Distributed Systems?
17. Explain the Deference between Messages oriented Communication and Stream Oriented Communication.
18. What are Layered Protocols?
19. What are RPC and LRPC?
20. What is the advantage of RPC 2 over RPC?
21. How do we provide security to RMI classes?
22. What are Layered Protocols?
23. What is Remote Method Invocation?
24. What is Distributed File System (DFS)?
25. What do you mean by Auto mounting?
26. What is the advantage of RPC2 over RPC?
27. What are advances in CODA as to AFS?
28. Which is the most Important Feature of CODA?
29. What are Stubs and Skeletons?
30. How communication does takes place in NFS?
31. Explain the Naming concept in NFS?

32. How Synchronization takes place in NFS?
33. How do you implement locking in NFS?
34. What is vice and Virtue related to CODA?