MOBILE SERVICE DISCOVERY PROTOCOL (MSDP) FOR MOBILE AD-HOC NETWORKS

A PROJECT REPORT

Submitted by

K.S. KARTHIKEYAN **71205104017**

S. MANOJ **71205104020**

In partition fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI-600 025

APRIL 2009

BONAFIDE CERTIFICATE

Certified that this project report entitled "**MOBILE SERVICE DISCOVERY PROTOCOL (MSDP) FOR MOBILE AD-HOC NETWORKS**" is the bonafide work of K.S. Karthikeyan, and S. Manoj, who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. S.Thangasamy, Ph.D.,

Head of the department,

Department of Computer Science

&Engineering,

Kumaraguru College Of Technology,

Coimbatore-641006.

SIGNATURE

Mrs. V.Vanitha, M.E.,

Assistant Professor,

Department of Computer Science

&Engineering,

Kumaraguru College Of Technology,

Coimbatore-641006.

The candidate with University Register Number 71205104017 and 71205104020 were examined by us in the project viva-voce examination held on 27.04.09

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled **"MOBILE SERVICE DISCOVERY PROTOCOL (MSDP) FOR MOBILE AD-HOC NETWORKS"** is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Chennai.

Place: Coimbatore

Date: 21.04.09

(K.S.Karthikeyan)

(S.Manoj)

ACKNOWLEDGEMENT

The exhilaration achieved on successful completion of any task should be shared with the people behind the venture. At the onset, we thank the management of our college for having provided the excellent facilities to carry on the project.

We extend our sincere thanks to our vice principal Mr.R.Annamalai, Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facility to work on this project.

We would like to make a special acknowledgement and thanks to Dr.S.Thangasamy, Ph.D., Dean, Professor and Head of Department of Computer Science &Engineering, for his support and encouragement throughout the project.

We express deep gratitude and gratefulness to our guide Mrs.V.Vanitha, M.E, Assistant Professor, Department of Computer Science &Engineering, for her supervision, enduring patience, active involvement and guidance.

We would like to thank our project coordinator, Mrs.P.Devaki, M.S., Assistant Professor for her support during the course of our project.

We would like to convey our honest thanks to all members of our Department staff for their enthusiasm and wealth of experience from which we have greatly benefited.

TABLE OF CONTENTS

ABSTRACT

In a Mobile Ad-hoc NETwork (MANET), every node can be mobile and act as a routing agent to forward packets on behalf of others. Two nodes can communicate with each other in a multi-hop manner through the help of other nodes. A MANET can be formed without any fixed infrastructure, and thus has low construction cost and can be built rapidly. MANETs are especially useful for disaster relief and military operations.

One of a Critical Challenge in MANET is service discovery. An efficient service discovery mechanism should have low network resource consumption, fast response time, and high query success rate. Many service discovery mechanisms exist in today's Internet. However, these mechanisms are not effective in handling the service discovery needs in MANET's where the networks are dynamically changing. This creates the need for new service discovery protocols for MANETs.

The proposed Mobile Service Discovery Protocol (MSDP) uses a dynamic clustering algorithm to group nodes in a MANET, and utilizes Distributed Hash Tables (DHTs) to efficiently cache service information in a peer-to-peer manner.

LIST OF FIGURES

LIST OF ABBREVIATIONS

DHTs	-	Distributed Hash Tables
DNS	-	Domain Name System
GSD	-	Group-based Service Discovery
MANET	-	Mobile Ad-hoc Network
MSDP	-	Mobile Service Discovery Protocol
ODMRP	-	On-Demand Multicast Routing Protocol
P2P	-	Peer-to-Peer
PAN	-	Personal Area Network
RMI	-	Remote Method Invocation
RPC	-	Remote Procedure Calls
SDP	-	Service Discovery Protocol
SLP	-	Service Location Protocol
TCP	-	Transmission Control Protocol

CHAPTER 1

INTRODUCTION

1.1. Mobile ad-hoc network

The field of wireless networking emerges from the integration of personal computing, cellular technology, and the Internet. This is due to the increasing interactions between communication and computing, which is changing information access from "anytime anywhere" into "all the time, everywhere." At present, a large variety of networks exists, ranging from the well-known infrastructure of cellular networks to non-infrastructure wireless ad-hoc networks.

Unlike a fixed wireless network, wireless ad-hoc or on-the-fly networks are characterized by the lack of infrastructure. Nodes in a mobile ad-hoc network are free to move and organize themselves in an arbitrary fashion. Each user is free to roam about while communicating with others. The path between each pair of the users may have multiple links, and the ratio between them can be heterogeneous. This allows an association of various links to be a part of the same network. Mobile ad-hoc networks can operate in a stand-alone fashion or could possibly be connected to a larger network such as the Internet.

Ad-hoc networks are suited for use in situations where an infrastructure is unavailable or to deploy one is not cost effective. One of many possible uses of mobile ad-hoc networks is in some business environments, where the need for collaborative computing might be more important outside the office environment than inside, such as in a business meeting outside the office to brief clients on a given assignment. The cooperation of the users is necessary to the operation of ad-hoc networks therefore game theory provides a good basis to analyze the networks.

A mobile ad-hoc network can also be used to provide crisis management services applications, such as in disaster recovery, where the entire communication infrastructure is destroyed and resorting communication quickly is crucial. By using a mobile ad-hoc network, an infrastructure could be set up in hours instead of weeks, as is required in the case of wired line communication. Another application example of a mobile ad-hoc network is Bluetooth, which is designed to support a personal area network by eliminating the need of wires between various devices, such as printers and personal digital assistants. The famous IEEE 802.11 or Wi-Fi protocol also supports an ad-hoc network system in the absence of a wireless access point.

In conclusion, mobile ad-hoc networks allow the construction of flexible and adaptive networks with no fixed infrastructure. These networks are expected to play an important role in the future wireless generation. Future wireless technology will require highly-adaptive mobile networking technology to effectively manage multi-hop ad-hoc network clusters, which will not only operate autonomously but also will be able to attach at some point to the fixed networks.

1.2. Data and Services Components

The mass market of wireless devices suggests novel service deployment scenarios where there are no constraints on device mobility and distributed applications are the result of impromptu collaborations among wireless peers. In these scenarios, not only wireless nodes should have location-aware access to distributed resources without requiring static knowledge about their execution environment, but also resources should be permanently available, not depending on node movement, disconnection, and battery shortage. Some research activities are investigating MANET specific solutions to bind/rebind to newly discovered distributed resources, thus enabling wireless clients to automatically

redirect requests to service components also by considering their mutual location.

Only few state-of-the-art proposals have started to face the very challenging issue of resource replication in mobile environments, with the goal of increasing access probability and effectiveness. Most investigations have recently addressed the issue of information availability in cellular and infrastructure- mode IEEE 802.11 networks. However, in that context, it is possible to exploit the fixed part of the network infrastructure, e.g., to store and replicate personal data at highly available servers on wired stable links. The most critical issue in infrastructure-based scenarios is to properly manage user disconnections during update operations on shared data, possibly by automatically handling the reconciliation of multiple modified copies. Due to the complete lack of a static support infrastructure, the effective replication of data and service components in MANET dynamic environments is a hard challenge, which requires re-thinking and significantly modifying traditional replication approaches. So far, the research has mainly focused on replication to ensure data availability in case of network partitions. Most proposals assume that wireless nodes are aware of their physical position. Other research activities aim at answering strict requirements about replica synchronization, and therefore impose a heavy overhead, in terms of both network traffic and requested time to ensure the consistency of all replicas.

Due to connectivity issues, e.g., high loss rates and frequent network partitioning, the management of data/service component replicas is a very hard task to perform in an effective and lightweight way when dealing with general-purpose MANETs and with strict consistency requirements. Therefore, system focus on a specific deployment scenario of increasing relevance for the service provisioning market, called dense MANET in the following. It includes a large number of wireless devices located in a relatively small area at the same time.

