DISTRIBUTED SYSTEMS LAB [PC752CS] LABORATORY MANUAL

For

B.E - VII Semester

Prepared By

Dr.Shruthi SK

Assistant Professor

Name: ……..

Roll. No: ……..

Academic Year: ……..

METHODIST
College of Engineering & Technology

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

College Code : 1607

Dr. Prabhu G Benakop
B.E., M.E., Ph.D.
SM., IEEE, LMISTE, LMISOI

Principal

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.

PRINCIPAL

PRINCIPAL
METHODIST COLLEGE OF ENGG.& TECH.
King Koti Road, Abids, Hyderabad.

METHODIST

Estd : 2008

COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To become a leader in providing Computer Science and 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.

METHODIST

Estd : 2008

COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES (POs)

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 modeling 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 (PSOs)

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.

METHODIST

Estd : 2008

COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

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 and 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
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Estd : 2008

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, misbehaviors 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
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Estd : 2008
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

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Estd : 2008 Address : King Koti Road, Abids, Hyderabad, Telangana, 500001 | Email : principal@methodist.edu.in

List of COs

DISTRIBUTED SYSTEMS LAB (PC 752 CS)

Course Code	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

DS Lab (PC 752 CS) CO-PO Mapping

PO / CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PS01	PSO2	PSO3
PC752CS.1	2	3	2		3				2				1		
PC752CS.2	2	2		1	2										
PC752CS.3	2	1		1	3				2			1	1		
PC752CS.4	3	1	1	2	2										
PC752CS.5	3	3	3		2				2			2	1		
PC752CS.6	3	3	3		2				3			2	1		
PC752CS	2.5	2.1	2.2	1.3	2.3				2.2			1.6	1.0		

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Estd : 2008 Address : King Koti Road, Abids, Hyderabad, Telangana, 500001 | Email : principal@methodist.edu.in

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			
-	-	-	-	3	**25**	**50**	**1.5**

Course Objectives

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

Course Outcomes

After completing this course, the student will be able to

- Write programs that communicate data between two hosts
- Configure NFS
- To implement inter process communication and remote communication
- Use distributed data processing frameworks and mobile application tool kits

List of Experiments to be performed:

1. Implementation FTPClient
2. Implementation of NameServer
3. Implementation of ChatServer
4. Understanding of working of NFS (Includes exercises on Configuration ofNFS)
5. Write a program to implement hello world service using RPC or Write a program to implement date service using RPC.
6. Implement a word count application which counts the number of occurrences of each word a large collection of documents Using Map Reducemodel.
7. Develop an application using 3 -tier architectures.

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Estd : 2008 Address : King Koti Road, Abids, Hyderabad, Telangana, 500001 | Email : principal@methodist.edu.in

Exp. No.	Name of the Experiment	Date of Conducted	Date of Submission	Faculty Signature
1.	Implementation FTP Client			
2	Implementation of Name Server			
3	Implementation of Chat Server			
4	Understanding of Working of NFS (includes exercises Configuration of NFS)			
5.	Write a program to implement Hello world service using RPC.(It uses Publisher & Subscriber model to coordinate.)			
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 using 3-tier architectures.			
	ADDITIONAL EXPERIMENTS			
1	Write a program to implement Berkely Algorithm			
2	Write a program to implement Lamport clock Algorithm			

Introduction

Distributed Computing is a field of computer science that studies distributed systems. A distributed system 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 distributedsystem 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 thefollowing

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.

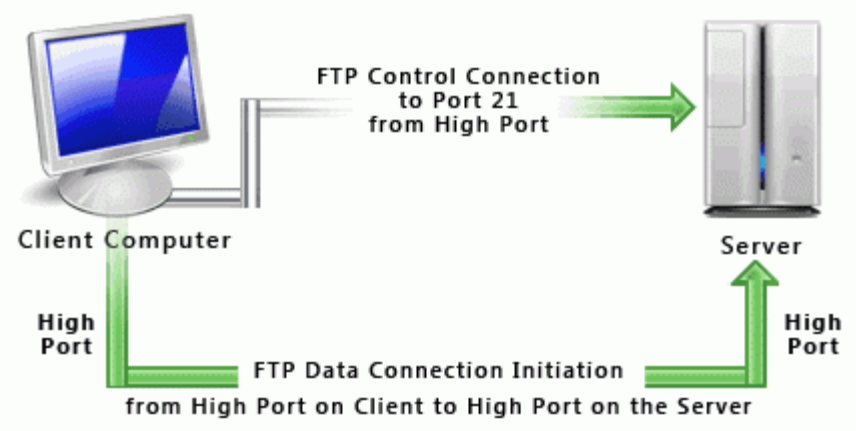
Experiment-1: Implementation FTP Client:

