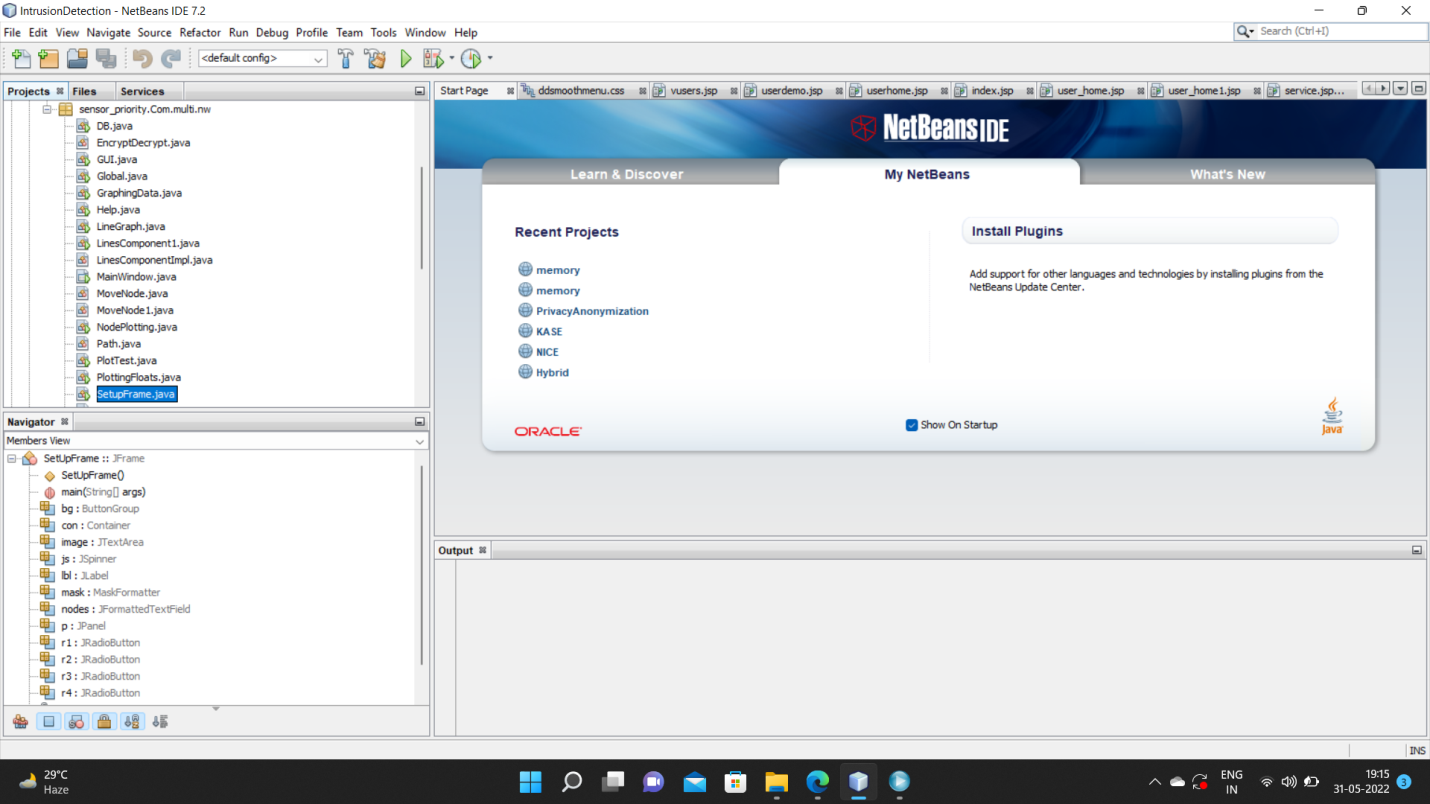
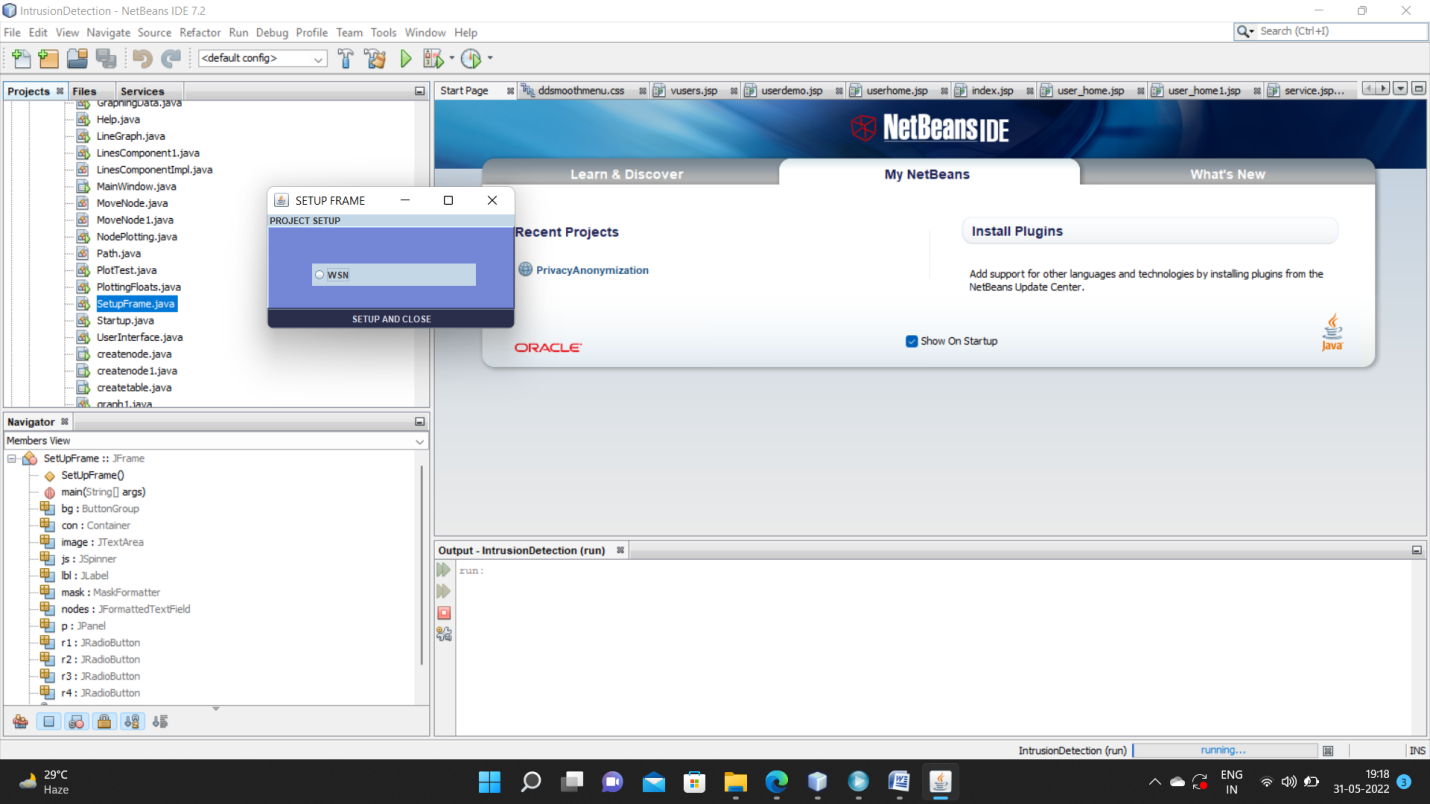
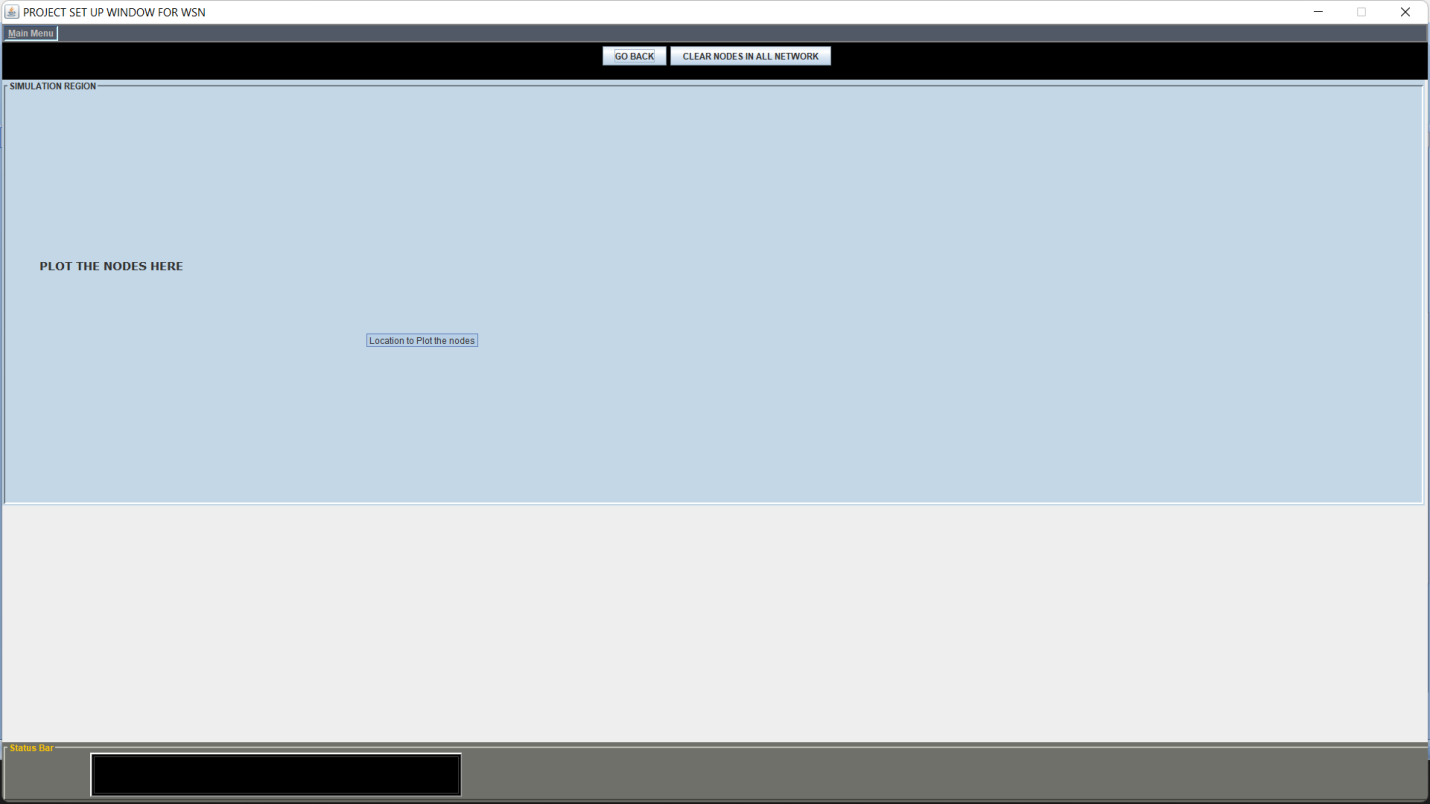
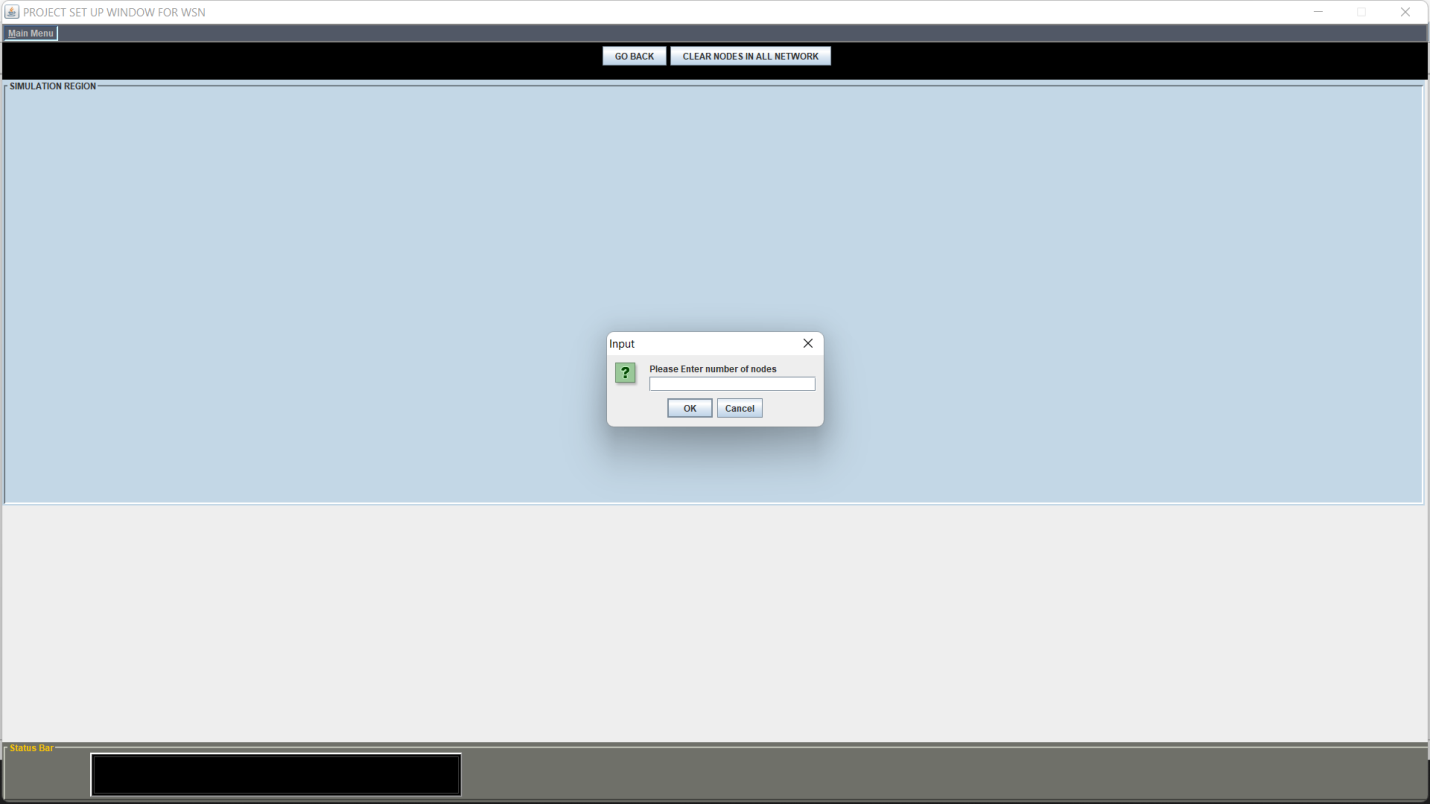
Intrusion Detection:

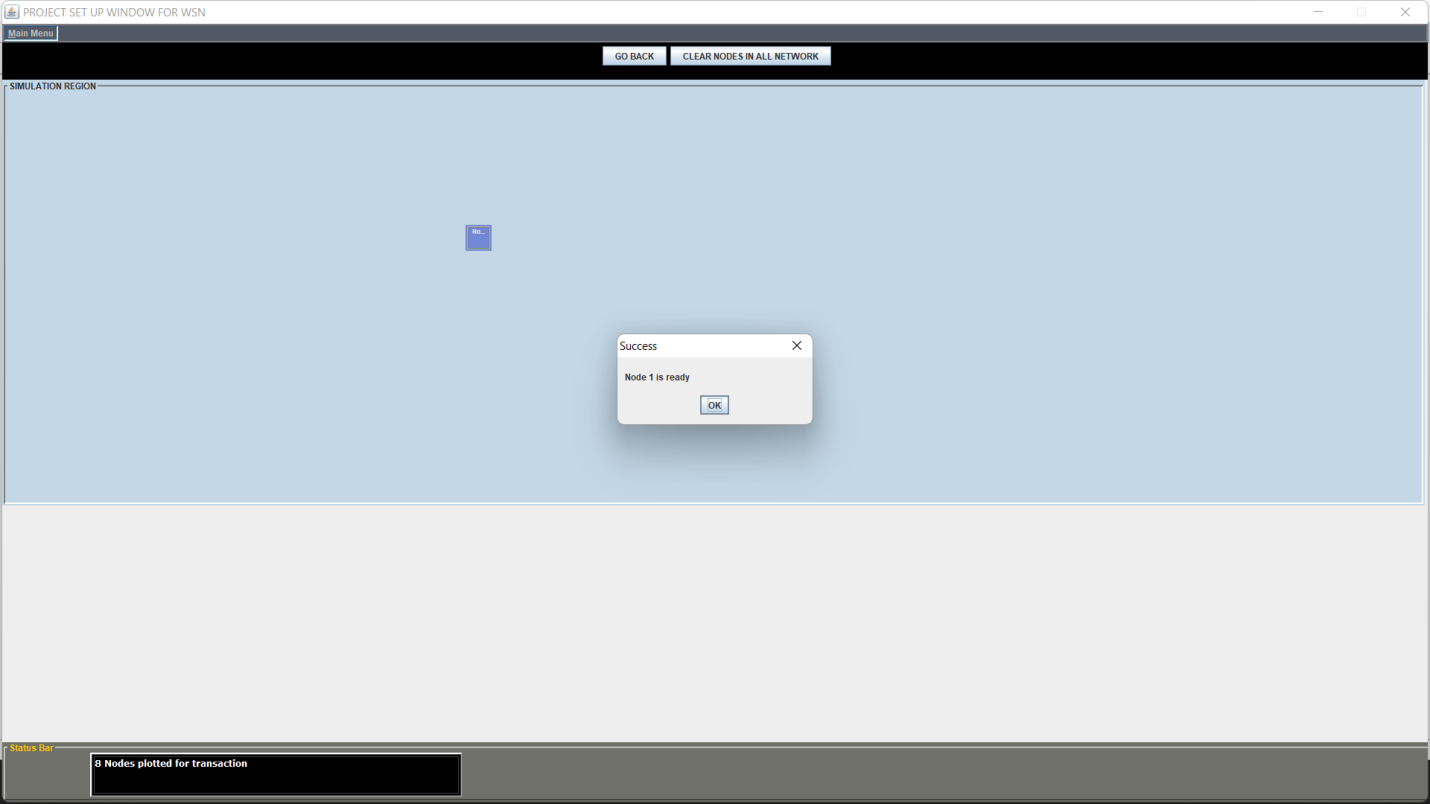


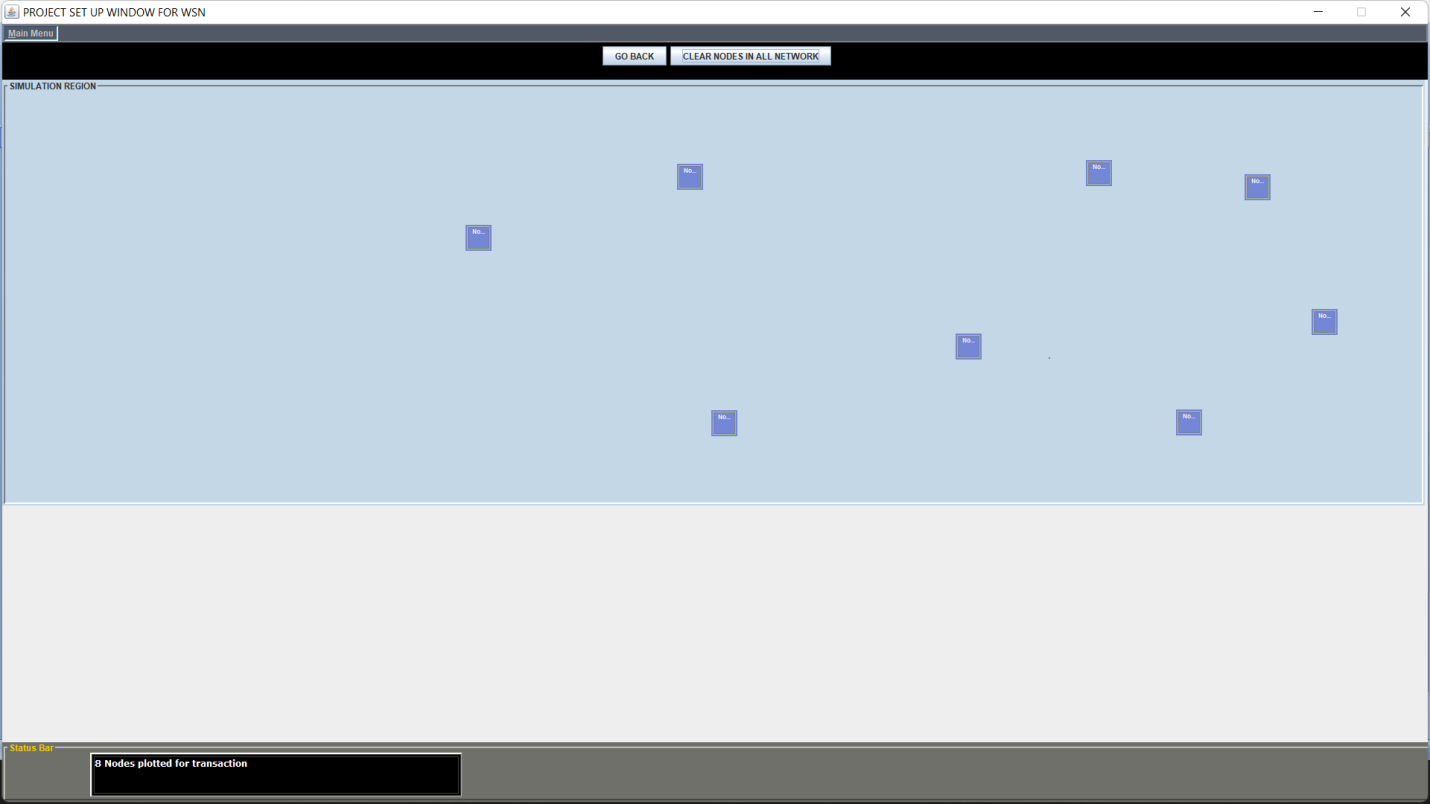


**Home Pages:**

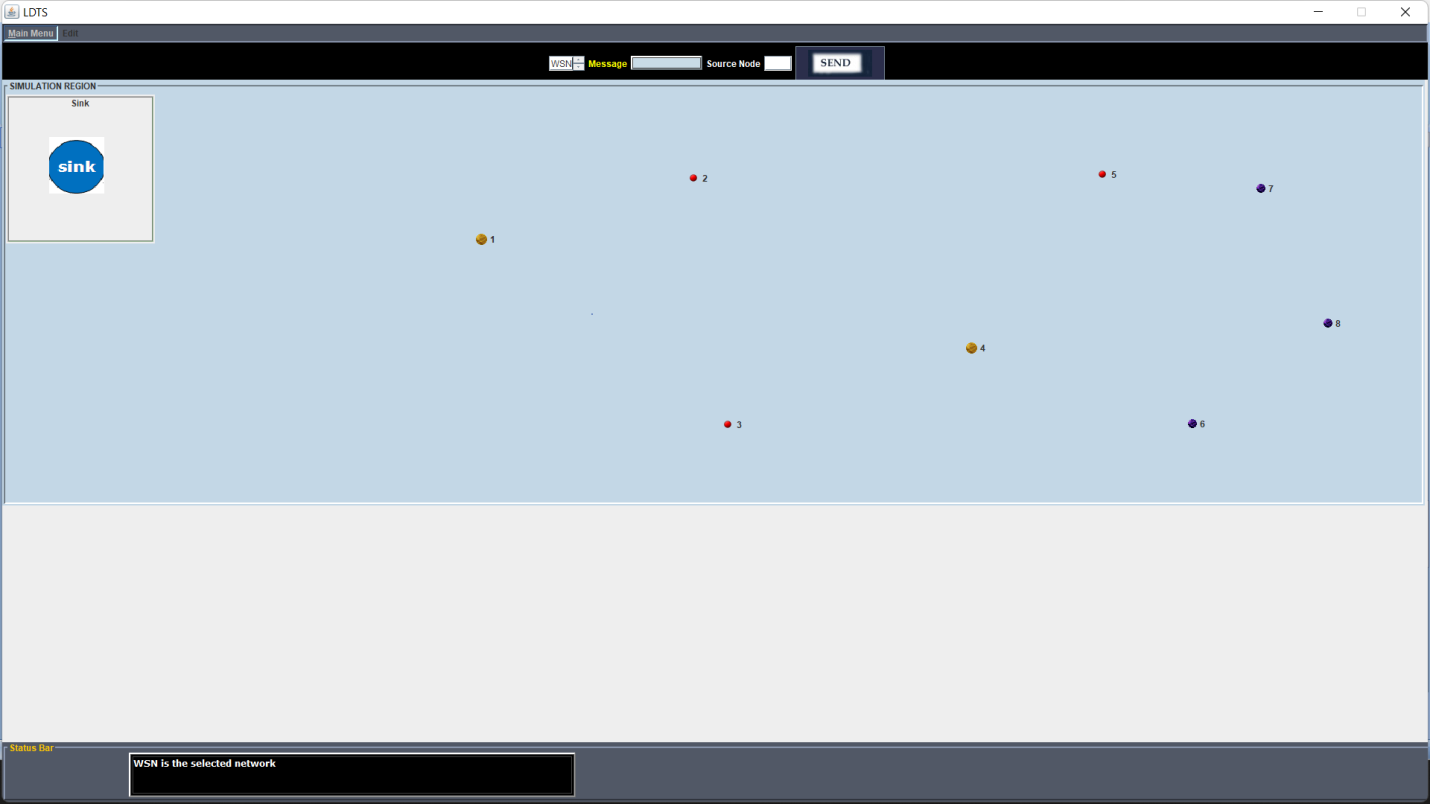


