2) Nameserver

```
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().readLin
 e());
  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));
 }
}
```

<u>5) RPC- helloworld</u>

Server

```
Package rpc_helloworld;

Import javax.xml.ws.Endpoint;

Public class Publisher {

Public static void main(String[] args){

Endpoint.publish("http://localhost:7779/ws/hello",new HelloWorld Imp 1());

} }
```

Client

```
Package rpc_helloworld;

Import java.net.Malformed URL Exception;

Import java.net.URL;

Import java.util.logging.Level;

Import java.util.logging.Logger;

Import javax.xml.namespace.QName;

Import javax.xml.ws.Service;

Pubic class RPC_HelloWorld {

Public static void main(String[] args){

try {

URL url = new URL("http://localhost:7779/ws/hello?wsdl");

QName qname= new QName("http://rpc_helloworld/","HelloWorld Imp 1 Service");

Service service=Sevice.create(url,qname);

HelloWorld hello= service.getFort(HelloWorld.class);

System.out.println(hello.getHelloWorld("Hello World !"));

}

Catch(Malformed URL Exception ex) {

System.out.println("WSDL document url error:"+ex);

}

}

}
```

Description: A remote procedure call is an inter-process communication technique that is used for client-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 sends the 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.

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);
 } } }
```

3) Chat Server

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)

 {

 } } }
```

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);
 }
 }
}
```

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);

 }

 }

}
```

Lamport Clock Output:

```
      e21  e22  e23
 e11   0    0    0
 e12   0    0    1
 e13   0    0    0
 e14   0    0    0
 e15   0   -1    0
The time stamps of
events in P1:
1 2 3 4 5
The time stamps of
events in P2:
1 2 3
```

Lamport clock

```
// Java program to illustrate the Lamport's
// Logical Clock
import java.util.*;
public class GFG {

  // Function to find the maximum
timestamp
  // between 2 events
  static int max1(int a, int b)
  {
    // Return the greatest of the two
    if (a > b)
      return a;
    else
      return b;
  }

  // Function to display the logical
timestamp
  static void display(int e1, int e2, int p1[],
int p2[])
  {
    int i;
    System.out.print(
      "\nThe time stamps of events in
P1:\n");

    for (i = 0; i < e1; i++) {
      System.out.print(p1[i] + " ");
    }

    System.out.println(
      "\nThe time stamps of events in P2:");

    // Print the array p2[]
    for (i = 0; i < e2; i++)
      System.out.print(p2[i] + " ");
  }

  // Function to find the timestamp of
events
  static void lamportLogicalClock(int e1, int
e2,
                                  int m[][])
  {
    int i, j, k;
    int p1[] = new int[e1];
    int p2[] = new int[e2];
    // Initialize p1[] and p2[]
    for (i = 0; i < e1; i++)
      p1[i] = i + 1;

    for (i = 0; i < e2; i++)
      p2[i] = i + 1;
    for (i = 0; i < e2; i++)
      System.out.print("\te2" + (i + 1));

    for (i = 0; i < e1; i++) {
System.out.print("\n e1" + (i + 1) + "\t");
  for (j = 0; j < e2; j++)
  System.out.print(m[i][j] + "\t");
  }

  for (i = 0; i < e1; i++) {
    for (j = 0; j < e2; j++) {

      // Change the timestamp if the
      // message is sent
      if (m[i][j] == 1) {
        p2[j] = max1(p2[j], p1[i] + 1);
        for (k = j + 1; k < e2; k++)
          p2[k] = p2[k - 1] + 1;
      }

      // Change the timestamp if the
      // message is received
      if (m[i][j] == -1) {
        p1[i] = max1(p1[i], p2[j] + 1);
        for (k = i + 1; k < e1; k++)
          p1[k] = p1[k - 1] + 1;
      }  }  }

  // Function Call
display(e1, e2, p1, p2);
}

public static void main(String args[])
{
  int e1 = 5, e2 = 3;
  int m[][] = new int[5][3];
  // message is sent and received
  // between two process

  /*dep[i][j] = 1, if message is sent
                  from ei to ej
    dep[i][j] = -1, if message is received
                   by ei from ej
    dep[i][j] = 0, otherwise*/
  m[0][0] = 0;
  m[0][1] = 0;
  m[0][2] = 0;
  m[1][0] = 0;
  m[1][1] = 0;
  m[1][2] = 1;
  m[2][0] = 0;
  m[2][1] = 0;
  m[2][2] = 0;
  m[3][0] = 0;
  m[3][1] = 0;
  m[3][2] = 0;
  m[4][0] = 0;
  m[4][1] = -1;
  m[4][2] = 0;

  // Function Call
  lamportLogicalClock(e1, e2, m);
} }
```