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

Description: The File Transfer Protocol (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 stubtry
    {

        /* 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)
                     {
```

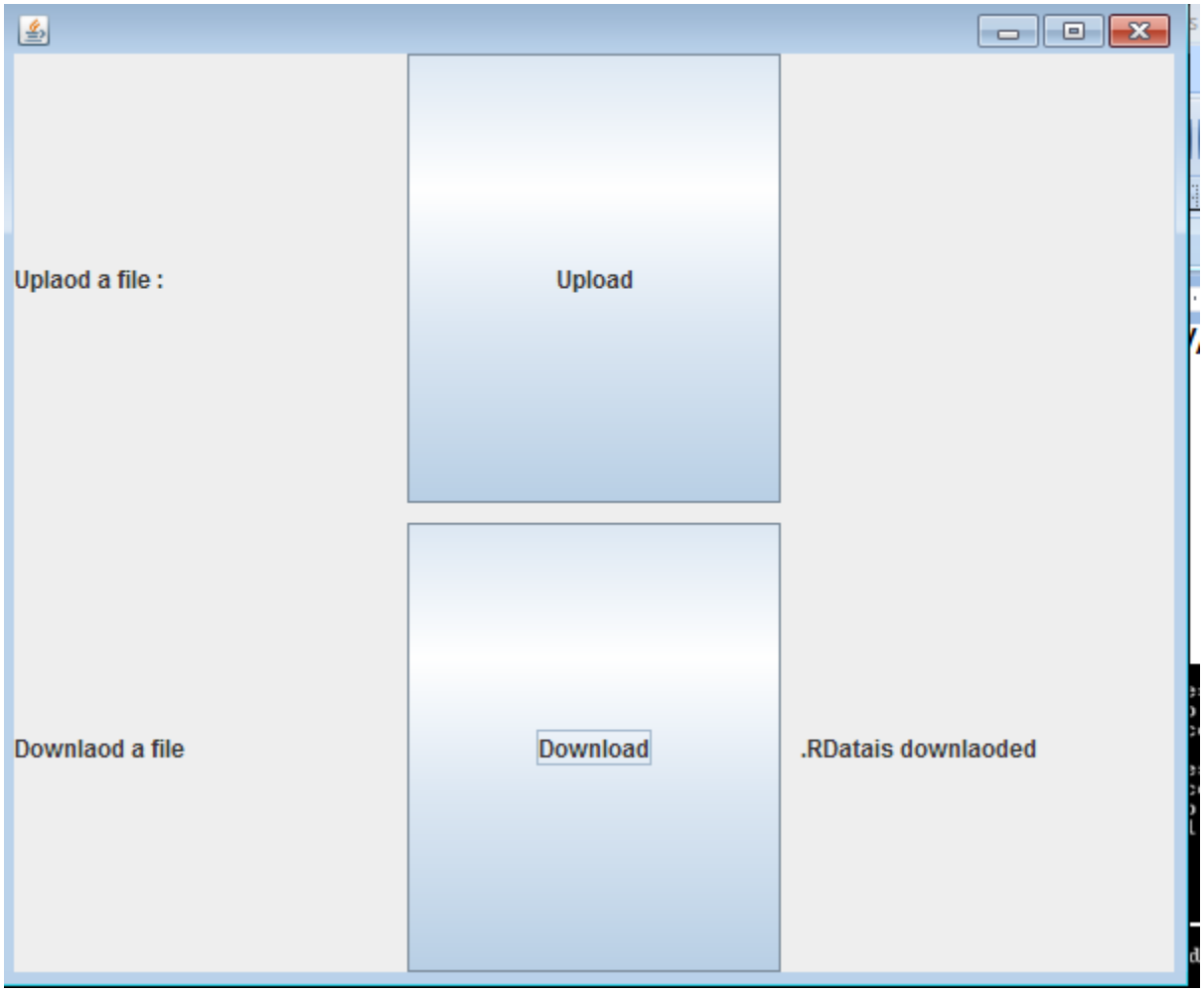
```
                    fout.write((char) ch);
                }
                fout.close();//s.close(); lmsg2.setText(filename+"is
                downlaoded");repaint();
            }
        }
catch (Exception ee)
{
        // TODO: handle exception
        System.out.println(ee);
        }
    }