CHAPTER 2

LITERATURE REVIEW

2.1. Existing System

Many service discovery mechanisms are already widely used in today's Internet. One example is the MX record in the Domain Name System (DNS) proposed by IETF. By looking up the MX record in the DNS servers, users can locate mail servers and access mail services. Another example is the search service provided by Google. With the proliferation of the Internet in recent years, the number of WebPages has grown tremendously. In order to let users find the information they want, search engines like Google have created large databases to cache the information on WebPages throughout the Internet, and allow users to use keywords to search for the information over the Internet. The above mechanisms are mainly designed for use over the Internet. They rely on central servers for information storage and dissemination, and thus are not suitable for the dynamically changing MANET environment.

The ubiquitous personal computers and laptops and the availability of Personal Area Network (PAN) technologies motivated the design of several service discovery protocols. Examples are Universal Plug and Play (UPnP), Jini, Salutation, Service Location Protocol (SLP), and Bluetooth Service Discovery Protocol (SDP).

However, these protocols aim to perform service discovery among hard ward devices, and do not fit our purposes here.

Peer-to-peer (P2P) networks have been increasingly popular. Besides being a growing way of file sharing over the Internet, P2P networking technologies also provide a promising means for service information sharing and dissemination over MANETs.

Napster and Gnutella

Early P2P networks like Napster relied on centralized index servers. In Napster, some servers were responsible for storing all the node information in the P2P network. Thus it suffered from scalability and robustness issues. Later P2P technologies like Gnutella were designed to operate distributed. All nodes in Gnutella store some information about other nodes, and nodes intended to join the network can contact any existing nodes in the network. This has made Gnutella much more robust than Napster. However, information queries are flooded throughout the network in Gnutella, so it still suffers from scalability issues. Recently, Structured Peer-to-Peer networks have brought new research interests in P2P networking. Structured P2P networks are also distributed systems. With the help of Distributed Hash Tables (DHTs), nodes in structured P2P networks can have a definite guide to route their query packets. The use of DHTs has made structured P2P even more scalable and robust comparing to Gnutella.

Orion

Many research works on service discovery in MANET have been conducted over the past few years. A. Klemm et el. proposed a P2P file sharing protocol called ORION for ad hoc networks. ORION searches files by broadcasting messages, and making nodes on the searching path cache some information about the files. D. Chakraborty et el. introduced Group-based Service Discovery (GSD). GDS uses DARPA Agent Markup Language (DAML) to describe service information. It also utilizes the hierarchical semantic structures of DAML to create different groups of service information, and service queries can be routed through the help of the grouped information. L. Cheng proposed to integrate service discovery messages into the On-Demand Multicast Routing Protocol (ODMRP) to reduce the service query overheads. M. Klein et el. proposed a Multi-layered Cluster structure which is formed

according to service information Service discovery messages can be delivered with the help of this clustered structure. O. Ratsimor et el. introduced an Alliance-based Service Discovery protocol which is called Allia. Users in Allia can define rules regarding service discovery. According to these rules, dynamic groups can be formed to cache service information, which speeds up the query process and reduces the network resources needed. S. Helal et el. demonstrates a service discovery protocol implementation named Konark, which is programmed to run on cell phones. U. C. Kozat proposed to integrate service discovery protocols into network layer routing protocols. This cross-layered design could reduce the network resources needed in service discovery.

The service discovery protocol proposed in this project has two key unique features. First, it can dynamically group nodes in a MANET into clusters for the purpose of caching service information. Dividing a large MANET into several smaller regions can effectively reduce the message overhead caused by service discovery messages. Second, it utilizes Distributed Hash Tables to efficiently cache service information in clusters. The use of DHT not only further reduces the message overhead, but also provides a definite guide which tells nodes where to search and cache their service information. The combination of these two mechanisms leads to high scalability and robustness making the proposed protocol suitable for ad hoc networks.

CHAPTER 3

3. PROPOSED METHODOLOGY

3.1. Proposed System

In a Mobile Ad-hoc Networks (MANET), every node can be mobile and act as a routing agent to forward packets on behalf of others. Two nodes can communicate with each other in a multi-hop manner through the help of other nodes. A MANET can be formed without any fixed infrastructure, and thus has low construction cost and can be built rapidly. MANETs are especially useful for disaster relief and military operations. Another practical application of MANET is the recently proposed Cooperative Network. A cooperative network is a collaboration of Wide Area Network (WAN) and MANET technologies. For instance, the users of a same 3G service provider in a same area can exchange data with each other through MANET, and thus ease the burden of base stations and the core network.

A critical challenge in designing MANETs is service discovery. In this project, we define the problem of service discovery as how a user can efficiently search and locate the service information desires in a MANET. Service information refers to any information that is used to describe, locate and access a given service. An efficient service discovery mechanism should have low network resource consumption, fast response time, and high query success rate.

Many service discovery mechanisms exist in today's Internet and some are widely used in our daily life. However, these mechanisms are not effective in handling the service discovery needs in MANETs where the networks are dynamically changing. This creates the need for new service discovery protocols for MANETs.

Peer-to-Peer (P2P) networks have drawn extensive attention in recent years because of their good performance and high scalability in data and

information dissemination. P2P networks and MANET share several common characteristics, such as unstable links between nodes and unexpected node arrivals and departures. Having these characteristics in common makes the collaboration of P2P networks and MANETs attractive. However, recent research works show that deploying P2P networks directly in MANETs will induce heavy message overhead. This is because P2P networks are overlay networks, and two peers in a P2P network may be many physical hops apart in the underlying physical network. Thus enhances are needed to make P2P mechanisms work efficiently in MANETs.

This paper presents a peer-to-peer service discovery protocol that is suitable to work in MANET environment. The design of our protocol follows the following guidelines:

- **Low message overhead**: Nodes in a MANET communicate with each other through wireless means. Comparing to wired networks, message overhead has greater influence on the overall performance of a MANET due to the nature of wireless communication. Thus a service discovery protocol should have low message overhead.
- **Scalability and robustness**: The number of nodes in a MANET can change dynamically, and sometimes frequently. The size of a MANET can also vary widely. Therefore, the service discovery protocol for MANETs should have stable performance in a dynamically changing network, and should also be scalable and robust.

3.2. The Mobile Service Discovery Protocol

In MANET, a subset of all the nodes is capable of providing one or more services; others may search and use any services which are not restricted to the services in the MANET. Services can be files, environmental information or

hardware devices such as printers. The goal of a service discovery protocol is to efficiently help users find the services they want in a given MANET. The proposed Mobile Service Discovery Protocol (MSDP) comprises two major parts, which are dynamic cluster formation and service discovery. The following two subsections will explain how they work.

3.2.1. Dynamic Cluster Formation

In the process of forming dynamic clusters, every node can be in one of the following states: Unknown, Member or Head. Every node is in the unknown state when it first enters into the network. Based on the cluster size all the nodes form clusters. Each node should be in one of the cluster. Duplication of nodes are not allowed i.e. each node should not be in more than one cluster.

Each cluster must have a head among the nodes. The node which is having more number of services should act as a head. This is because during the service discovery, the possibility of service being found in the head increases. Except head, all the nodes in the cluster changes into member state.

For each node there must be a time count. Once they enter into the network the timer starts. If the time expires, either the node is to be refreshed with fresh time count or it should be out of the network.

All nodes are in *Unknown* state at the beginning (Fig. 1 (a)). They exchange HMs and learn the existence of their neighbors. The timers of the nodes with local minimal *UID*s will expire first, which make these nodes change their states to *Head* (Fig. 1 (b)). When the *Unknown* nodes receive HMs from the *Head* nodes and learn that they are under the coverage of at least one *Head* nodes, they change their states to *Member* and become cluster members (Fig. 1 (c)). It is also possible that a node does not have local minimal *UID* nor is it under coverage of any *Head* nodes (Node 6 in Fig. 1); after the timer of this

node expires, it will also change its state to *Head* (Fig. 1 (d)). Finally, all nodes are either in *Head* or *Member* state and clusters are formed.