Node Creation:

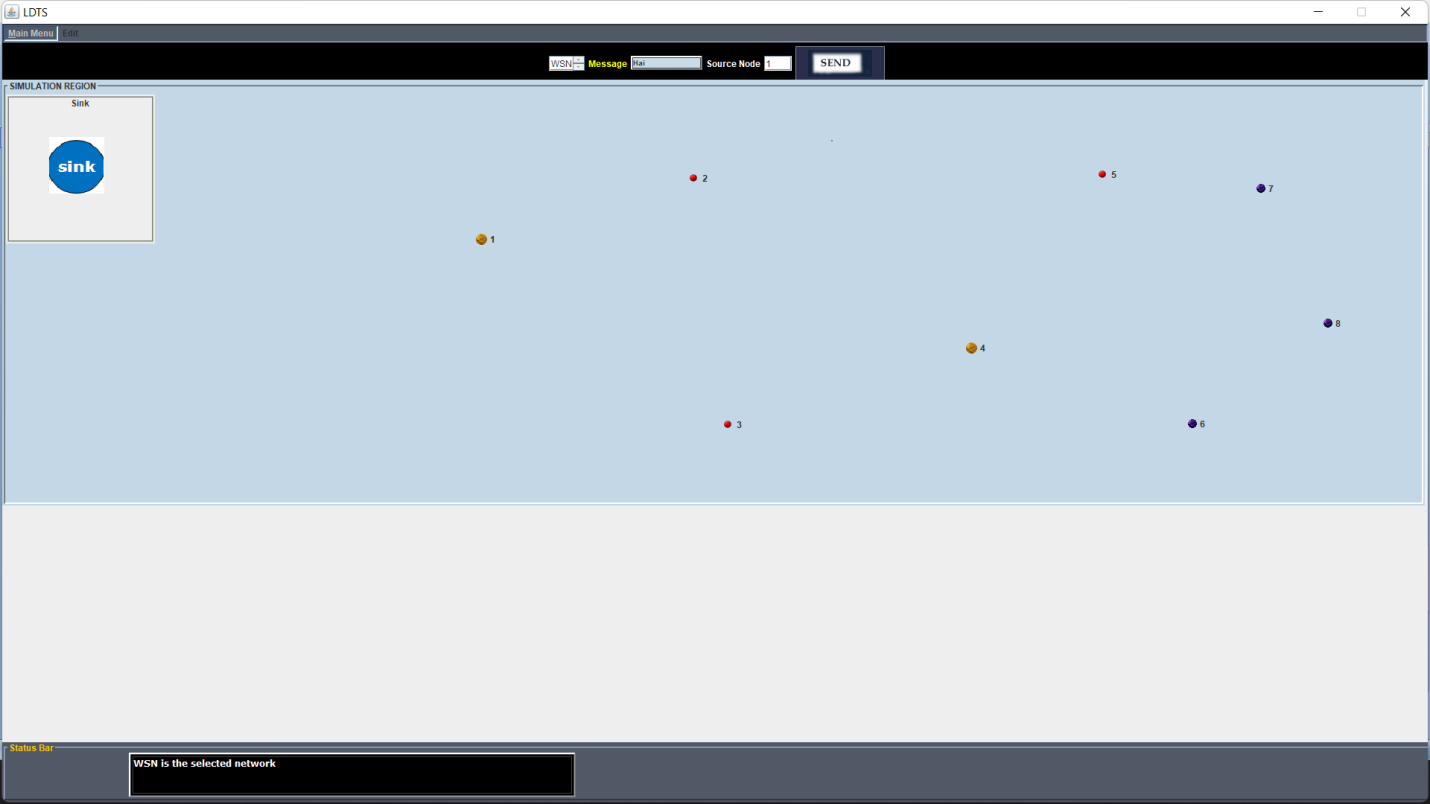




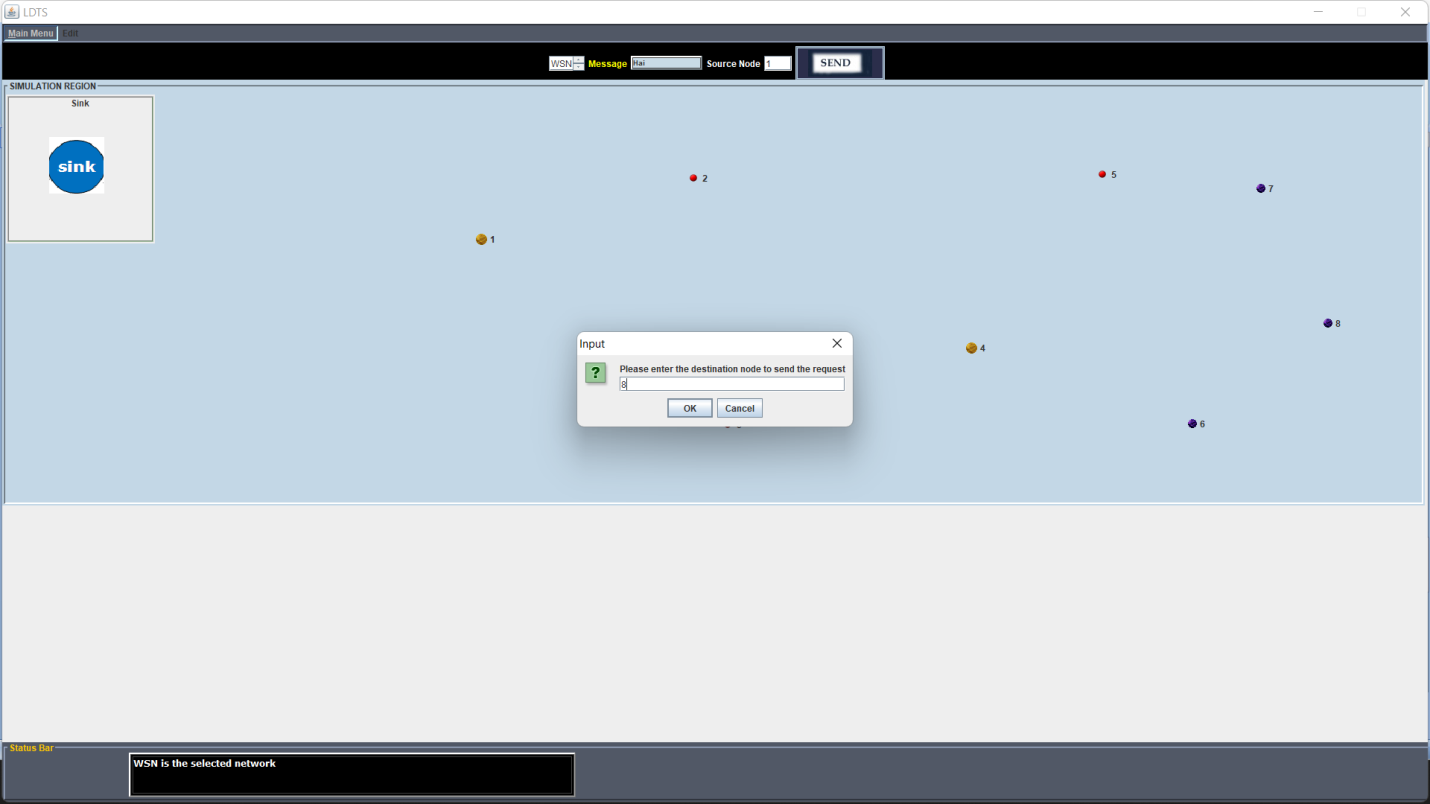
Node creation:

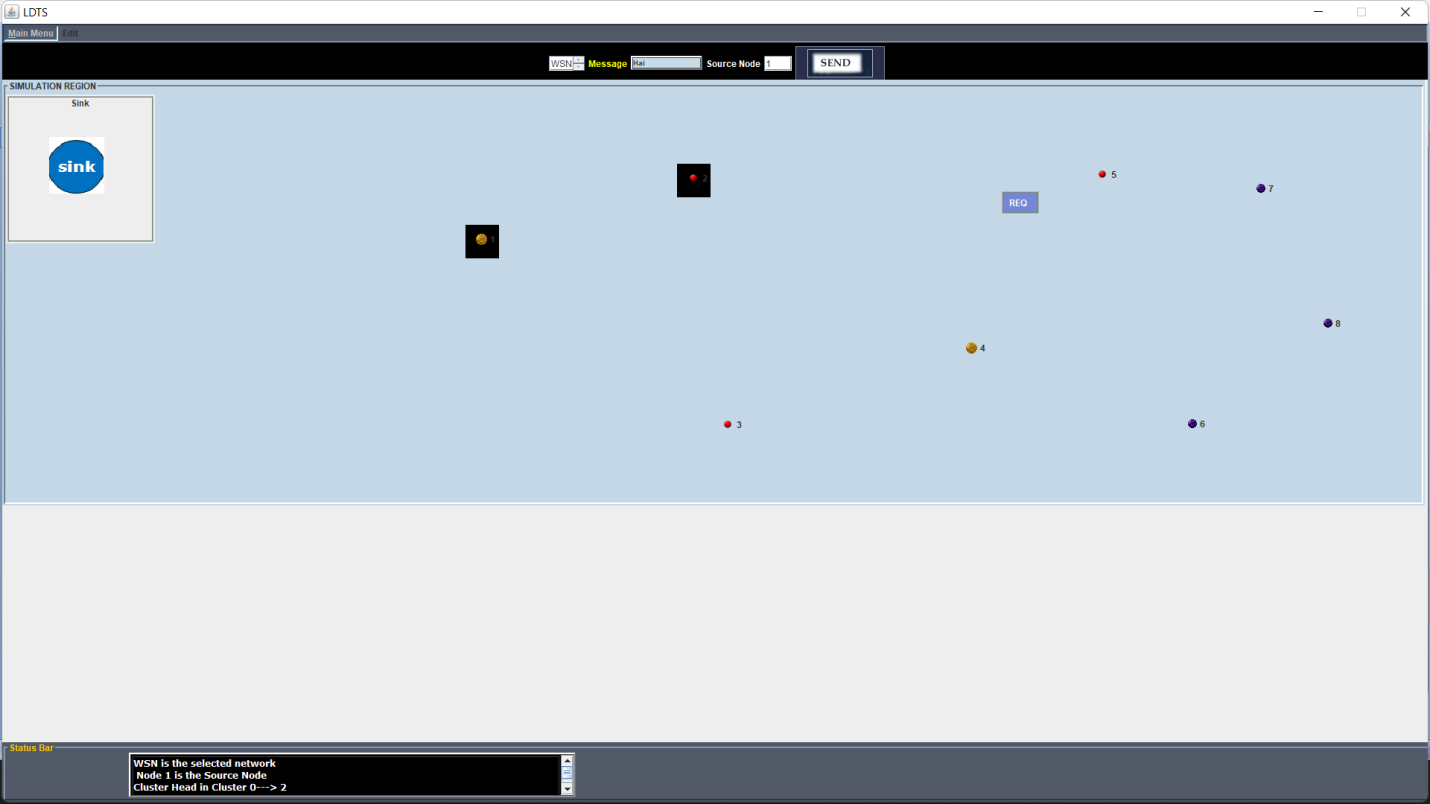


Data Transfer:

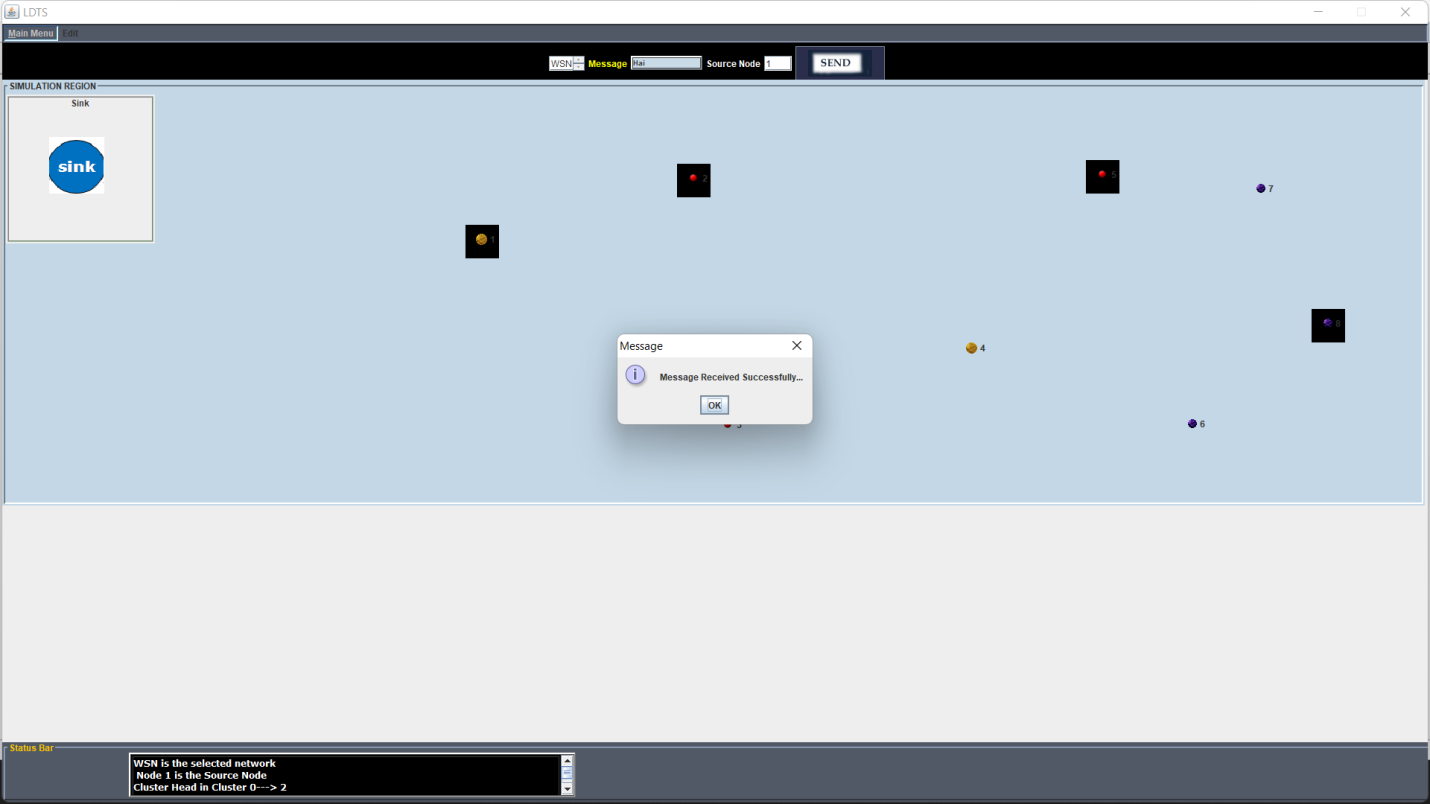


Data Transfer:

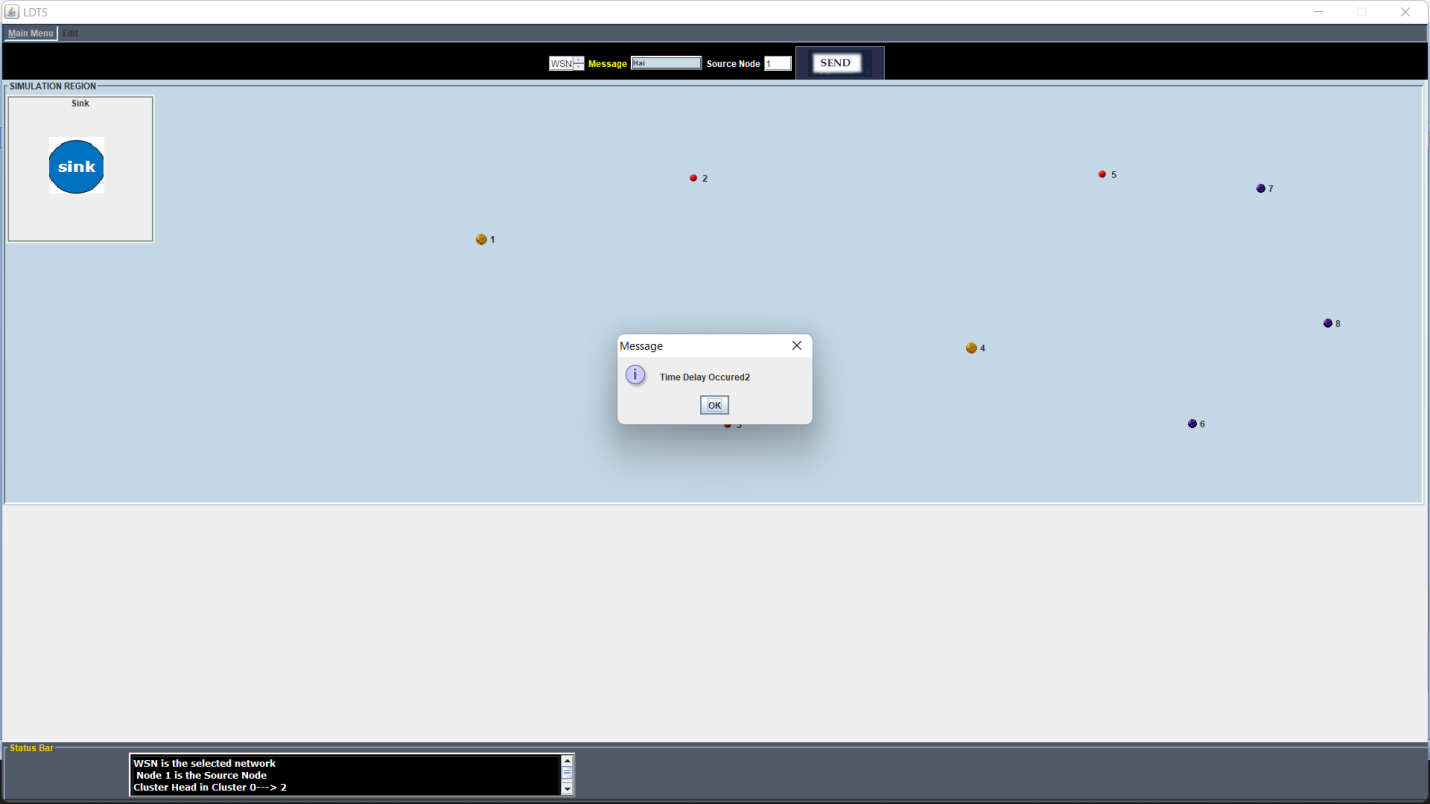




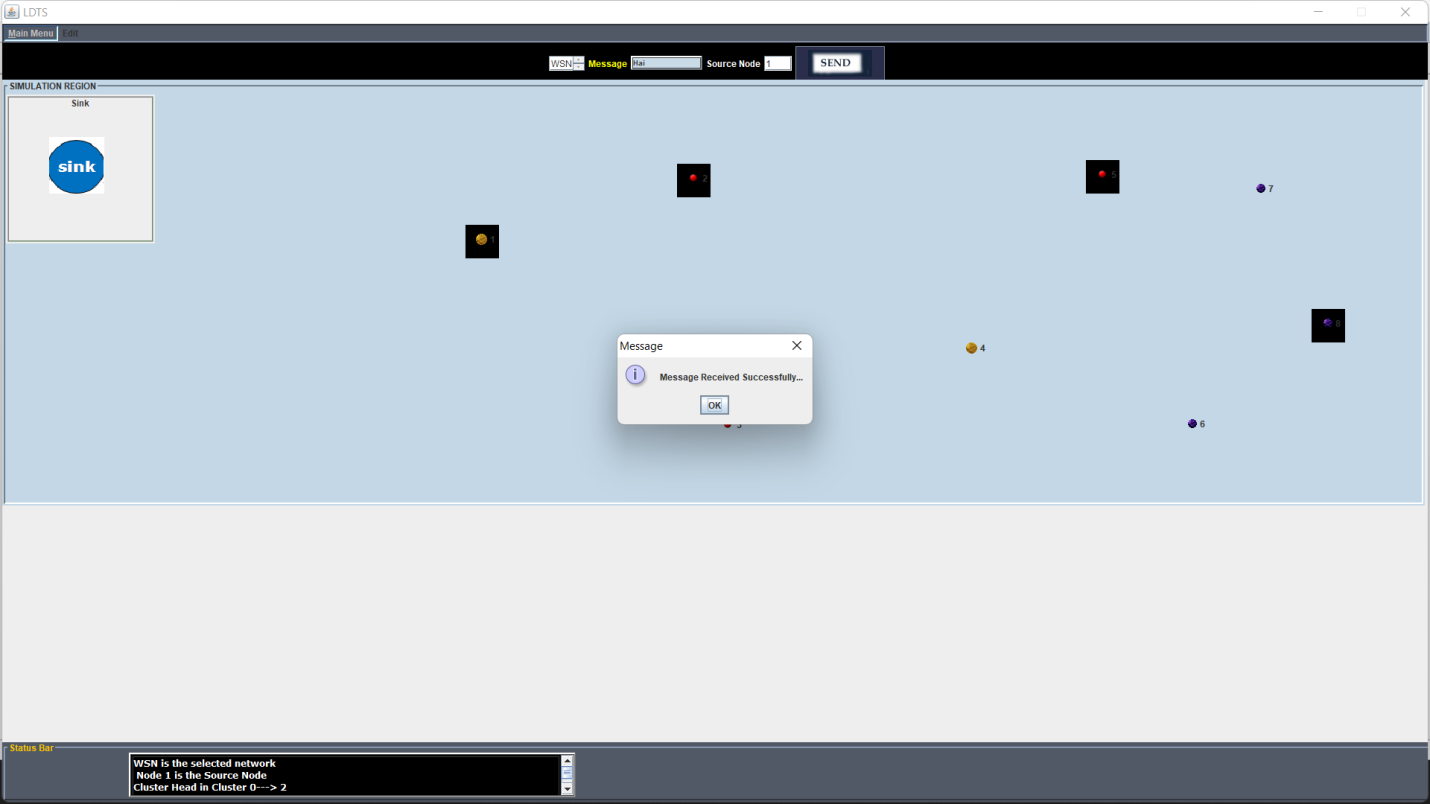
Message Received:



Intrusion Detection (Attacker Identified)



Data Received:



Sample Code:

Sensor code:

package sensor\_priority.Com.multi.nw;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.text.\*;

class SetUpFrame extends JFrame

{

Container con;

JPanel p;

JLabel title;

JTextArea image;

JRadioButton r1,r2,r3,r4;

ButtonGroup bg;

JSpinner js;

MaskFormatter mask;

JFormattedTextField nodes;

JLabel lbl;

private static final long serialVersionUID = 3l;

public SetUpFrame()throws Exception

{

// this.setUndecorated(true);

con=getContentPane();

con.setBackground(Global.backgroundColour);

con.setLayout(new BorderLayout());

JLabel lbl = new JLabel(" PROJECT SETUP");

con.add(lbl,BorderLayout.NORTH);

p=new JPanel();

p.setLayout(null);

p.setBorder(BorderFactory.createRaisedBevelBorder());

p.setBackground(Global.tempNodesColour);

con.add(p,BorderLayout.CENTER);

r1=new JRadioButton("INFRASTRUT");

r2=new JRadioButton("NODECENTRIC");

r3=new JRadioButton("SENSOR");

r4=new JRadioButton("WSN");

r1.setBounds(60,20,220,30);

r2.setBounds(60,80,220,30);

r3.setBounds(60,140,220,30);

r4.setBounds(60,50,220,30);

r1.setBackground(Global.backgroundColour);

r2.setBackground(Global.backgroundColour);

r3.setBackground(Global.backgroundColour);

r4.setBackground(Global.backgroundColour);

r1.setForeground(Color.DARK\_GRAY);

r2.setForeground(Color.DARK\_GRAY);

r3.setForeground(Color.DARK\_GRAY);

r4.setForeground(Color.DARK\_GRAY);

bg=new ButtonGroup();

bg.add(r1);

bg.add(r2);

bg.add(r3);

bg.add(r4);

// p.add(r1);

// p.add(r2);

// p.add(r3);

p.add(r4);

JButton b = new JButton("SETUP AND CLOSE");

GUI.button(b,"Close Me");

con.add(b,BorderLayout.SOUTH);

this.setResizable(true);

this.setTitle(" SETUP FRAME ");

b.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent ae)

{

if(r1.isSelected()==true){

new NodePlotting("net1",1,"INFRASTRUT");

dispose();

}

else if(r2.isSelected()==true){

new NodePlotting("net2",2,"NODECENTRIC");

dispose();

}

else if(r3.isSelected()==true){

new NodePlotting("net3",3,"SENSOR");

dispose();

}

else if(r4.isSelected()==true){

new NodePlotting("nodelog",4,"WSN");

dispose();

}

else

JOptionPane.showMessageDialog(null,"Please select one run option","Warning",JOptionPane.WARNING\_MESSAGE);

}

});

setDefaultCloseOperation(0);

setSize(350,200);

setVisible(true);

setLocation(350,250);

}

public static void main(String args[])throws Exception

{

new SetUpFrame();

}

}

Sender:

package sensor\_priority.Com.multi.nw;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.Connection;

import java.util.Random;

public class Startup extends JFrame

{

private static final long serialVersionUID = 3l;

public static JTextArea jta;

static JPanel panel,err[];

JPanel main;

JScrollPane jsp;

Container c;

JButton send,clear;

String serverIp;

static JCheckBox jcb;

String processor;

static JLabel lbl1,lbl2;

JMenuBar jmb;

JMenu menu;

JMenuItem upload,cls;

Connection con = DB.getConnection();

public Startup() throws Exception

{

super(" Intrusion Detection ");

Global.ACCESS\_KEY = new Random().nextInt(1000);

// DB.executeUpdate(con, "update basestation set accesskey="+Global.ACCESS\_KEY);

DB.initialize(con);

//UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

c = getContentPane();

c.setBackground(Color.darkGray);

c.setLayout(null);

panel= new JPanel();

panel.setBounds(10,10,320,290);

panel.setBackground(Color.LIGHT\_GRAY);

panel.setLayout(null);

c.add(panel);

jta = new JTextArea();

jta.setForeground(new Color(103,119,130));

jta.setLineWrap(true);

jta.setFont(new Font("Italic",Font.BOLD,14));

jta.setEditable(false);

jsp = new JScrollPane(jta);

jsp.setBorder(BorderFactory.createTitledBorder("------- Welcome Screen -------"));

jsp.setBounds(10,10,300,200);

jta.setText("Intrusion Detection");

panel.add(jsp);

clear = new JButton("Exit");

GUI.button(clear,"Exit");

clear.addActionListener(new Exit());

clear.setBackground(Color.GRAY);

clear.setForeground(Color.white);

clear.setBounds(5,265,310,20);

panel.add(clear);

jmb = new JMenuBar();

setJMenuBar(jmb);

menu = new JMenu("Main Menu");

jmb.add(menu);

cls = new JMenuItem("Set Up Frame");

menu.add(cls);

upload = new JMenuItem("Simulation Window");

menu.add(upload);

GUI.forMenuBar(jmb);

GUI.forMenu(menu,'M');

GUI.assign(cls,'C');

GUI.assign(upload,'U');

cls.addActionListener(new Clear());

upload.addActionListener(new Uploads());

main = new JPanel();

main.setBackground(Color.white);

main.setLayout(new FlowLayout());

main.setBounds(5,215,350,50);

panel.add(main);

err = new JPanel[12];

for(int i=0;i<err.length;i++) {

err[i] = new JPanel();

err[i].setBorder(BorderFactory.createRaisedBevelBorder());

main.add(err[i]);

}

setSize(350,350);

setLocation(300,200);

setVisible(true);

setResizable(false);

setDefaultCloseOperation(3);

}

public class Clear implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

try {

dispose();

new SetUpFrame();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

public class Exit implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

try {

System.exit(0);

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

public class Uploads implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

try

{

dispose();

new UserInterface();

}

catch(Exception e1)

{

e1.printStackTrace();

}

}

}

public static void main(String args[]) throws Exception

{

new Startup();

for(int i=0;i<11;i++) {

err[i].setBackground(Color.red);

Thread.sleep(200);

err[i].setBackground(Color.gray);

}

}

}

Node Movement Analysis:

package sensor\_priority.Com.multi.nw;

import java.applet.Applet;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.sql.\*;

import java.util.ArrayList;

import java.util.Hashtable;

import java.util.LinkedList;

import java.util.Random;

import javax.sound.sampled.Line;

import javax.swing.\*;

class MoveNode extends Thread {

JDesktopPane jdp;

JPanel lbl;

int packets;

Connection con = DB.getConnection();

String table="nodelog";

JPanel p;

String message,attack="";

JPanel[] plot;

Random random = new Random();

Random random1 = new Random();

Random random2 = new Random();

Random rq=new Random();

int ty=0;

int sensorBaseStationNode;

int[] pixels;

//Random random = new Random();

int speed = 10;

int destinationNode;

int sourceNode;

Graphics g;

int qq=1;

int cnt,cnt12;

public int getSourceNode() {

return sourceNode;

}

public void setSourceNode(int sourceNode) {

this.sourceNode = sourceNode;

}

public String getattacker() {

return attack;

}

public void setattacker(String amsg) {

this.attack = amsg;

}

public int getDestinationNode() {

//JOptionPane.showMessageDialog(null,"Destination...."+ destinationNode);

return destinationNode;

}

public void setDestinationNode(int destinationNode) {

this.destinationNode = destinationNode;

}

public int[] getPixels() {

return pixels;

}

public void setPixels(int[] pixels) {

this.pixels = pixels;

}

public int getSensorBaseStationNode() {

return sensorBaseStationNode;

}

public void setcnt(int cnt)

{

this.cnt=cnt;

// JOptionPane.showMessageDialog(null,"hellllo"+ this.cnt);

//JOptionPane.showMessageDialog(null,"ho"+ cnt);

}

public int getcnt() {

//JOptionPane.showMessageDialog(null,"jiiiiiiiii"+cnt);

return cnt;

}

public void setSensorBaseStationNode(int sensorBaseStationNode) {

this.sensorBaseStationNode = sensorBaseStationNode;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public String getTable() {

return table;

}

public void setTable(String table) {

this.table = table;

}

MoveNode(JDesktopPane jdp,JPanel[] plot){

this.jdp = jdp;

this.plot = plot;

}

public void run() {

send();

}

private void send() {

try {

String msg=getMessage();

Thread.sleep(100);

ResultSet rs123 = DB.getResultSet(con,"select max(node) as maxnode from nodelog");

int node123=0;

if (rs123.next())

{

node123=rs123.getInt("maxnode");

}

Random r=new Random();

int q=r.nextInt(node123);

int q1=random1.nextInt(node123);

System.out.println("Move Node............."+q);

//ResultSet rs = DB.getResultSet(con,"select \* from "+getTable()+" order by node asc");

ResultSet rs = DB.getResultSet(con,"select \* from nodelog order by node asc");

while(rs.next()){

int node = rs.getInt("node");

int xpos = rs.getInt("xpos");

int ypos = rs.getInt("ypos");

// int key = rs.getInt("accesskey");

int key=0;

// plot[node].setToolTipText(getMessage());

if("net1".equals(table)) {

if ((msg.equalsIgnoreCase("Fire")))// && (q==node) || (q1==node))

{

System.out.println("Inside Net1");

// JOptionPane.showMessageDialog(null, "Shortest Path"+q+ "&" +q1);

// if ((node==1) ||(q==node) || (q1==node) || (node123==node))

// {

System.out.println("sadsasad");

long startTime = System.currentTimeMillis();

MoveComponent mv = new MoveComponent(jdp,xpos,ypos,table);

mv.join();

long endTime = System.currentTimeMillis();