}
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
          }
        }
}
```

Output:



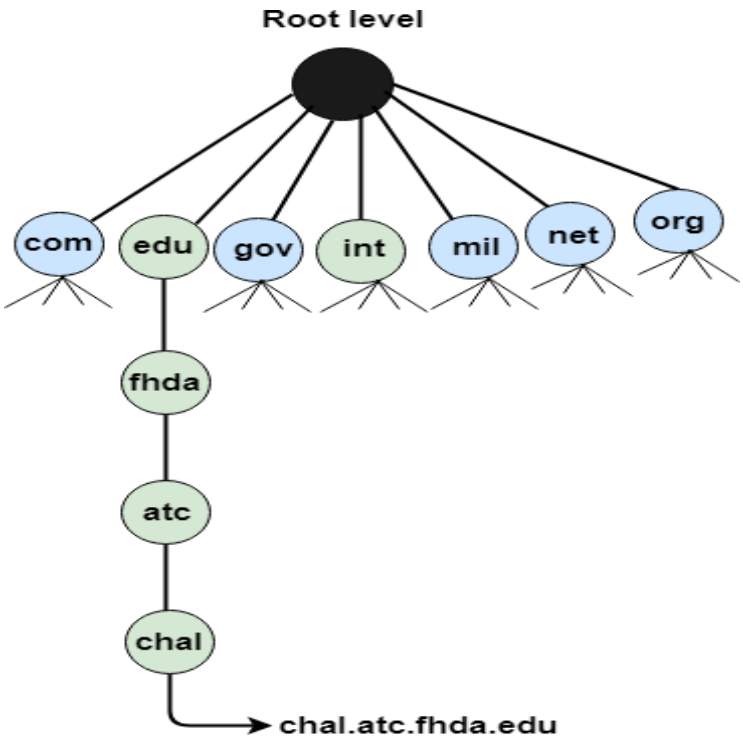

```
C:\Users\LAB4-57\Desktop>java FTPClient
Client connected to server
java.net.ConnectException: Connection refused: connect

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
```

Experiment-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.



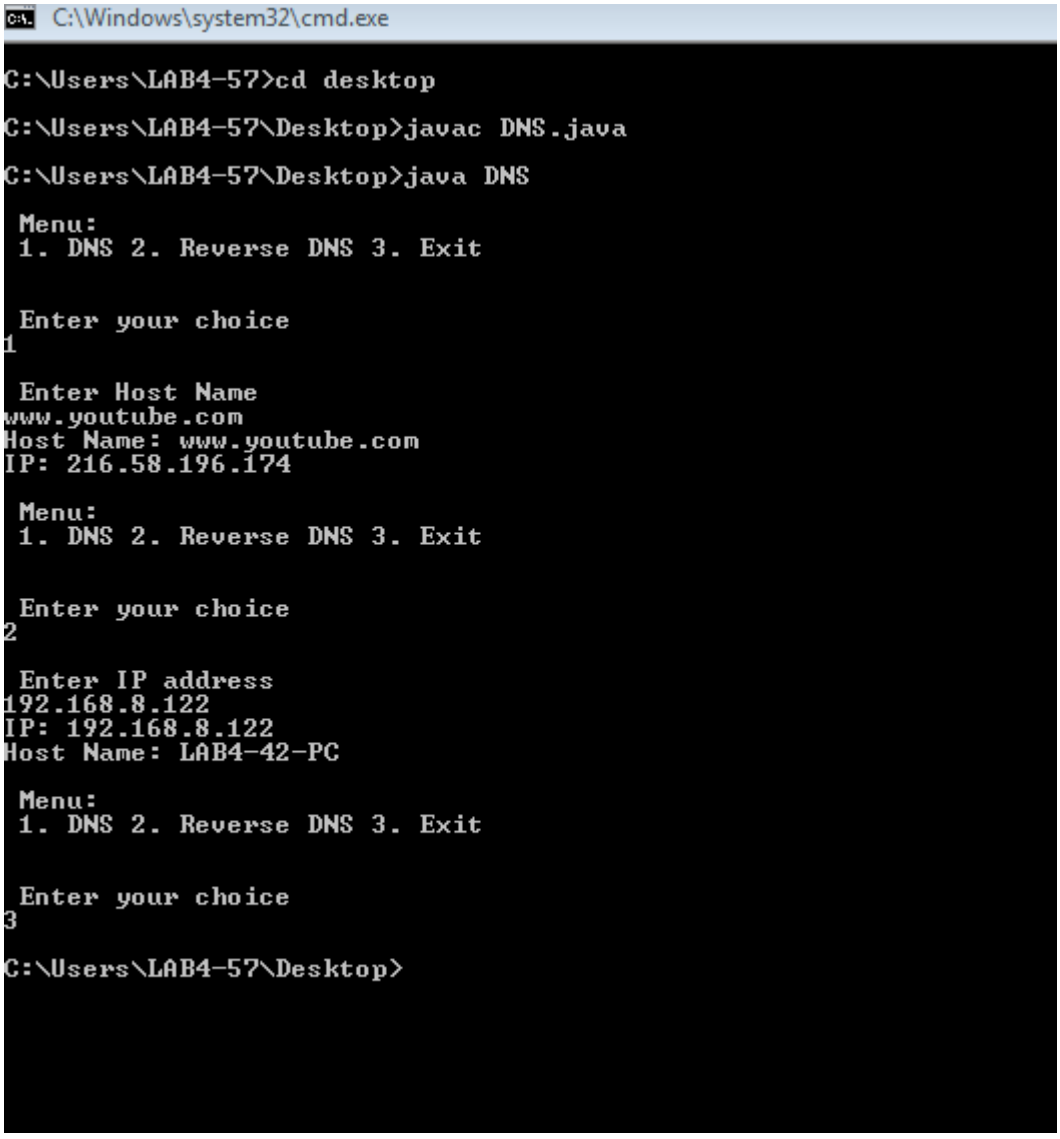
Program:

```
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));
}
}
```

Output:


C:\Windows\system32\cmd.exe
C:\Users\LAB4-57>cd desktop
C:\Users\LAB4-57\Desktop>javac DNS.java
C:\Users\LAB4-57\Desktop>java DNS
Menu:
1. DNS 2. Reverse DNS 3. Exit
Enter your choice
1
Enter Host Name
www.youtube.com
Host Name: www.youtube.com
IP: 216.58.196.174
Menu:
1. DNS 2. Reverse DNS 3. Exit
Enter your choice
2
Enter IP address
192.168.8.122
IP: 192.168.8.122
Host Name: LAB4-42-PC
Menu:
1. DNS 2. Reverse DNS 3. Exit
Enter your choice
3
C:\Users\LAB4-57\Desktop>

Experiment-3: Implementation of Chat Server

Aim: To develop a client server application this implements Chat Server. Let theclient 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 allclients. 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);
```

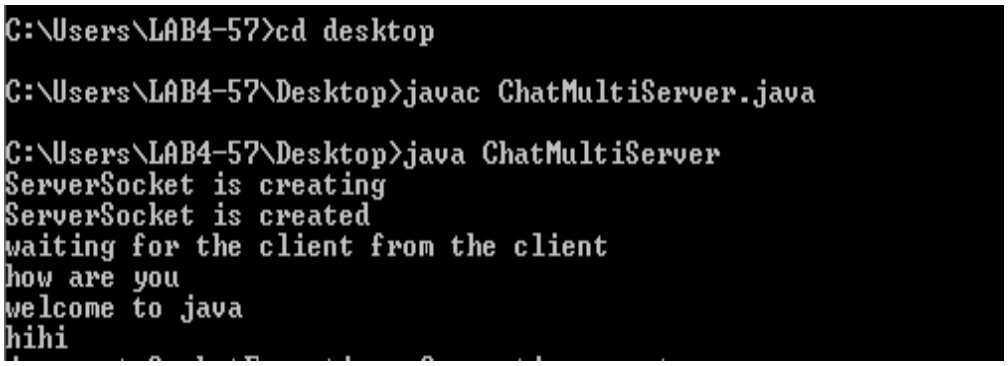
```
  }

  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)
    {
    }
  }
}
```

```
C:\Users\LAB4-57\Desktop>javac CCLogin.java

C:\Users\LAB4-57\Desktop>java CCLogin
connecting to server
client1 connected to server
Hi      Prashanth u can start chating
```