4) Working of NFS

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)

Word Count program 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:

```
[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
```

6) Word Count Application

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));
} } }
```

Berkley Alogirthm

```
# Python3 program imitating a clock server
from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time

# datastructure used to store client address
and clock data
client_data = {}

''' nested thread function used to receive
        clock time from a connected client '''
def startReceivingClockTime(connector,
address):
        while True:
                # receive clock time
                clock_time_string =
connector.recv(1024).decode()
                clock_time =
parser.parse(clock_time_string)
                clock_time_diff =
datetime.datetime.now() - \

        clock_time

                client_data[address] = {

        "clock_time"     : clock_time,

        "time_difference" : clock_time_diff,

        "connector"      : connector
                                                }

                print("Client Data updated
with: "+ str(address),


        end = "\n\n")
                time.sleep(5)


''' master thread function used to open
portal for
        accepting clients over given port '''
def startConnecting(master_server):

        # fetch clock time at slaves / clients
        while True:
                # accepting a client / slave
clock client
                master_slave_connector,
addr = master_server.accept()
                slave_address = str(addr[0])
+ ":" + str(addr[1])
print(slave_address + " got connected
successfully")

current_thread = threading.Thread(
target = startReceivingClockTime,
args =
(master_slave_connector,slave_address, ))

current_thread.start()

# subroutine function used to fetch average
clock difference
def getAverageClockDiff():

        current_client_data =
client_data.copy()

        time_difference_list =
list(client['time_difference']

for client_addr, client in client_data.items())

sum_of_clock_difference =
sum(time_difference_list, \
datetime.timedelta(0, 0))
average_clock_difference =
sum_of_clock_difference \
                                                /
len(client_data)

        return average_clock_difference
''' master sync thread function used to
generate cycles of clock synchronization in
the network '''

def synchronizeAllClocks():

while True:

        print("New synchronization cycle
started.")
        print("Number of clients to be
synchronized: " + \

        str(len(client_data)))

        if len(client_data) > 0:

                average_clock_difference =
getAverageClockDiff()

for client_addr, client in client_data.items():
try:

        synchronized_time = \

        datetime.datetime.now() + \

        average_clock_difference
        client['connector'].send(str(

        synchronized_time).encode())

except Exception as e:

        print("Something went wrong while " 
+ \

        "sending synchronized time " + \

        "through " + str(client_addr))

                else :
                        print("No client data." 
+ \

        " Synchronization not applicable.")

                print("\n\n")

                time.sleep(5)

# function used to initiate the Clock Server / 
Master Node
def initiateClockServer(port = 8080):

        master_server = socket.socket()
        master_server.setsockopt(socket.S
OL_SOCKET,

        socket.SO_REUSEADDR, 1)

print("Socket at master node created 
successfully\n")

master_server.bind(('', port))

        # Start listening to requests
        master_server.listen(10)
        print("Clock server started...\n")

        # start making connections
        print("Starting to make 
connections...\n")
        master_thread = threading.Thread(

        target = startConnecting,

        args = (master_server, ))
        master_thread.start()

        # start synchronization
        print("Starting synchronization 
parallelly...\n")
        sync_thread = threading.Thread(

        target = synchronizeAllClocks,

        args = ())
        sync_thread.start()


# Driver function
if __name__ == '__main__':

        # Trigger the Clock Server
        initiateClockServer(port = 8080)
```

```
Output

New synchronization
cycle started.

Number of clients to be
synchronized: 3


Client Data updated
with: 127.0.0.1:57284


Client Data updated
with: 127.0.0.1:57274


Client Data updated
with: 127.0.0.1:57272
```

Lamport:
[Lamport's logical clock - GeeksforGeeks]

Berkley:
[Berkeley's Algorithm - GeeksforGeeks]