UserInterface.log(" \*\*\*\*\*\*\*\*\*\*\* Node "+node+" is authenticated \*\*\*\*\*\*\*\*\*\* ");

UserInterface.log("\*\*\*\*\*\*\*\*\*\*\*\* Message Received At Node "+node+"..");

plot[node].setToolTipText(getMessage());

DB.executeUpdate(con, "update "+getTable()+" set status='VALID' where node="+node);

UserInterface.baseStationJta.append("Node:"+node+" is accepted "+msg+"\n");

// }

// plot[node].setBackground(Global.PACKETS\_COLOR);

//break;

//Thread.sleep(500);

}

else

{

long startTime = System.currentTimeMillis();

MoveComponent mv = new MoveComponent(jdp,xpos,ypos,table);

mv.join();

break;

}

}

else if("net2".equals(table)) {

System.out.println("Inside Net2");

MoveComponent mv = new MoveComponent(jdp,xpos,ypos,table);

mv.join();

if(Global.ACCESS\_KEY!=key) {

plot[node].setBackground(Global.BLOCKED\_NODE\_COLOR);

UserInterface.log(" Node "+node+" is blocked due to the invalid key respond");

DB.executeUpdate(con, "update "+getTable()+" set status='BLOCKED' where node="+node);

UserInterface.baseStationJta.append("Node:"+node+" is blocked \n");

} else {

UserInterface.log("\*\*\*\*\*\*\*\*\*\* Node:"+node+" is authenticated \*\*\*\*\*\*\*\* \n");

DB.executeUpdate(con, "update "+getTable()+" set status='VALID' where node="+node);

UserInterface.baseStationJta.append("Node:"+node+" is accepted \n");

}

}

}

if("net3".equals(table)) {

System.out.println("Inside Net3");

ResultSet rs1 = DB.getResultSet(con,"select \* from "+getTable()+" order by node asc");

int sx = plot[sensorBaseStationNode].getX();

int sy = plot[sensorBaseStationNode].getY();

int difference = 0;

UserInterface.log(" Node "+sensorBaseStationNode+" is the Base Station Node");

while(rs1.next()) {

int node = rs1.getInt("node");

int xpos = rs1.getInt("xpos");

int ypos = rs1.getInt("ypos");

int key = rs1.getInt("accesskey");

plot[node].setToolTipText(message);

if(sx>xpos)

difference = sx - xpos;

else

difference = xpos - sx;

if(sy>ypos)

difference += sy - ypos;

else

difference += ypos - sy;

System.out.println(" DIFFERENCE THRESHOLD "+difference);

if(difference<Global.NODE\_DISTANCE\_THRESHOLD && node!=sensorBaseStationNode) {

JPanel l = new JPanel();

JLabel la=new JLabel("REQ");

la.setBounds(10,10,30,10);

l.add(la);

l.setBackground(Global.tempNodesColour);

l.setFont(new Font("Arial",12,Font.BOLD));

l.setBorder(BorderFactory.createRaisedBevelBorder());

l.setBounds(xpos, ypos, Global.moveNodeWidth, Global.moveNodeHeight);

jdp.add(l);

Thread.sleep(500);

l.setBounds(0,0,0,0);

jdp.remove(l);

if(Global.ACCESS\_KEY!=key) {

plot[node].setBackground(Global.BLOCKED\_NODE\_COLOR);

UserInterface.log(" Node "+node+" is blocked due to the invalid key respond");

DB.executeUpdate(con, "update "+getTable()+" set status='BLOCKED' where node="+node);

UserInterface.baseStationJta.append("Node:"+node+" is blocked \n");

} else {

UserInterface.log("\*\*\*\*\*\*\*\*\*\* Node:"+node+" is authenticated \*\*\*\*\*\*\*\* \n");

DB.executeUpdate(con, "update "+getTable()+" set status='VALID' where node="+node);

UserInterface.baseStationJta.append("Node:"+node+" is accepted \n");

}

UserInterface.log("----> SENSOR) Node:"+sensorBaseStationNode+" moved to the next location <---------");

}

}

int tx = random.nextInt(100);

int ty = random.nextInt(100);

boolean flag = random.nextBoolean();

if(flag)

plot[sensorBaseStationNode].setBounds((sx+tx), (sy+ty), Global.nodeBigWidth, Global.nodeHeight);

else

plot[sensorBaseStationNode].setBounds((sx-tx), (sy-ty), Global.nodeBigWidth, Global.nodeHeight);

}

//if("net4".equals(table)) {

if("nodelog".equals(table)) {

String msg1="huji";

System.out.println("Inside Net4");

// int clus1x=0,clus1y=0;

String qry0="select max(power) AS maxnode from nodelog where cluster='Cluster0'";

ResultSet rr0=DB.getResultSet(con,qry0);

int nn0=0;

if (rr0.next())

{

nn0=rr0.getInt("maxnode");

}

String qry120="select \* from nodelog where power="+nn0+" and cluster='Cluster0'";

ResultSet rr120=DB.getResultSet(con,qry120);

int nn120=0;

int clus1x0=0,clus1y0=0;

if (rr120.next())

{

nn120=rr120.getInt("node");

clus1x0=rr120.getInt("xpos");

clus1y0=rr120.getInt("ypos");

}

String qry="select max(power) AS maxnode from nodelog where cluster='Cluster1'";

ResultSet rr=DB.getResultSet(con,qry);

int nn=0;

if (rr.next())

{

nn=rr.getInt("maxnode");

}

String qry12="select \* from nodelog where power="+nn+" and cluster='Cluster1'";

ResultSet rr12=DB.getResultSet(con,qry12);

int nn12=0;

int clus1x=0,clus1y=0;

if (rr12.next())

{

nn12=rr12.getInt("node");

clus1x=rr12.getInt("xpos");

clus1y=rr12.getInt("ypos");

}

Random rnd123=new Random();

int qq=rnd123.nextInt(15);

//int qq=13;

///cnt++;

MoveNode mv85 = new MoveNode(jdp,plot);

cnt12=mv85.getcnt();

//JOptionPane.showMessageDialog(null,"jiooooo"+cnt12);

// JOptionPane.showMessageDialog(null,"Value of qq"+ qq+ " "+cnt12);

if(qq>0 && qq<=5)// && cnt==0)

{

JOptionPane.showMessageDialog(null,"No Attacker Found");

}

else if (qq>5 && qq<=10)//&& cnt ==1)

{

ty=rq.nextInt(5);

if(ty>=1)

{

JOptionPane.showMessageDialog(null,"Time Delay Occured"+ty);

JOptionPane.showMessageDialog(null,"Attacker at"+ty);

}

}

else if (qq>11 && qq<=15)// && cnt==2)

{

//ty=rq.nextInt(5);

ty=rq.nextInt(5);

if(ty>=1)

{

JOptionPane.showMessageDialog(null,"Packet Resizable Attack");

JOptionPane.showMessageDialog(null,"Attacker at "+ ty);

message=" dsdsdsasds "+message+" dshdjhdshad";

}

//cnt=0;

}

String qry1="select max(power) AS maxnode from nodelog where cluster='Cluster2'";

ResultSet rr1=DB.getResultSet(con,qry1);

int nn1=0;

int clus2x=0,clus2y=0;

if (rr1.next())

{

nn1=rr1.getInt("maxnode");

}

String qry21="select \* from nodelog where power="+nn1+" and cluster='Cluster2'";

ResultSet rr21=DB.getResultSet(con,qry21);

int nn21=0;

if (rr21.next())

{

nn21=rr21.getInt("node");

clus2x=rr21.getInt("xpos");

clus2y=rr21.getInt("ypos");

}

int pch1=(nn0-((nn0\*5)/100));

int pch2=(nn-((nn\*5)/100));

int pch3=(nn1-((nn1\*5)/100));

ResultSet rs1 = DB.getResultSet(con,"select \* from nodelog order by node asc");

int sx = plot[sourceNode].getX();

int sy = plot[sourceNode].getY();

UserInterface.log(" Node "+sourceNode+" is the Source Node");

UserInterface.log("Cluster Head in Cluster 0---> "+ nn120);

UserInterface.log("Power of Cluster Head in Cluster 0 ---> "+ nn0);

UserInterface.log("Cluster Head in Cluster I ---> "+ nn12);

UserInterface.log("Power of Cluster Head in Cluster I ---> "+ nn);

UserInterface.log("Cluster Head in Cluster II ---> "+ nn21);

UserInterface.log("Power of Cluster Head in Cluster II ---> "+ nn1);

while(rs1.next()) {

System.out.println("Inside Net4 rs1");

int node = rs1.getInt("node");

int xpos = rs1.getInt("xpos");

int ypos = rs1.getInt("ypos");

// int key =rs1.getInt("accesskey");

System.out.println("Node " +node);

System.out.println("Xop" +xpos);

System.out.println("Yop" +ypos);

// UserInterface.log("Node ---> " +node);

// UserInterface.log("Xop ---> " +xpos);

//UserInterface.log("Yop ---> " +ypos);

//UserInterface.log(

System.out.print("Message"+msg1);

// JOptionPane.showMessageDialog(null, "Position Identified using Shortest Node : ","Shortest Node",JOptionPane.PLAIN\_MESSAGE);

String grp="",grp1="";

if(getDestinationNode()==node) {

UserInterface.log("Destination Selected ---> "+ getDestinationNode());

int spow=0;

// DB.executeUpdate(con, "update nodelog set power='"+spow+"' where node="+sourceNode+"");

int dpow=0;

String qry57="select \* from nodelog where node="+getDestinationNode()+"";

ResultSet rs57 = DB.getResultSet(con, qry57);

while(rs57.next())

{

grp1=rs57.getString("cluster");

dpow=rs57.getInt("power");

}

dpow=dpow-(dpow\*2/100);

DB.executeUpdate(con, "update nodelog set power='"+dpow+"' where node="+getDestinationNode()+"");

UserInterface.log("Destination Node belongs to ---> "+ grp1);

String pro=sourceNode +" is sending Message to "+getDestinationNode();

String pqry="insert into process values('"+grp+"','"+pro+"')";

DB.executeUpdate(con, pqry);

String routepath="";

if (grp.equals(grp1))

{

UserInterface.log("Message is being sent within the same group");

long startTime = System.currentTimeMillis();

int[] nearest = Path.getNearestNodeInfo(con, getSourceNode(), getDestinationNode(), sx, sy,nn120,clus1x0,clus1y0,nn12,clus1x,clus1y,nn21,clus2x,clus2y);

String position = " (XPOS : "+xpos+") (YPOS : "+ypos+" ) \n Shortest path "+nearest[0]+" \n (XPOS : "+nearest[1]+") (YPOS :"+nearest[1]+" )";

String pp1="(XPOS : "+xpos+") (YPOS : "+ypos+" )\n";

System.out.println("Nearest 0"+nearest[0]);

System.out.println("Nearest 1"+nearest[1]);

System.out.println("Nearest 2"+nearest[2]);

MoveComponent mv1 = new MoveComponent(jdp, nearest[1], nearest[2], table);

long endTime = System.currentTimeMillis();

mv1.setSourceX(sx);

mv1.setSourceY(sy);

mv1.setDestinationNode(nearest[0]);

mv1.join();

//routepath=" ------> "+getSourceNode()+" ------->";

//routepath+=" ------> "+nearest[0]+" ------->";

int npow=0;

String y57="select \* from nodelog where node="+nearest[0]+"";

ResultSet ry57 = DB.getResultSet(con, y57);

while(ry57.next())

{

grp1=ry57.getString("cluster");

npow=ry57.getInt("power");