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);
     }
  }
}
```


C:\Users\LAB4-57>cd desktop
C:\Users\LAB4-57\Desktop>javac ChatMultiServer.java
C:\Users\LAB4-57\Desktop>java ChatMultiServer
ServerSocket is creating
ServerSocket is created
waiting for the client from the client
how are you
welcome to java
hihi

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);


          }
       }
}
```

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

C:\Users\LAB4-57\Desktop>javac Client1.java

C:\Users\LAB4-57\Desktop>java Client1
connecting to server
client1 connected to server
Hi       u can start chating
how are you
welcome to java
hihi
```

Experiment-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/abccd

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)

Experiment-5: Write a program to implement Hello world service using RPC.

Description:

A remote procedure call is an inter-process communication technique that is used forclient-server based applications. It is also known as a subroutine call or a function call.

A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sendsthe required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

The sequence of events in a remote procedure call is given as follows −

- The client stub is called by the client.
- The client stub makes a system call to send the message to the server and puts the parameters in the message.
- The message is sent from the client to the server by the client's operating system.
- The message is passed to the server stub by the server operating system.
- The parameters are removed from the message by the server stub.
- Then, the server procedure is called by the server stub.

Now let's see how we can implement a simple hello world java RPC program.

First we want to create 4 files. Three of them are for server side and the other one is for clientside. Luckily it requires no extra configuration settings. So you don't need to load any extra jar file forit. Just import some libraries which already inbuilt in JDK.

- *HelloWorld.java*
- *HelloWorldImpl.java*

Server side:

Then you have to create a class that actually implements the above interface, which will be your Endpoint implementation.

- *Publisher.java*

Finally you create your Endpoint publisher which actually deploys the web service and creates and publishes the endpoint for the specified implementer object at a given address. The necessary server infrastructure will be created and configured by the JAX-WS implementation. You have to run the publisher to make your Web Service available to clients.

Client Side

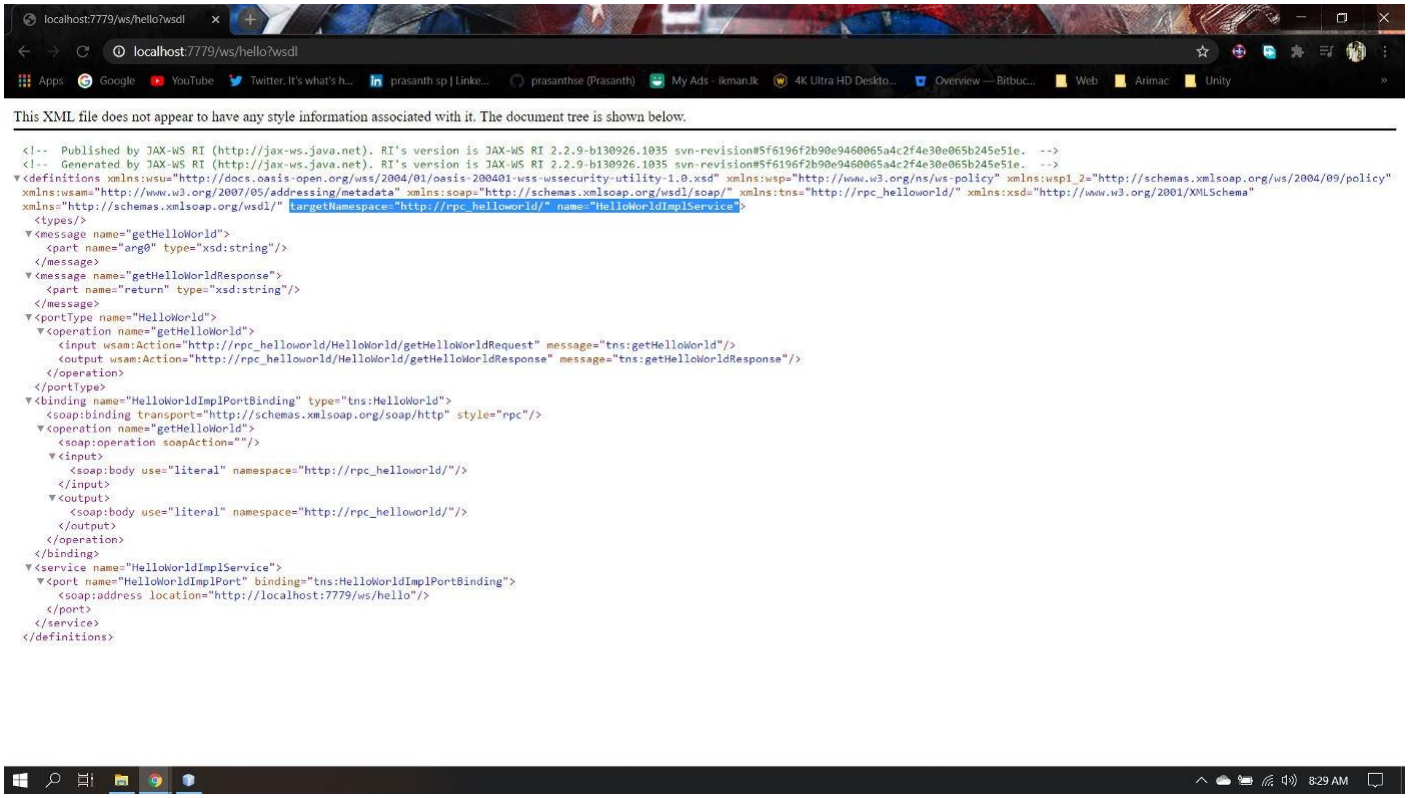
- *HelloWorldClient.java*

This is client that consumes the above Web Service.

After create all these files, you need to run your Publisher.Java and then go to your browser andtype the following:

- http://localhost:7779/ws/hello?wsdl

Then you will get the response in XML format. After that you need to copy the text that assign for **targetNamespace.** Here in my example the text is "**http://rpc_helloworld/**".



Then paste the text in your Client side file, as the QName first parameter (The image is shown above). Now run your program and you will get the output.