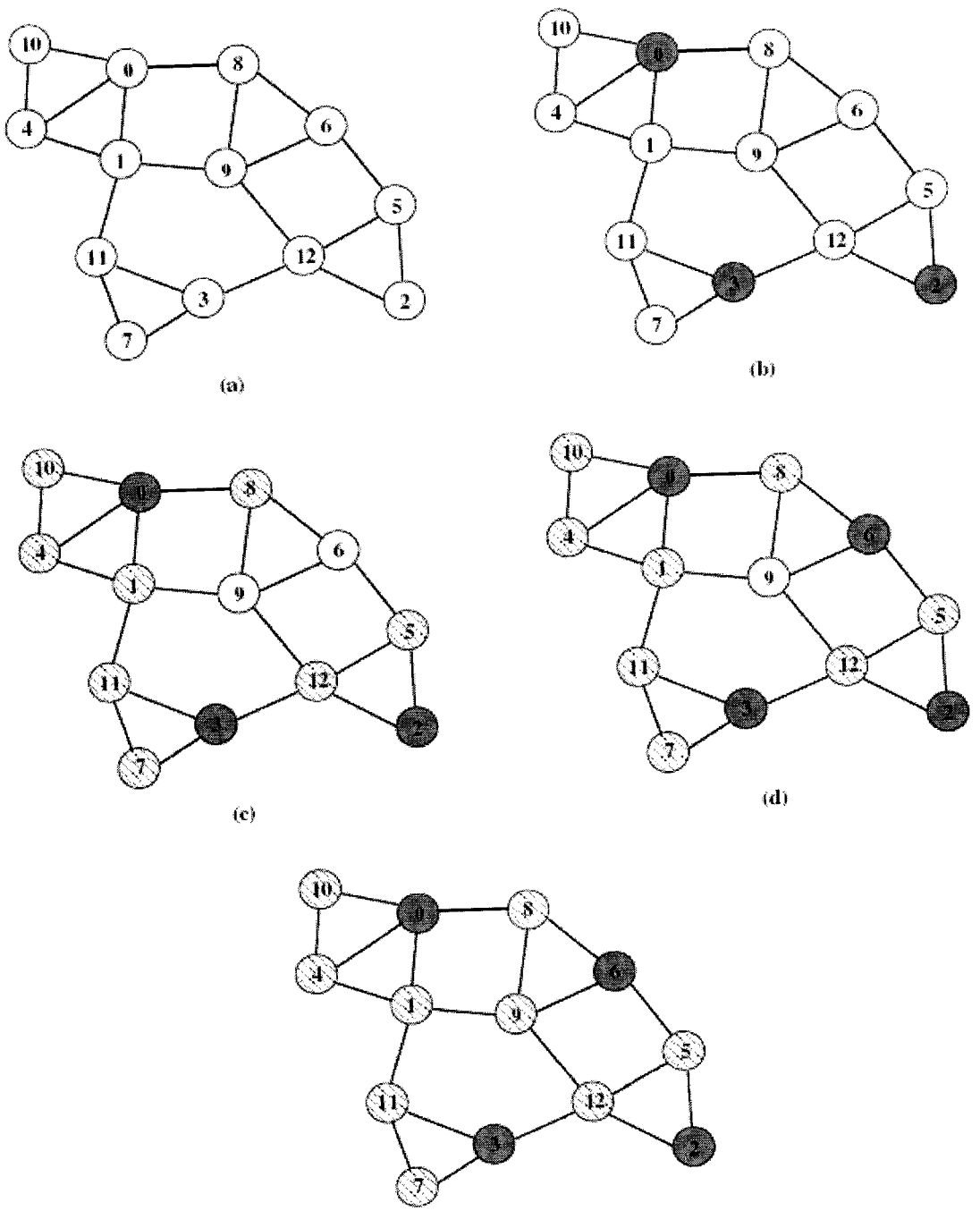


Figure 4.1. Dynamic Cluster Formation. White, stripped and grey represent Unknown, Member and Head state, respectively.

3.2.2. Service Discovery

Every head node treats all the member nodes within cluster size hops as its cluster member. The service description should be entered for each node during its creation. The service description denotes the services it can offer.

When a user wants to find a specific service, he packs the service description in a Service Request packet, and sends the packet to the head of the cluster he belongs to. When a head receives the Service Request, it sends the request to all its members. Finally, when the Member node receives the Service Request, it checks its own service database if the requested service is found, it sends back a Service Reply containing the service information. Otherwise, it sends back a Service Reply with no service information, indicating that no service information is found.

If the user receives a Service Reply which indicates that no service information is found in the local cluster, it sends out another Service Request which, thought the help of the local cluster head, is then broadcasted to all head nodes. Other head nodes that receive this Service Request then query all the Member nodes they have for the requested service information. Eventually, the user will receive a Service Reply if the requested service does exist in the MANET. After assuring the availability of the requested service, the user sends out a Service Publish packet containing the service information to his cluster head.

3.2.3. Distributed Hash Table

A distributed hash table is a system that is commonly used in peer-to-peer file sharing networks such as Napster. A DHT node is the mechanics that are used to find specific information shared on the network. When a user inputs specific search words such as a name, the DHT nodes are responsible for searching the database to find that information and bring it up as a search result.

In the MSDP, the SHA1 hash function is used to generate a 128-bit integer from a given service information string. We also use Consistent Hashing to map the integer generated by the SHA1 function to one of the keys in the *UID* key space. In consistent hashing, the keys in the key space are sorted in increasing order and form a circular linked-list, i.e. the last element of the sorted list links to the first element.

The distributed hash table structure is highly efficient, enabling messages to be routed within *log N* hops, where N is the number of nodes in the network. Messages are given an ID value based on their hash. This value is used in routing, with messages being sent to the resolver whose identifier is equal to or succeeds the message identifier. The messages move twice as close to the destination on each successive hop through the network.

Each node maintains a set of links to other nodes (its *neighbors* or routing table). Together these links form the overlay network. A node picks its neighbors according to a certain structure, called the network's topology.

Modular Diagram:

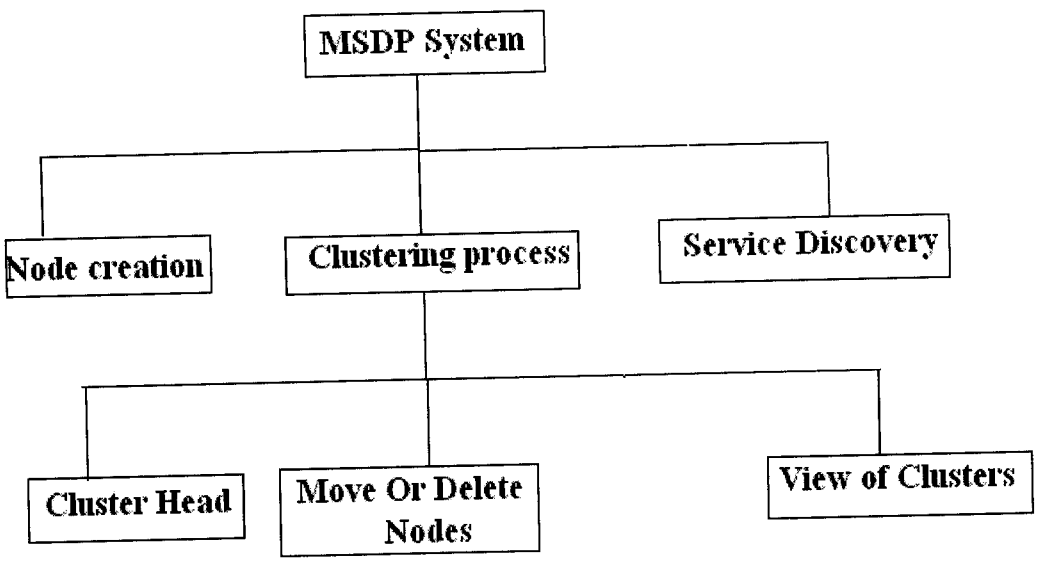


Figure 5.1. Module Diagram

3.2.4 Node Creation

In this module new nodes are to be created. During the node creation node name, service name, UID, Node type and Time count are to be got from the user. After the creation of node, the information should be added to the table.

3.2.5 Clustering Process

All the nodes which are created are clustered based on the cluster size. Cluster size should be given by the user. The Clustering process involves

1). Selection of Cluster Head.

2). Move or Delete Nodes.

3). View of Clusters.

Cluster Head

The nodes provide maximum number of services is selected as the cluster head. All the search process should be taken place through the cluster head.

Move or Delete nodes

After the creation of cluster, a node from one cluster can be moved to another cluster provided the cluster should have less number of nodes than the cluster size. Note that there should not be any duplication.

The nodes can also deleted from a cluster and the space can be reused.