}

npow=npow-(npow\*2/100);

DB.executeUpdate(con, "update nodelog set power='"+npow+"' where node="+nearest[0]+"");

// DB.executeUpdate(con, "update nodelog set power='"+spow+"' where node="+sourceNode+"");

plot[node].setToolTipText(message);

LinesComponent1 l=new LinesComponent1();

l.addLine(500, 500, 500, 500);

// UserInterface.baseStationJta.append("Node:"+node+" is accepted "+msg+"\n");

BufferedWriter bw = null;

File f1 = new File(".\\RecievedFiles\\Packet1.txt");

f1.delete();

try {

bw = new BufferedWriter(new FileWriter(".\\RecievedFiles\\Packet1.txt", true));

bw.write(message);

bw.newLine();

bw.flush();

} catch (IOException ioe) {

ioe.printStackTrace();

} finally {

if (bw != null) try {

bw.close();

} catch (IOException ioe2) {

// just ignore it

System.out.println(ioe2);

}

}

MoveComponent mv = new MoveComponent(jdp,xpos,ypos,table);

mv.setSourceX(nearest[1]);

mv.setSourceY(nearest[2]);

mv.setDestinationNode(getDestinationNode());

mv.join();

//routepath +=getDestinationNode();

System.out.println(" Difference Time \*\*\*\*\*\*\*\*\*\*\*\*\* "+(endTime-startTime));

//new graph1();

String pro1=" Message send successfully from "+sourceNode +" to "+ getDestinationNode()+" within the same cluster";

String pqry1="insert into process values('"+grp+"','"+pro1+"')";

DB.executeUpdate(con, pqry1);

JOptionPane.showMessageDialog(null,"Message Received Successfully...");

//JOptionPane.showMessageDialog(null,"Node Positions Identified \n"+ position);

// JOptionPane.showMessageDialog(null, "Position Identified using Shortest Node : "+nearest[0],"Shortest Node",JOptionPane.PLAIN\_MESSAGE);

}

else if(!grp.equals(grp1))

{

UserInterface.log("Message is being sent between the two groups");

ArrayList al=new ArrayList();

long startTime = System.currentTimeMillis();

int[] nearest = Path.getNearestNodeInfo(con, getSourceNode(), getDestinationNode(), sx, sy,nn120,clus1x0,clus1y0,nn12,clus1x,clus1y,nn21,clus2x,clus2y);

String position = " (XPOS : "+xpos+") (YPOS : "+ypos+" ) \n Shortest path "+nearest[0]+" \n (XPOS : "+nearest[1]+") (YPOS :"+nearest[1]+" )";

String pp1="(XPOS : "+xpos+") (YPOS : "+ypos+" )\n";

System.out.println("Nearest 0"+nearest[0]);

System.out.println("Nearest 1"+nearest[1]);

System.out.println("Nearest 2"+nearest[2]);

System.out.println("Nearest 3"+nearest[3]);

System.out.println("Nearest 4"+nearest[4]);

System.out.println("Nearest 5"+nearest[5]);

System.out.println("Nearest 6"+nearest[6]);

System.out.println("Nearest 7"+nearest[7]);

System.out.println("Nearest 8"+nearest[8]);

UserInterface.log("Nearest 6"+nearest[6]);

UserInterface.log("Nearest 7"+nearest[7]);

UserInterface.log("Nearest 8"+nearest[8]);

// al.add(sourceNode);

al.add(sourceNode);

// al.add(nearest[0]);

al.add(nearest[3]);

//al.add(nearest[6]);

al.add(getDestinationNode());

String qry56="select \* from nodelog where node="+sourceNode+"";

ResultSet rs56 = DB.getResultSet(con, qry56);

while(rs56.next())

{

grp=rs56.getString("cluster");

spow=rs56.getInt("power");

}

spow=spow-(spow\*2/100);

UserInterface.log("Source Node belongs to ---> "+ grp);

DB.executeUpdate(con, "update nodelog set power='"+spow+"' where node="+sourceNode+"");

plot[getSourceNode()].setBackground(Global.BLOCKED\_NODE\_COLOR);

JOptionPane.showMessageDialog(null,"TY "+ty+" QQ "+qq);

if(ty>0 && ty<=5 && qq>=0 && qq<5)

{

qq+=5;

}

if(ty>0 && ty<=5 && qq>5 && qq<=10)

{

Random rn=new Random();

JOptionPane.showMessageDialog(null,"ARRAY SIZE "+al.size());

ty=rn.nextInt(al.size());

JOptionPane.showMessageDialog(null,"SELECTED VALUE "+ty);

if(ty>0)

{

plot[ty].setBackground(Global.ATTACKER\_NODE\_COLOR);

UserInterface.log(">>>>>Time Delay Occured at " +ty);

Thread.sleep(2000);

}

qq+=5;

//plot[ty].setBackground(Global.ATTACKER\_NODE\_COLOR);

}

else if(ty>0 && ty<=5 && qq>=11 && qq<=15)

{

Random rn=new Random();

JOptionPane.showMessageDialog(null,"ARRAY SIZE "+al.size());

ty=rn.nextInt(al.size());

JOptionPane.showMessageDialog(null,"SELECTED VALUE "+ty);

if(ty>0)

{

plot[ty].setBackground(Global.ATTACKER\_NODE\_COLOR);

}

qq=0;

}

MoveComponent mv1 = new MoveComponent(jdp, nearest[1], nearest[2], table);

long endTime = System.currentTimeMillis();

mv1.setSourceX(sx);

mv1.setSourceY(sy);

//mv1.setDestinationNode(nearest[0]);

mv1.setDestinationNode(nearest[0]);

Thread.sleep(100);

mv1.join();

plot[nearest[0]].setBackground(Global.BLOCKED\_NODE\_COLOR);

int npow=0;

String y57="select \* from nodelog where node="+nearest[0]+"";

ResultSet ry57 = DB.getResultSet(con, y57);

while(ry57.next())

{

grp1=ry57.getString("cluster");

npow=ry57.getInt("power");

}

npow=npow-(npow\*10/100);

DB.executeUpdate(con, "update nodelog set power='"+npow+"' where node="+nearest[0]+"");

UserInterface.log(">>>>>update nodelog set power='"+npow+"' where node="+nearest[0]+" ");

Hashtable h=new Hashtable();

Random hh= new Random();

EncryptDecrypt ed=new EncryptDecrypt();

String m1=msg+""+hh.nextInt(9999);

UserInterface.log("Key At "+nearest[0]+" "+ed.EncryptDecrypt1(m1));

MoveComponent mv91 = new MoveComponent(jdp,180, 150, table);

mv91.setSourceX(180);

mv91.setSourceY(150);

mv91.setDestinationNode(nearest[1]);

//mv91.setDestinationNode(0);

Thread.sleep(100);

mv91.join();

String m=msg+""+hh.nextInt(9999);

UserInterface.log("Key At "+nearest[3]+" "+ed.EncryptDecrypt1(m));

MoveComponent mv51 = new MoveComponent(jdp,nearest[7] ,nearest[8], table);

mv51.setSourceX(nearest[7]);

mv51.setSourceY(nearest[8]);

mv51.setDestinationNode(nearest[6]);

Thread.sleep(100);

mv51.join();

plot[nearest[6]].setBackground(Global.BLOCKED\_NODE\_COLOR);

String ky1=h.keys().toString();

int n1pow=0;

String y157="select \* from nodelog where node="+nearest[6]+"";

ResultSet ry157 = DB.getResultSet(con, y157);

while(ry157.next())

{

grp1=ry157.getString("cluster");

n1pow=ry157.getInt("power");

}

n1pow=n1pow-(n1pow\*2/100);

DB.executeUpdate(con, "update nodelog set power='"+n1pow+"' where node="+nearest[6]+"");

UserInterface.log(">>>>>update nodelog set power='"+n1pow+"' where node="+nearest[6]+" ");

plot[node].setToolTipText(message+ky1);

plot[node].setToolTipText(message);

BufferedWriter bw = null;

File f1 = new File(".\\RecievedFiles\\Packet1.txt");

f1.delete();

try {

bw = new BufferedWriter(new FileWriter(".\\RecievedFiles\\Packet1.txt", true));

bw.write(message);

bw.newLine();

bw.flush();

} catch (IOException ioe) {

ioe.printStackTrace();

} finally {

if (bw != null) try {

bw.close();

} catch (IOException ioe2) {

// just ignore it

}

}

//MoveComponent mv = new MoveComponent(jdp,xpos,ypos,table);

MoveComponent mv = new MoveComponent(jdp,nearest[4], nearest[5],table);

//mv.setSourceX(nearest[1]);

//mv.setSourceY(nearest[2]);

mv.setSourceX(180);

mv.setSourceY(150);

mv.setDestinationNode(getDestinationNode());

Thread.sleep(100);

mv.join();

plot[nearest[3]].setBackground(Global.BLOCKED\_NODE\_COLOR);

int n2pow=0;

String y257="select \* from nodelog where node="+nearest[3]+"";

ResultSet ry257 = DB.getResultSet(con, y257);

while(ry257.next())

{

grp1=ry257.getString("cluster");

n2pow=ry257.getInt("power");

}

n2pow=n2pow-(n2pow\*10/100);

DB.executeUpdate(con, "update nodelog set power='"+n2pow+"' where node="+nearest[3]+"");

UserInterface.log(">>>>>update nodelog set power='"+n2pow+"' where node="+nearest[3]+" ");

UserInterface.log(" \*\*\*\*\*\*\*\*\*\*\* Node "+node+" is authenticated \*\*\*\*\*\*\*\*\*\* ");

UserInterface.log(" \*\*\*\*\*\*\*\*\*\*\* Decrypting Message \*\*\*\*\*\*\*\*\*\* ");

UserInterface.log("\*\*\*\*\*\*\*\*\*\*\*\* Message Received At Node "+node+"..");

plot[node].setToolTipText(getMessage());

MoveComponent mv2 = new MoveComponent(jdp,xpos,ypos,table);

mv2.setSourceX(nearest[4]);

mv2.setSourceY(nearest[5]);

mv2.setDestinationNode(getDestinationNode());

mv2.join();

plot[getDestinationNode()].setBackground(Global.BLOCKED\_NODE\_COLOR);

int dpow1=0;

String yd="select \* from nodelog where node="+getDestinationNode()+"";

ResultSet ryd= DB.getResultSet(con, yd);

while(ryd.next())

{

grp1=ryd.getString("cluster");

dpow1=ryd.getInt("power");

}

dpow1=dpow1-(dpow1\*2/100);

DB.executeUpdate(con, "update nodelog set power='"+dpow1+"' where node="+getDestinationNode()+"");

UserInterface.log(">>>>>update nodelog set power='"+n2pow+"' where node="+nearest[3]+" ");

System.out.println(" Difference Time \*\*\*\*\*\*\*\*\*\*\*\*\* "+(endTime-startTime));

String pro1=" Message send successfully from "+sourceNode +" to "+ getDestinationNode()+" between different cluster";

String pqry1="insert into process values('"+grp+"','"+pro1+"')";

DB.executeUpdate(con, pqry1);

String pro2= getDestinationNode() +" Received Message Successfully from "+sourceNode ;

String pqry2="insert into process values('"+grp1+"','"+pro2+"')";

DB.executeUpdate(con, pqry2);

JOptionPane.showMessageDialog(null,"Message Received Successfully...");

plot[getSourceNode()].setBackground(Global.backgroundColour);

plot[nearest[0]].setBackground(Global.backgroundColour);

plot[nearest[6]].setBackground(Global.backgroundColour);

plot[nearest[3]].setBackground(Global.backgroundColour);

plot[getDestinationNode()].setBackground(Global.backgroundColour);

if(ty>0 && ty<=5 && qq>0 && qq<=5)