Experiment-6: Implement a word count application which counts the number ofoccurrances 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 resultsof 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:

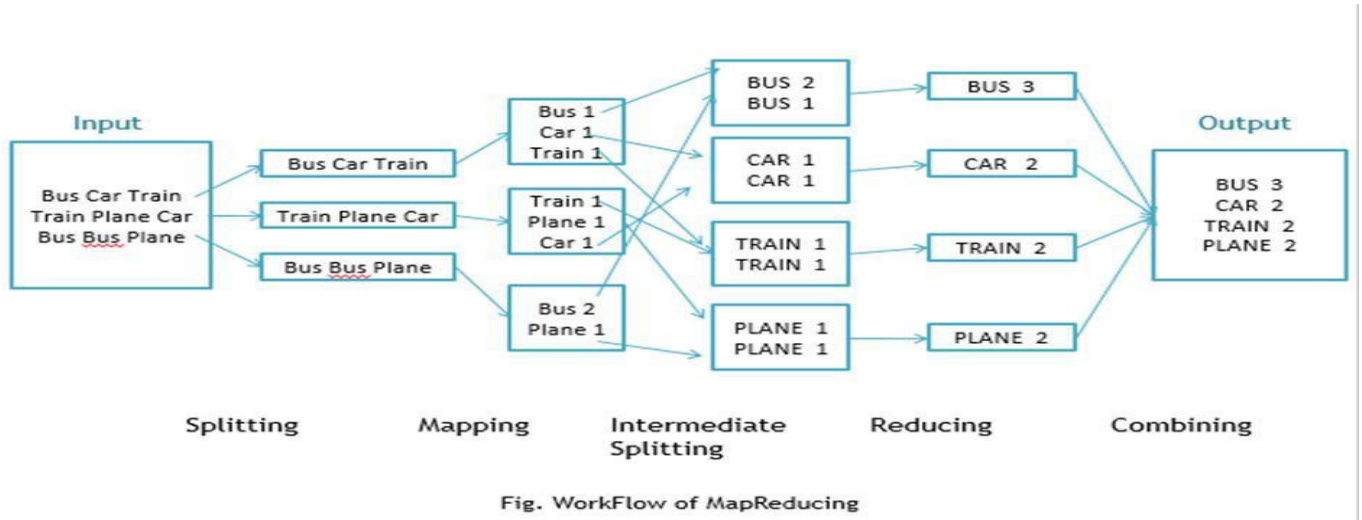


Fig. WorkFlow of MapReducing

Workflow of MapReduce consists of 5 steps:

1. **Splitting** – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
2. **Mapping** – as explained above.
3. **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.
4. **Reduce** – it is nothing but mostly group by phase.
5. **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. Therest 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***
5. Program:

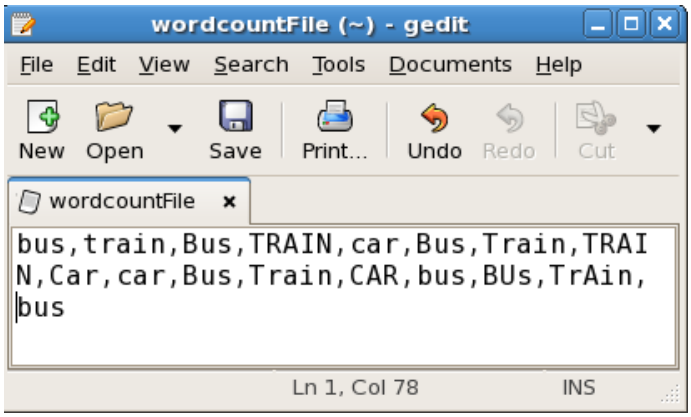
```
package PackageDemo;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.Mapper;import
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));
}
}
}
```

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:

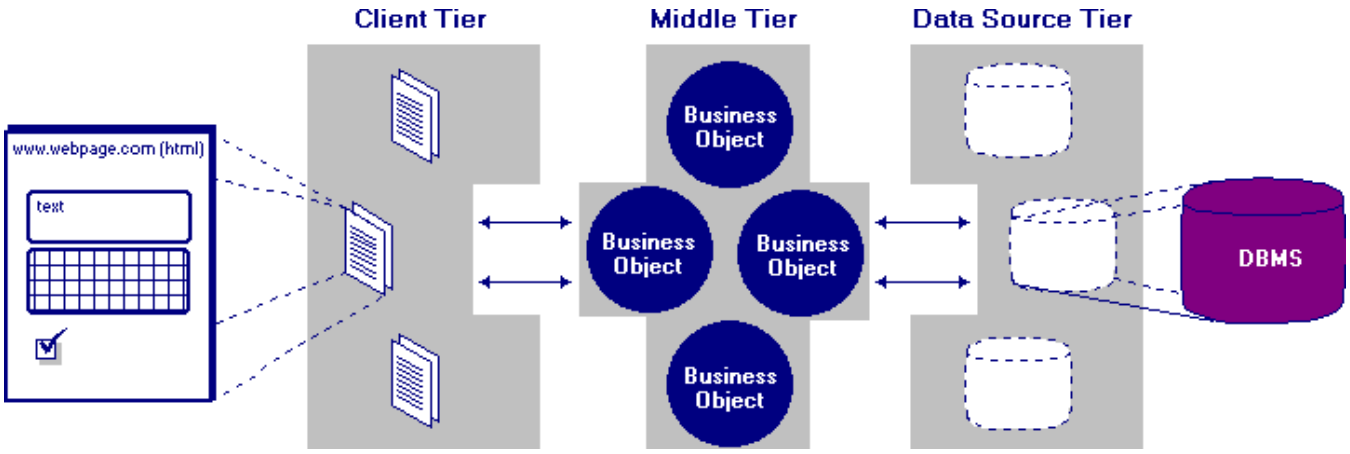
(Hadoop jar jarfilename.jar packageName.ClassName PathToInputTextFilePathToOutputDirectry)

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

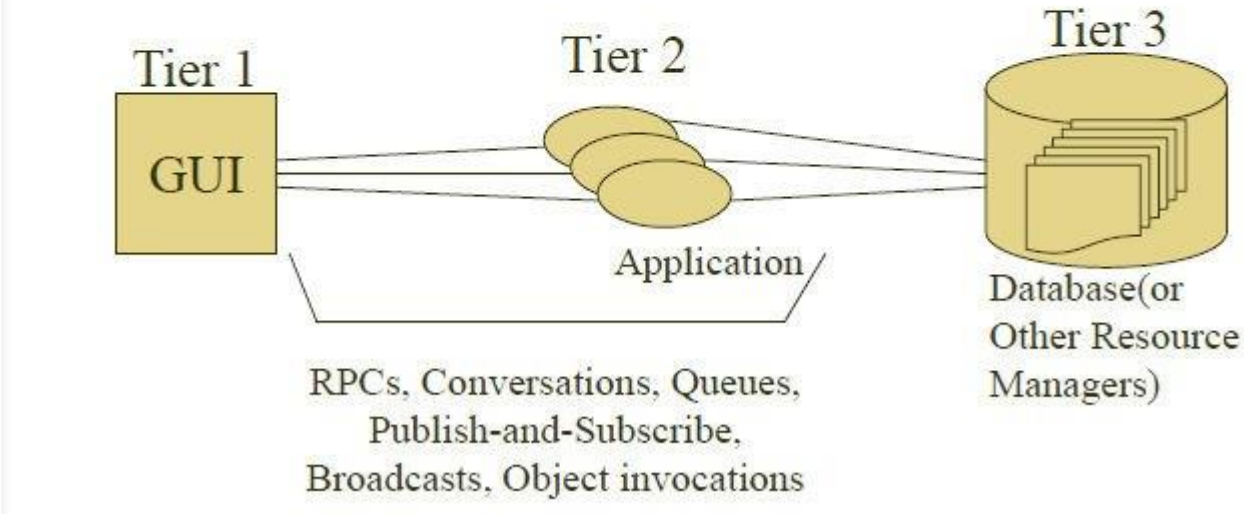
3. Open the result:

[training@localhost
~]$ hadoop fs -ls
MRDir1Found 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

Experiment-7: Develop an application using 3-tier architectures.



A 2-Tier Client-Server Application Architecture



A 3-Tier Client-Server Application Architecture

ADDITIONAL EXPERIMENTS

EXPERIMENT-1

Aim: Implementing Lamport Clock Implementation

Description