{

plot[ty].setBackground(Global.backgroundColour);

}

if(ty>0 && ty<=5 && qq>5 && qq<=10)

{

plot[ty].setBackground(Global.backgroundColour);

}

}

}

}

}

} catch(Exception e){

e.printStackTrace();

}

}

}

class MoveComponent extends Thread {

JDesktopPane mvp;

LinesComponent lc=new LinesComponent();

int xpos;

int ypos;

JPanel panel;

JLabel ll;

String table;

boolean flag = false;

Random random = new Random();

DefaultDesktopManager ddm;

int destinationNode;

int sourceX;

public int getSourceX() {

return sourceX;

}

public void setSourceX(int sourceX) {

this.sourceX = sourceX;

}

public int getSourceY() {

return sourceY;

}

public void setSourceY(int sourceY) {

this.sourceY = sourceY;

}

int sourceY;

public int getDestinationNode() {

return destinationNode;

}

public void setDestinationNode(int destinationNode) {

this.destinationNode = destinationNode;

}

MoveComponent(JDesktopPane p,int xpos,int ypos,String table) {

this.mvp = p;

this.xpos = xpos;

this.ypos = ypos;

this.table = table;

start();

ddm = new DefaultDesktopManager();

}

public synchronized void run() {

try {

int lineNo = 0;

panel = new JPanel();

ll=new JLabel("REQ");

ll.setBounds(10,10,30,10);

ll.setForeground(Color.white);

panel.add(ll);

panel.setBounds(0,0,Global.moveNodeWidth,Global.moveNodeHeight);

panel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createBevelBorder(1,Global.nodesColour,Color.GRAY),"",2,3,new Font("Arial",Font.BOLD,8),Color.white));

panel.setLayout(null);

panel.setBackground(Global.tempNodesColour);

mvp.add(panel);

mvp.add(lc);

Thread.sleep(10);

if("net1".equals(table)) {

long startTime = System.currentTimeMillis();

ll.setVisible(true);

for(int i=UserInterface.baseStationPanel.getY()+100;i<ypos;i++) {

ddm.dragFrame(panel,(UserInterface.baseStationPanel.getX()+100),i);

Thread.sleep(Global.speed);

lineNo = i;

}

for(int i=UserInterface.baseStationPanel.getX()+100;i<xpos;i++) {

ddm.dragFrame(panel,i,lineNo);

Thread.sleep(Global.speed);

}

long endTime = System.currentTimeMillis();

long difference = endTime - startTime;

System.out.println(" %% REQUEST Time Taken ----> "+difference);

UserInterface.log(" %% REQUEST Time Taken ----> "+difference);

ll.setText("REP");

startTime = System.currentTimeMillis();

for(int i=ypos;i>UserInterface.baseStationPanel.getY()+100;i--) {

ddm.dragFrame(panel,xpos,i);

if(getFlag()==true) {

Thread.sleep(Global.speed\*2);

} else {

Thread.sleep(Global.speed);

}

}

if(checkStatus()==true) {

setFlag(true);

}

for(int i=xpos;i>UserInterface.baseStationPanel.getX()+100;i--) {

ddm.dragFrame(panel,i,(UserInterface.baseStationPanel.getY()+100));

if(getFlag()==true) {

Thread.sleep(Global.speed\*2);

} else {

Thread.sleep(Global.speed);

}

}

endTime = System.currentTimeMillis();

long responseDifference = endTime - startTime;

long delay = 0;

if(responseDifference>=difference) {

delay = responseDifference - difference;

} else {

delay = difference-responseDifference;

}

System.out.println(" %% RESPONSE Time Taken ----> "+responseDifference);

System.out.println(" %% Delay ----> "+delay);

UserInterface.log(" %% RESPONSE Time Taken ----> "+responseDifference);

UserInterface.log(" %% Delay ----> "+delay);

}

else if("nodelog".equals(table)) {

long startTime = System.currentTimeMillis();

if(getSourceY()>ypos && getSourceX()<xpos) {

int sd = 0;

for(int i=getSourceX();i<xpos;i++) {

ddm.dragFrame(panel,i,getSourceY());

Thread.sleep(Global.speed);

sd = i;

}

for(int i=getSourceY();i>ypos;i--) {

ddm.dragFrame(panel,sd,i);

Thread.sleep(Global.speed);

}

Thread.sleep(10);

}

long endTime = System.currentTimeMillis();

long difference = endTime - startTime;

System.out.println(" %% REQUEST Time Taken ----> "+difference);

UserInterface.log(" %% REQUEST Time Taken ----> "+difference);

if(getSourceY()<ypos && getSourceX()<xpos) {

int sd = 0;

for(int i=getSourceX();i<xpos;i++) {

ddm.dragFrame(panel,i,getSourceY());

Thread.sleep(Global.speed\*2);

sd = i;

}

for(int i=getSourceY();i<ypos;i++) {

ddm.dragFrame(panel,sd,i);

Thread.sleep(Global.speed);

}

Thread.sleep(10);

//setFlag(true);

}

if(getSourceY()>ypos && getSourceX()>xpos) {

int sd = 0;

for(int i=getSourceX();i>xpos;i--) {

ddm.dragFrame(panel,i,getSourceY());

Thread.sleep(Global.speed);

sd = i;

Thread.sleep(10);

}

for(int i=getSourceY();i>ypos;i--) {

ddm.dragFrame(panel,sd,i);

Thread.sleep(Global.speed);

}

Thread.sleep(10);

//setFlag(true);

}

if(getSourceY()<ypos && getSourceX()>xpos) {

int sd = 0;

for(int i=getSourceX();i>xpos;i--) {

ddm.dragFrame(panel,i,getSourceY());

Thread.sleep(Global.speed);

sd = i;

// Thread.sleep(500);

}

for(int i=getSourceY();i<ypos;i++) {

ddm.dragFrame(panel,sd,i);

Thread.sleep(Global.speed);

}

Thread.sleep(10);

}

endTime = System.currentTimeMillis();

long responseDifference = endTime - startTime;

long delay = 0;

if(responseDifference>=difference) {

delay = responseDifference - difference;

} else {

delay = difference-responseDifference;

}

if (delay>=5000)

{

setFlag(true);

}

else

{

setFlag(false);

}

//LinesComponent1 lc=new LinesComponent1();

// lc.addLine(500, 500,200, 500);

//lc.addLine(500, 250,250, 500, Color.BLACK);

// panel.add(lc);

// mvp.add(panel);

System.out.println(" %% RESPONSE Time Taken ----> "+responseDifference);

System.out.println(" %% Delay ----> "+delay);

UserInterface.log(" %% RESPONSE Time Taken ----> "+responseDifference);

UserInterface.log(" %% Delay ----> "+delay);

}

else if("net2".equals(table)) {

for(int i=UserInterface.baseStationPanel.getY()+100;i<ypos;i++) {

ddm.dragFrame(panel,(UserInterface.baseStationPanel.getX()+100),i);

Thread.sleep(Global.speed);

lineNo = i;

}

for(int i=UserInterface.baseStationPanel.getX()+100;i<xpos;i++) {

ddm.dragFrame(panel,i,lineNo);

Thread.sleep(Global.speed);

}

ll.setText("REP");

for(int i=ypos;i>UserInterface.baseStationPanel.getY()+100;i--) {

ddm.dragFrame(panel,xpos,i);

Thread.sleep(Global.speed);

}

for(int i=xpos;i>UserInterface.baseStationPanel.getX()+100;i--) {

ddm.dragFrame(panel,i,(UserInterface.baseStationPanel.getY()+100));

Thread.sleep(Global.speed);

}

/\*for(int i=xpos;i>UserInterface.baseStationPanel.getX()+100;i--) {

ddm.dragFrame(panel,i,(UserInterface.baseStationPanel.getY()+100));

Thread.sleep(Global.speed);

}\*/

}

panel.setBounds(0,0,0,0);

mvp.remove(panel);

} catch (Exception e) {

e.printStackTrace();

}

}

boolean checkStatus() {

try {

if("net1".equals(table)) {

int status = random.nextInt(10);

if(status==0 || status==5 || status==7 || status==3)

return true;

else

return false;

}

} catch (Exception e) {

e.printStackTrace();

}

return false;

}

void setFlag(boolean flag) {

this.flag = flag;

}

boolean getFlag() {

return this.flag;

}

}

class MoveSensorComponent extends Thread {

JDesktopPane mvp;

int xpos;

int ypos;

JPanel panel;

boolean flag = false;

int[] pixels;

JPanel[] plot;

MoveSensorComponent(JDesktopPane p,int xpos,int ypos,int[] pixels,JPanel[] plot) {

this.mvp = p;

this.xpos = xpos;

this.ypos = ypos;

this.pixels = pixels;

this.plot = plot;

}

public synchronized void run() {

try {

} catch (Exception e) {

e.printStackTrace();

}

}

}

/\*\*

\*

\* @author Arun

\*/

class LinesComponent extends JComponent {

class Line{

final int x1;

final int y1;

final int x2;

final int y2;

final Color color;

public Line(int x1, int y1, int x2, int y2, Color color) {

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.color = color;

}

public final LinkedList<Line> lines = new LinkedList<Line>();

public void addLine(int x1, int x2, int x3, int x4) {

JOptionPane.showMessageDialog(null,"Adding Lines..........");

addLine(x1, x2, x3, x4, Color.black);

JOptionPane.showMessageDialog(null,"Added Lines..........");

}

public void addLine(int x1, int x2, int x3, int x4, Color color) {

JOptionPane.showMessageDialog(null,"Lines..........");

// lines.add(new Line(x1,x2,x3,x4, color));

JOptionPane.showMessageDialog(null,"OutSide Lines..........");

repaint();

//g.drawLine(x1, y1, line.x2, line.y2);

JOptionPane.showMessageDialog(null,"Inside Lines..........");

}

public void clearLines() {

lines.clear();

repaint();

}

//@Override

public void paintComponent(Graphics g) {

// super.paintComponent(g);

// for (Line line : lines) {

g.setColor(Color.BLUE);

JOptionPane.showMessageDialog(null,"HELO");

//g.drawLine(x1, y1, x2, y2);

g.drawLine(250,500,500, 250);

// }

}

}

}

Encryption/ Decryption:

package sensor\_priority.Com.multi.nw;

import java.security.InvalidKeyException;

import java.security.Key;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

public class EncryptDecrypt {

private static String algorithm = "DESede";

private static Key key = null;

private static Cipher cipher = null;

//private String input=null;

public String EncryptDecrypt1(String message)throws Exception

{

setUp();

byte[] encryptionBytes = null;

String input = message;

encryptionBytes = encrypt(input);

String str=encryptionBytes.toString();

// System.out.println(encryptionBytes);

//System.out.println("Recovered: " + decrypt(encryptionBytes));

String str2=str+"|"+decrypt(encryptionBytes);

return str2;

}

private static byte[] encrypt(String input)throws InvalidKeyException,BadPaddingException,IllegalBlockSizeException {

cipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] inputBytes = input.getBytes();

return cipher.doFinal(inputBytes);

}

private static String decrypt(byte[] encryptionBytes)throws InvalidKeyException,BadPaddingException,IllegalBlockSizeException {

cipher.init(Cipher.DECRYPT\_MODE, key);

byte[] recoveredBytes =

cipher.doFinal(encryptionBytes);

String recovered =

new String(recoveredBytes);

return recovered;

}

private static void setUp() throws Exception {

key = KeyGenerator.getInstance(algorithm).generateKey();

cipher = Cipher.getInstance(algorithm);

}

}