```
import java.net.InetAddress;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Arrays;

import java.util.List;

public class Main {

 public static void main(String[] args) {

   if (args.length == 0) {

        System.out.println("Usage: java Main (number of processses) [filename of commands]");

        return;

    }

            // TODO: add support for reading commands from a file

            String input;

            try {

        int n = Integer.parseInt(args[0]);

        LamportClock[] clocks = new LamportClock[n];
```

```
System.setProperty("java.net.preferIPv4Stack" , "true");

InetAddress group = InetAddress.getByName("224.255.255.255");

for (int i = 0; i < n; ++i) {

   int port = 8888;

   LamportClock lc = new LamportClock(group, port, i);

   lc.start();

   clocks[i] = lc;

}

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

while (true) {

   input = in.readLine();

   if (input.equals("exit"))

      return;



   /**

    * A message format is of the following:
```

```
 * EVENT_NAME ID_OF_SENDER (ID_OF_RECEIVER)

 *

 * EVENT_NAME is of the following:

 *  - SEND

 *  - LOCAL

 *

 * For example:

 * SEND 1 2 (process 1 sends a message to process 2)

 * LOCAL 3 (process 3 performs a local event)

 */

// perform a string split operation based on space

String[] splits = input.split(" ");

if (splits.length == 0) {

   continue;

}
```

```
switch(splits[0].toUpperCase()) {

        case "SEND":

    int clockArrayId = Integer.parseInt(splits[1]);

    long firstProcessId = clocks[clockArrayId].getId();

    long secondProcessId = clocks[Integer.parseInt(splits[2])].getId();

    String messageContent = "";

    if (splits.length >= 3) {

      List<String> wordsList = Arrays.asList(

        Arrays.copyOfRange(splits, 3, splits.length));

      messageContent = String.join(" ", wordsList);

    }

            Event e = new Event(1, firstProcessId, secondProcessId,
            messageContent);

    clocks[clockArrayId].updateTime(e);

    break;



  case "LOCAL":
```

```
    clockArrayId = Integer.parseInt(splits[1]);

    firstProcessId = clocks[clockArrayId].getId();

    secondProcessId = 0;

    messageContent = "";

    e = new Event(0, firstProcessId, secondProcessId, messageContent);

    clocks[clockArrayId].updateTime(e);

    break;

          case "REQUEST":

    clockArrayId = Integer.parseInt(splits[1]);

    firstProcessId = clocks[clockArrayId].getId();

    e = new Event(3, firstProcessId, -1, "");

    clocks[clockArrayId].updateTime(e);

    break;

  default:

    throw new RuntimeException("Invalid event name");
```

```
            }

        }

    } catch(Exception e) {

        System.err.println(e);

        return;

    }

}

        }
```

EXPERIMENT-2

Aim: Write a program using CORBA to demonstrate object brokering.Procedure:

Step no.	Details of the step
1	Define the IDL interface
2	Implement the IDL interface using idlj compiler
3	Create a Client Program
4	Create a Server Program
5	Start orbed.
6	Start the Server.
7	Start the client

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?

***