View of cluster

Each cluster can be viewed graphically. The nodes can be represented as a rectangular boxes and the link between the nodes in a cluster can be represented by the lines. The details of each cluster can also be viewed here.

3.2.6 Service Discovery

Sometimes a service which is not found in a node can be needed by the node. In such cases that service can be offered by some other node. To discover the service, the node which needs the service can send the service request to the

head of its local cluster. The head sends the request to all its members. If the service is found in any of the member it replied to the head with the service information. If not found it replied that there is no service information. If the service is not found in the local cluster it sends message to all the cluster head to discover the service. If the service is found in any of the node then it replied with the service information.

CHAPTER 4

4. RESULT

Thus Mobile Service Discovery Protocol (MSDP) using dynamic clustering algorithm to group nodes in a MANET, and utilizes Distributed Hash Tables (DHTs) to efficiently cache service information in a peer-to-peer manner is achieved. The main goal of minimizing message overhead is obtained.

CHAPTER 5

5. CONCLUSION

In this project we propose the Mobile Service Discovery Protocol, a service discovery protocol which is well suitable for mobile ad hoc environment. The proposed MSDP combines dynamic cluster formation and distributed Hash tables, and can efficiently organize the MANET to provide service information caching. Service discovery messages can be resolved locally in clusters, and thus reduces the message overhead caused by the service discovery process. With extensive simulation experiments, we verified that the proposed MSDP has steady performance in terms of service information discovery and caching, and is able to minimize the network resources consumption.

CHAPTER 6

6. FUTURE WORK

In the future, we plan to extend our work to a cross-layer design that combines network routing protocol with service discovery protocol. We also plan to extend our proposed MSDP to work under other environment and on different hardware technologies.

CHAPTER 7

7. REFERENCE

[1] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin. GSD:A Novel Group-based Service Discovery Protocol for MANETs. In *Proc. of the 4th International Workshop onMobile and Wireless Communications Network*, pages 140–144, 2002.

[2] L. Cheng. Service Advertisement and Discovery in Mobile Ad Hoc Networks. In *Proc. of Computer Supported Cooperative Work*, 2004.

[3] D. Eastlake and P. Jones. US Secure Hash Algorithm 1(SHA1). Internet RFC 3174, Sept. 2001.

[4] S. Frattasi, F. F. B. Can, and R. Prasad. Cooperative Services for 4G. In *Proc. of the 14th IST Mobile and Wireless Communications*, 2005.

[5] S. Helal, N. Desai, V. Verma, and C. Lee. Konark – A Service Discovery and Delivery Protocol for Ad-Hoc Networks. In *Proc. of Wireless Communications and Networking*, volume 3, pages 2107–2113, New Orleans, LA, USA,
Mar. 2003.

[6] H. Y. Hsieh and R. Sivakumar. On Using Peer-to-Peer Communication in Cellular Wireless Data Networks. *IEEE Transaction on Mobile Computing*, 3:57–72, January-March 2004.

[7] A. Klemm, C. Lindemann, and O. P. Waldhorst. A Special- Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks. In *Proc. of the 54th IEEE Vehicular Technology Conference*, volume 4, pages 2758–2763, Orlando, FL, USA, 2003.

[8] U. C. Kozat and L. Tassiulas. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2003*, volume 3, pages 1965–1975, San Francisco, CA, USA, Apr. 2003.

[9] L. Yan. Performance Evaluation and Modeling of Peer-to-Peer Systems over Mobile Ad Hoc Networks. Technical Report 678, TUCS, Mar. 2005.

Web Site:

http://www.salutation.org

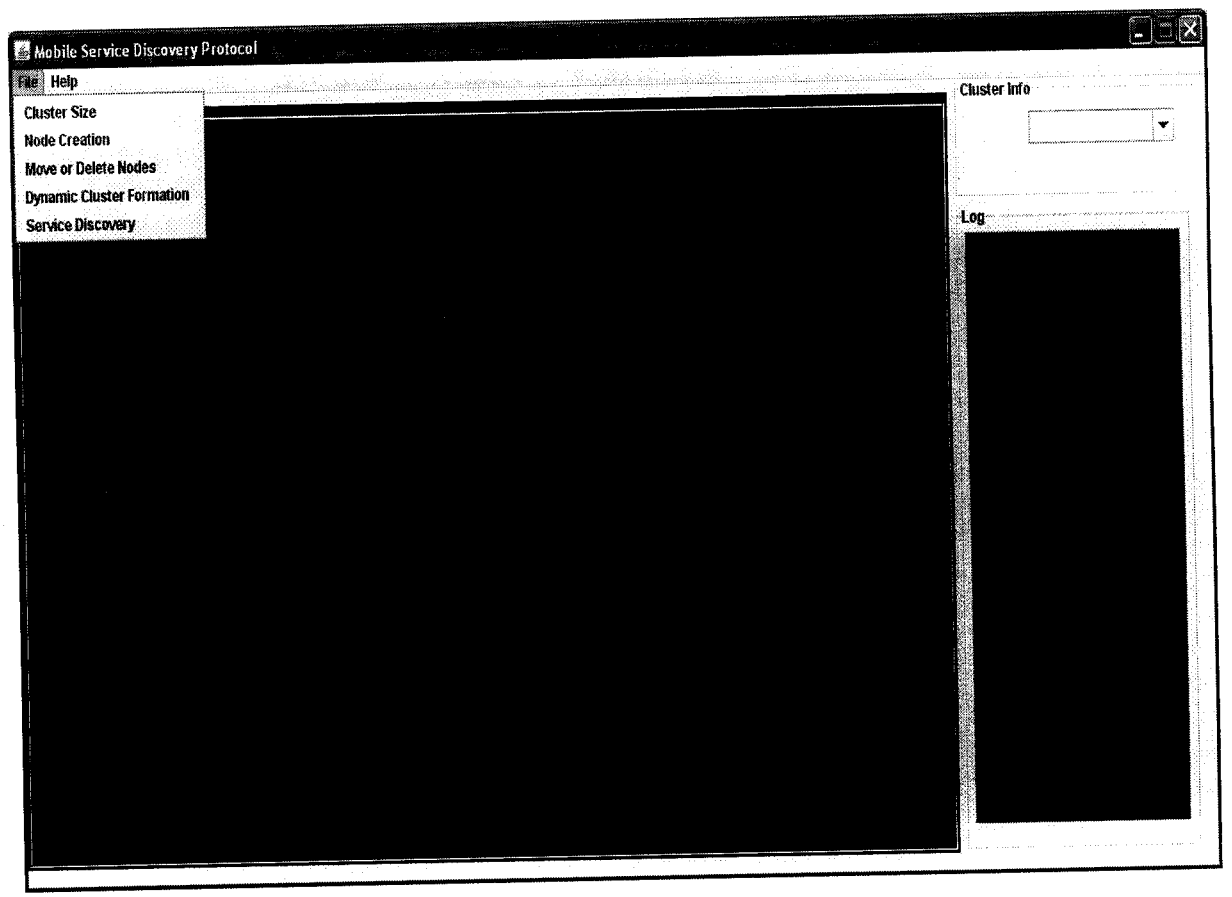
http://www.gnutella.com

http://www.napster.com

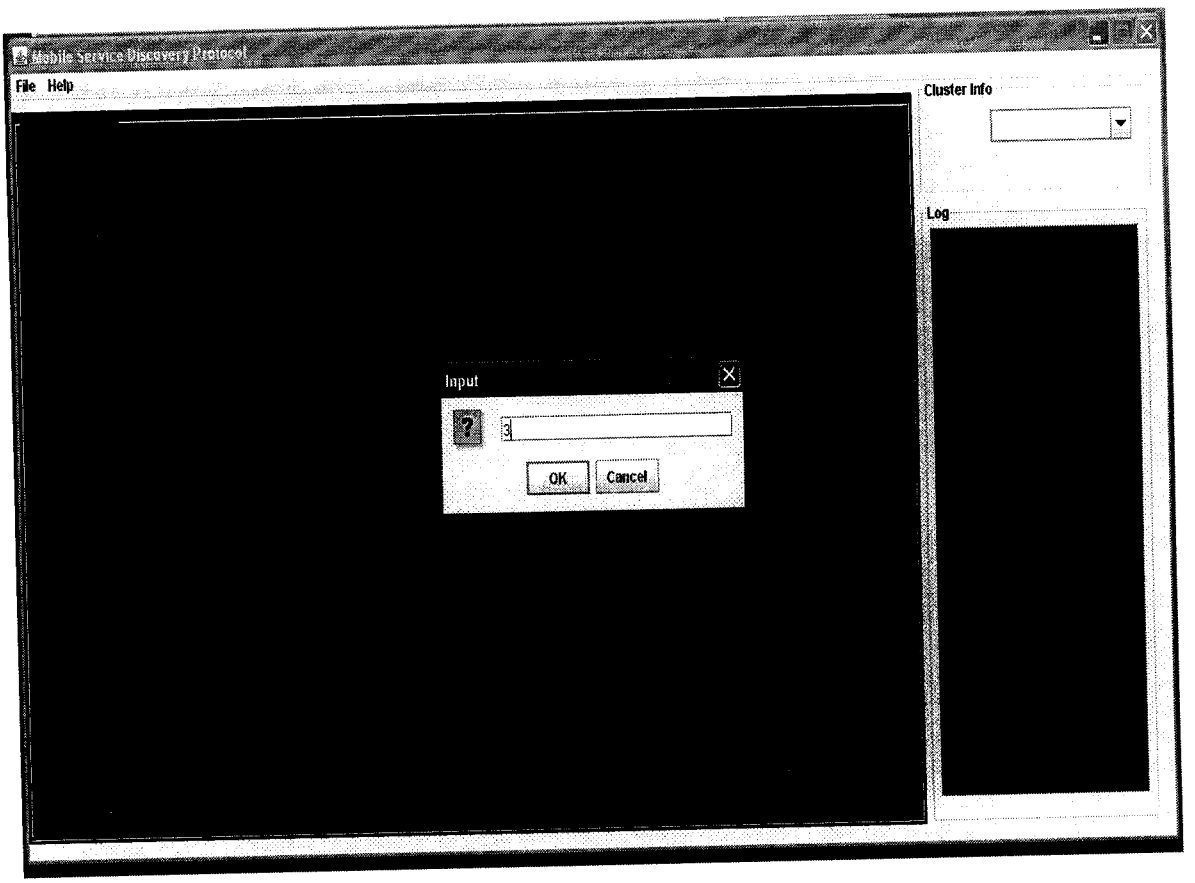
APPENDICES

A. SAMPLE SCREENS

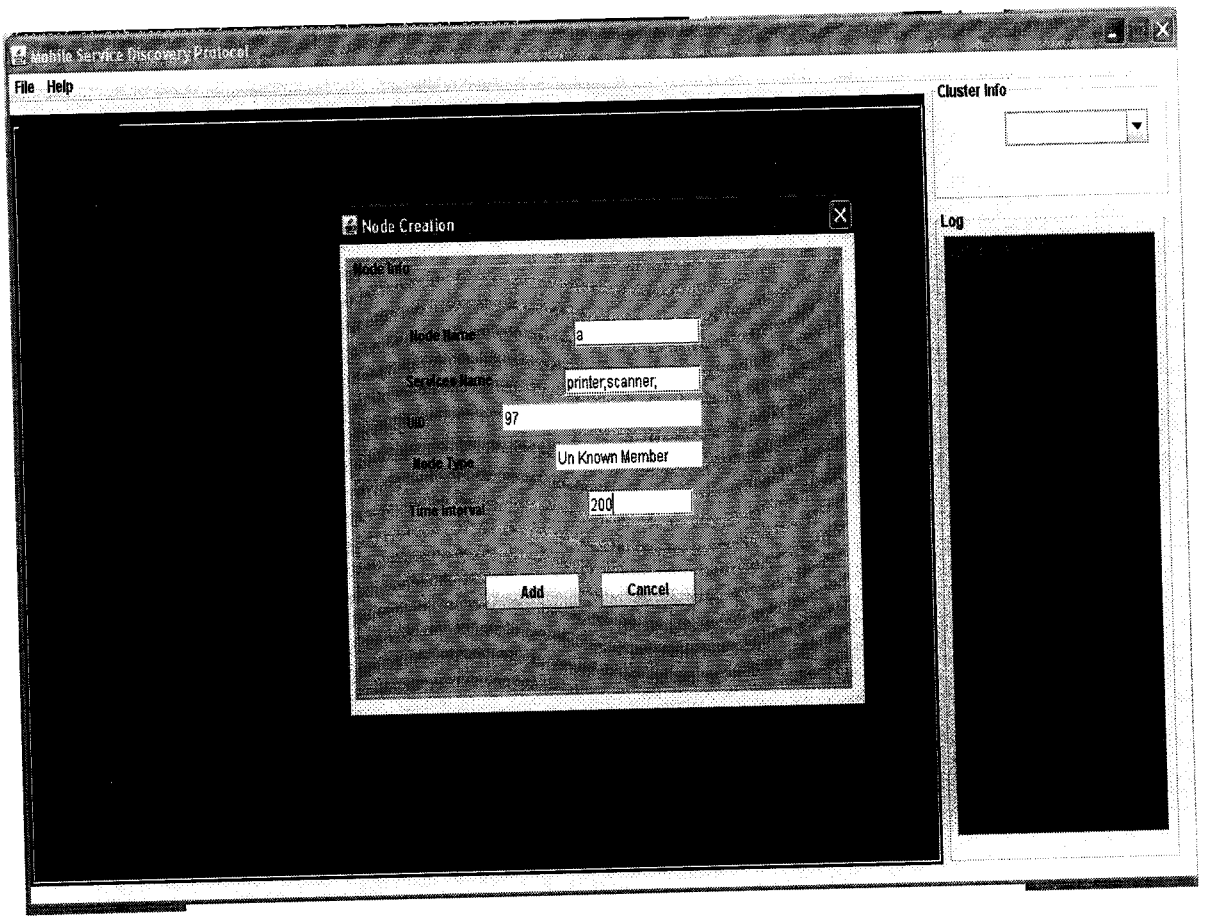
Home Screen:



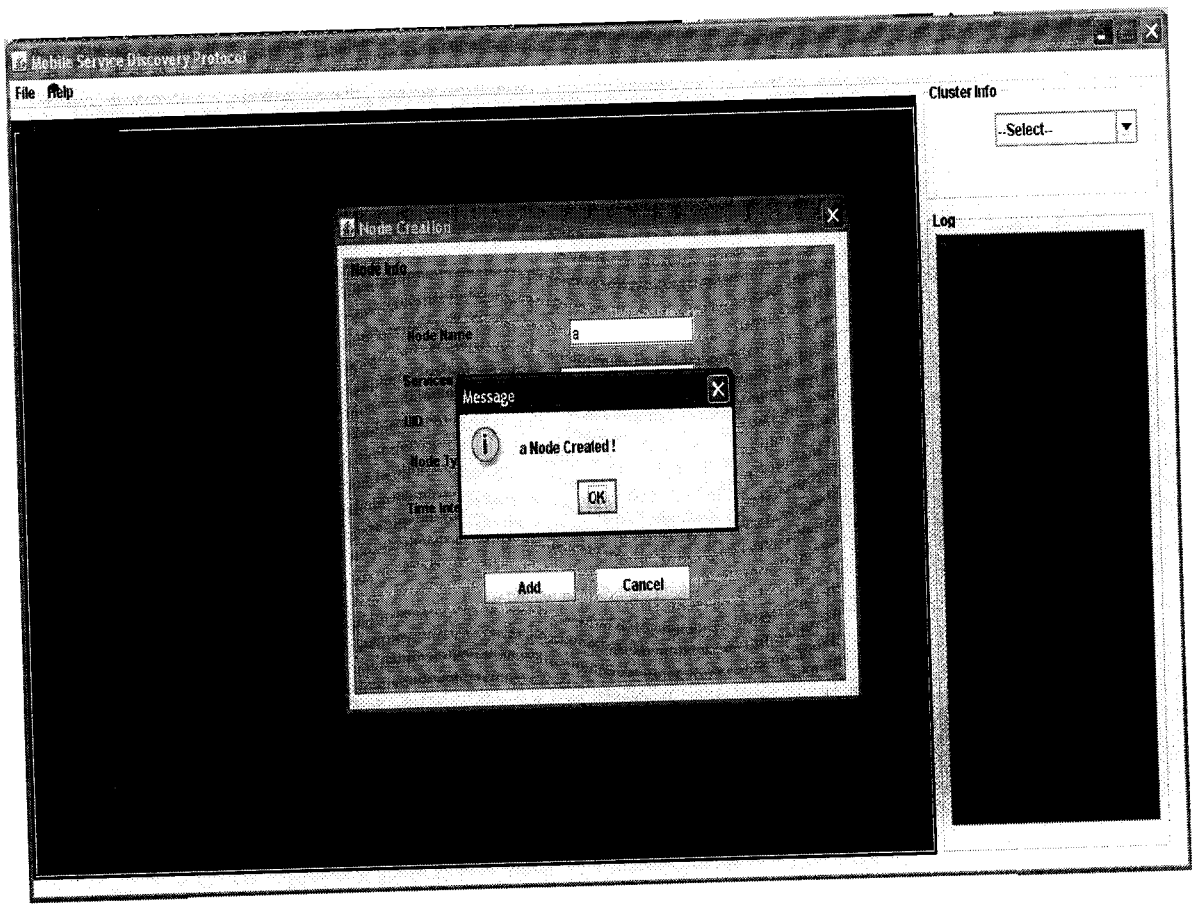
Cluster Size:



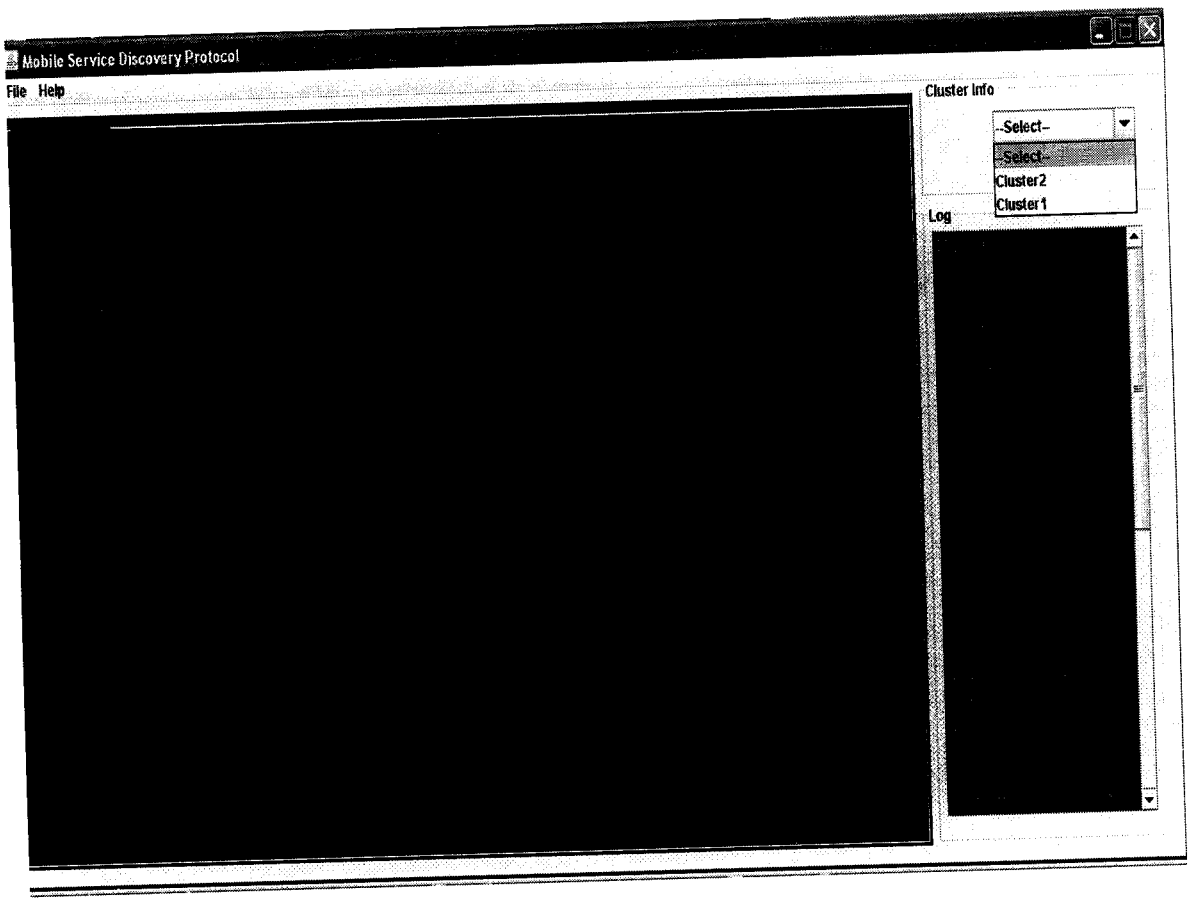
Node Details:



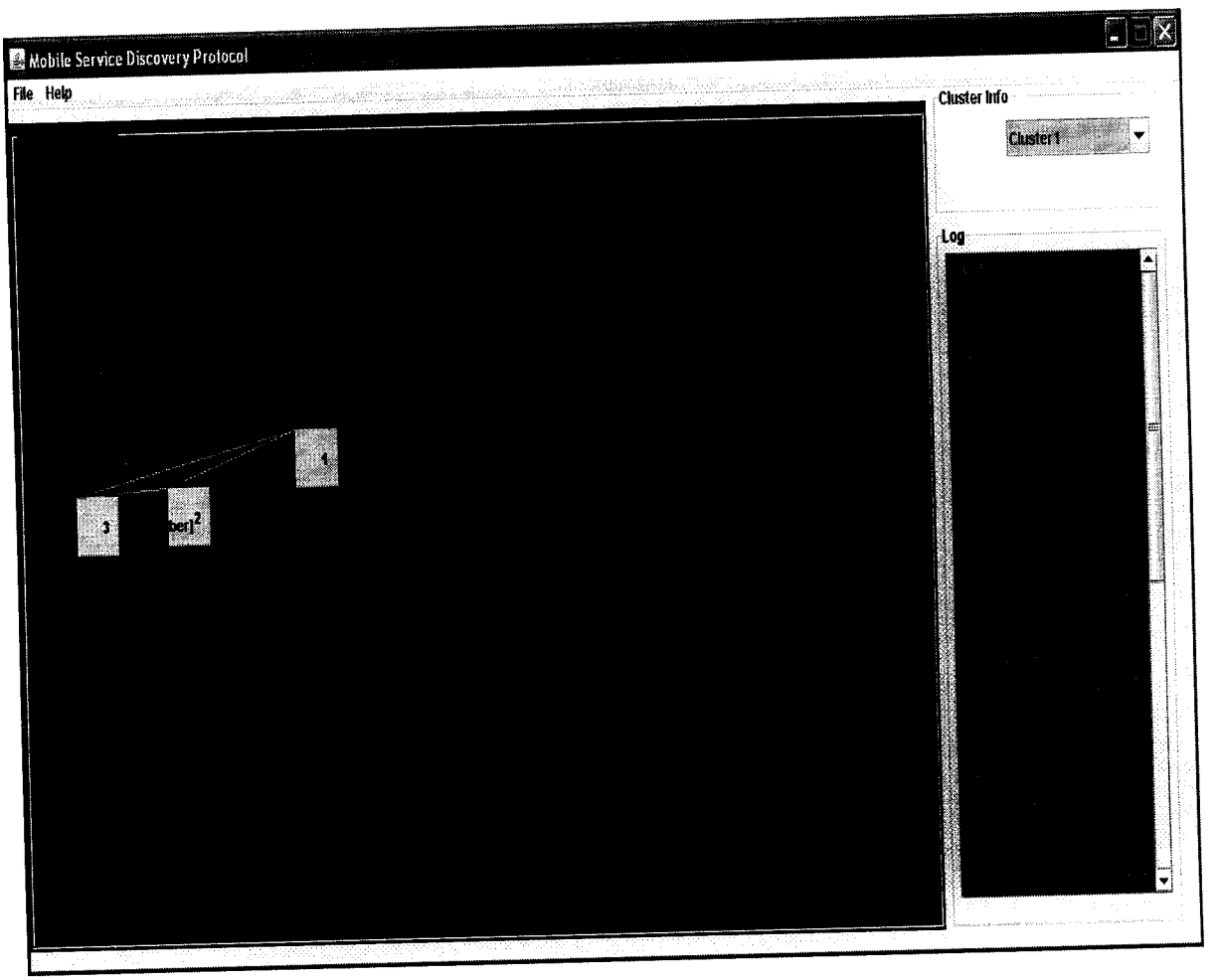
Node Creation:



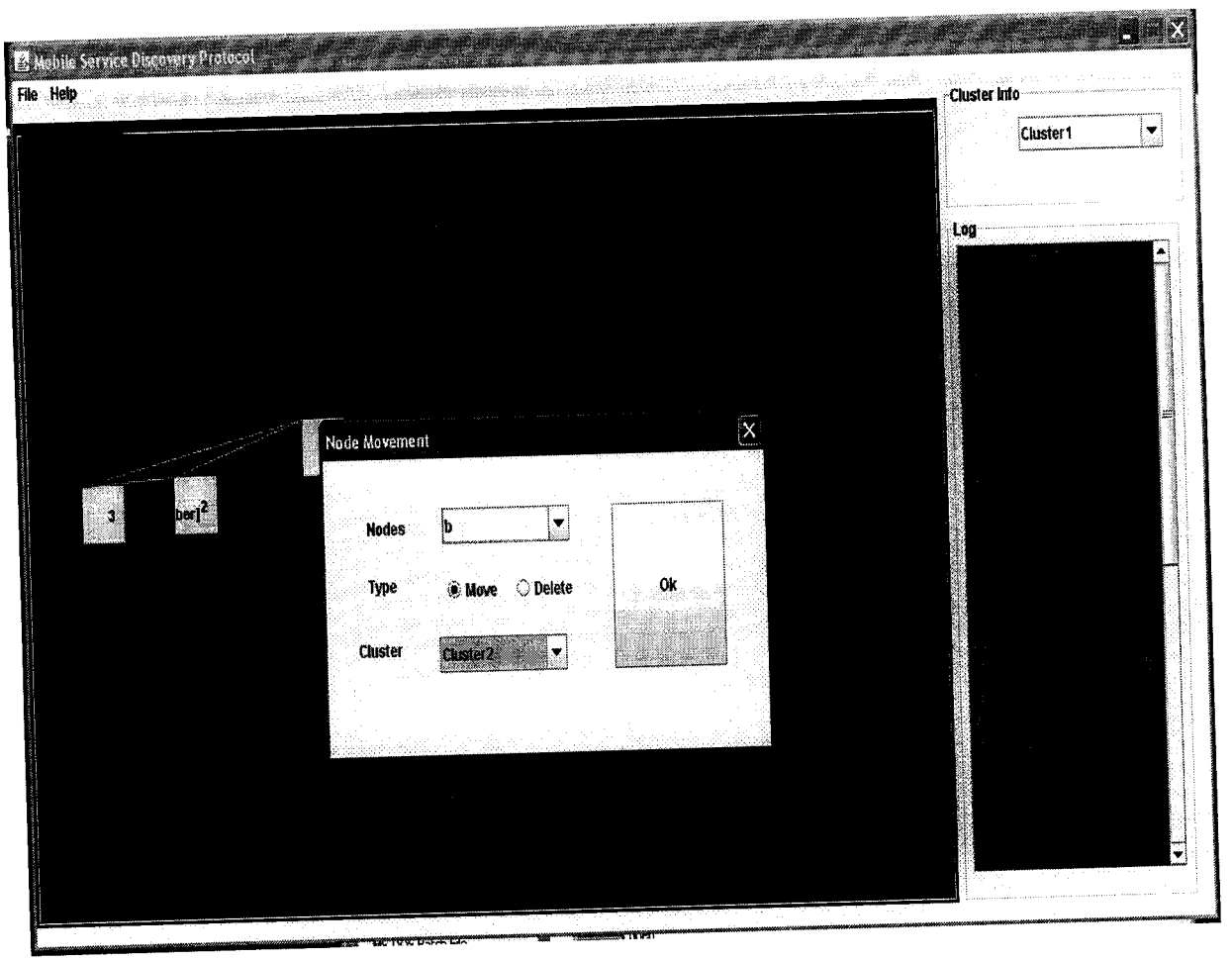
Log Details:


Mobile Service Discovery Protocol
File Help
Cluster Info
-Select-
-Select-
Cluster2
Cluster1
Log

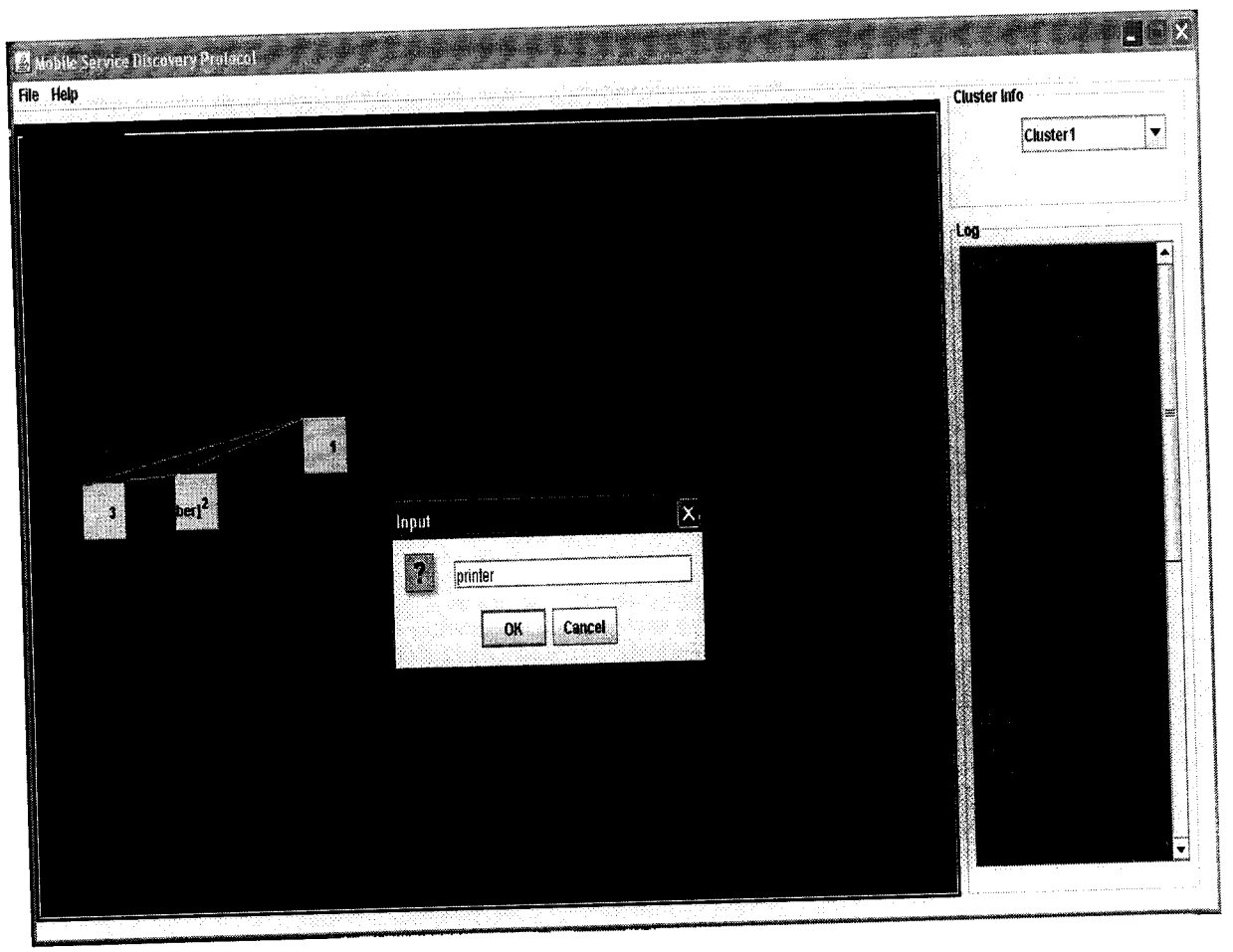
Cluster View:



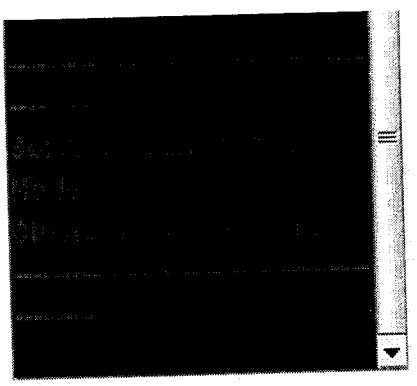
Move or Delete Nodes:



Getting Service To Discover:



Service Discovered:



B.SAMPLE CODE

MSDP:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Random;
import java.util.Set;
public class MSDP extends JFrame {
public ArrayList <Node> NodeList=new ArrayList();
public ArrayList <String> ClusterList=new ArrayList();
public HashMap <String,ArrayList> listHashMap=new HashMap();
public int totalCluster=0;
Random r=new Random();
public String currentCluster;
public static int clusterSize=0;
public boolean isPaint=false;
public MSDP() {
        initComponents();
                }
 private void panel1MouseClicked(MouseEvent e) {
 panel1.findOutDirection(new Point(e.getX(),e.getY())) ;
                                                }
 private void menuItem7ActionPerformed(ActionEvent e) {
 new NodeMove(new Frame(),this).setVisible(true);
          }
 private void menuItem1ActionPerformed(ActionEvent e) {
 //Create Node
  if(clusterSize!=0)
   {
           isPaint=false;
          new NodeCreation(new Frame(),this).setVisible(true);
   }            else {
     JOptionPane.showMessageDialog(null,"Please Define Cluster Size !");
    }
        }
 private void menuItem2ActionPerformed(ActionEvent e) {
 try {
      if(clusterSize!=0)
          {
             if(comboBox1.getSelectedItem().toString().equalsIgnoreCase("--Select--"))
              {
               JOptionPane.showMessageDialog(null,"Select Cluster Name!");
```

```
            }
          else {
             currentCluster=comboBox1.getSelectedItem().toString();
              isPaint=true;
              panel1.repaint();
           }
                }
        else {
           JOptionPane.showMessageDialog(null,"Please Define Cluster Size !");
        }
             } catch (Exception ex)  {
                ex.printStackTrace();
             }
     }
private void menuItem6ActionPerformed(ActionEvent e) {
      try  {
         if(clusterSize!=0)
         {
         findHead();
         isPaint=true;
         panel1.repaint();
         }            else {
            JOptionPane.showMessageDialog(null,"Please Define Cluster Size !");
          }

       } catch (Exception ex)  {
          ex.printStackTrace();
       }
 }
    private void menuItem4ActionPerformed(ActionEvent e) {
              try  {
                 String val=   JOptionPane.showInputDialog(null);
            FindServices(val);
              } catch (Exception ex)  {
                 ex.printStackTrace();
              }
              }
        private void menuItem5ActionPerformed(ActionEvent e) {
           isPaint=false;
            try  {
                  while(true)
             {
             String val=   JOptionPane.showInputDialog(null);
               if(val!=null) {
                  if(!CommonFunction.isNumber(val)) {
                     JOptionPane.showMessageDialog(null,"Please enter the  Number!");
                  }
                  else {
                     clusterSize=Integer.parseInt(val);
                     break;
```

```
            }
                    }
            else {
              break;
            }
          }
            System.out.println("Cluster Size "+clusterSize);
            textArea1.append("Cluster Size "+clusterSize+"\n");
              } catch (Exception ex)  {
                 ex.printStackTrace();
              }
               }
    private void initComponents() {
    menuBar1 = new JMenuBar();
    menu1 = new JMenu();
    menuItem1 = new JMenuItem();
    menuItem2 = new JMenuItem();
    menuItem4 = new JMenuItem();
    menuItem5= new JMenuItem();
    menuItem7= new JMenuItem();
    menu2 = new JMenu();
    menuItem3 = new JMenuItem();
    panel1 = new NodePanel (this);
    panel2 = new JPanel();
    comboBox1 = new JComboBox();
    panel3 = new JPanel();
    scrollPane1 = new JScrollPane();
    textArea1 = new JTextArea();
            //======== this ========
            setTitle("Mobile Service Discovery Protocol ");
            setResizable(true);
            setBackground(new Color(185, 178, 136));
            Container contentPane = getContentPane();
            contentPane.setLayout(null);
            //======== menuBar1 ========
            {
                    //======== menu1 ========
                    {
                            menu1.setText("File");
                      //---- menuItem1 ----
                      menuItem5.setText("Cluster Size");
                      menuItem5.addActionListener(new ActionListener() {
                            public void actionPerformed(ActionEvent e) {
                                menuItem5ActionPerformed(e);
                            }
                      });
                      menu1.add(menuItem5);
                            //---- menuItem1 ----
                            menuItem1.setText("Node Creation");
                            menuItem1.addActionListener(new ActionListener() {
```

```
            public void actionPerformed(ActionEvent e) {
            menuItem1ActionPerformed(e);
            }
            });
            menu1.add(menuItem1);
        //---- menuItem1 ----
        menuItem7.setText("Move or Delete Nodes ");
        menuItem7.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                menuItem7ActionPerformed(e);
            }
        });
        menu1.add(menuItem7);
            //---- menuItem2 ----
            menuItem2.setText("Dynamic Cluster Formation");
            menuItem2.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                            menuItem2ActionPerformed(e);
                    }
            });
            menu1.add(menuItem2);
            //---- menuItem4 ----
            menuItem4.setText("Service Discovery");
            menuItem4.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                            menuItem4ActionPerformed(e);
                    }
            });
            menu1.add(menuItem4);
        }
        menuBar1.add(menu1);
        //======== menu2 ========
        /* {
                menu2.setText("Help");

                //---- menuItem3 ----
                menuItem3.setText("About");
                menuItem3.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                menuItem3ActionPerformed(e);
                        }
                });
                menu2.add(menuItem3);
        }
        menuBar1.add(menu2);*/
}
comboBox1.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                        //      comboBox1ActionPerformed(e);
                        }
```

```
                });
comboBox1.addItem("--Select--");
        contentPane.add(menuBar1);
        menuBar1.setBounds(0, 0, 1175, 21);
        //======== panel1 ========
        {
    panel1.setBorder(new TitledBorder(new  EtchedBorder(EtchedBorder.RAISED,
    Color.blue, Color.white), "Ad-Hoc NetWork"));
            panel1.setBackground(Color.black);
            panel1.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    panel1MouseClicked(e);
                }
            });
            panel1.setLayout(null);
            { // compute preferred size
                Dimension preferredSize = new Dimension();
                for(int i = 0; i < panel1.getComponentCount(); i++) {
                Rectangle bounds = panel1.getComponent(i).getBounds();
                    preferredSize.width = Math.max(bounds.x +
                bounds.width, preferredSize.width);
                    preferredSize.height = Math.max(bounds.y +
                bounds.height, preferredSize.height);
                }
                Insets insets = panel1.getInsets();
                preferredSize.width += insets.right;
                preferredSize.height += insets.bottom;
                panel1.setMinimumSize(preferredSize);
                panel1.setPreferredSize(preferredSize);
            }
        }
        contentPane.add(panel1);
        panel1.setBounds(0, 30, 865, 565);
        //======== panel2 ========
        {
          panel2.setBorder(new TitledBorder(null, "Cluster Info",
          TitledBorder.LEADING, TitledBorder.TOP, null, new Color(0, 0, 204)));
              panel2.setLayout(null);
              panel2.add(comboBox1);
            comboBox1.setBounds(70, 25, 135,
            comboBox1.getPreferredSize().height);
              { // compute preferred size
                    Dimension preferredSize = new Dimension();
                    for(int i = 0; i < panel2.getComponentCount(); i++) {
                    Rectangle bounds = panel2.getComponent(i).getBounds();
                    preferredSize.width = Math.max(bounds.x + bounds.width,
                       preferredSize.width);
                    preferredSize.height = Math.max(bounds.y + bounds.height,
                  preferredSize.height);
```

```
                }
                Insets insets = panel2.getInsets();
                preferredSize.width += insets.right;
                preferredSize.height += insets.bottom;
                panel2.setMinimumSize(preferredSize);
                panel2.setPreferredSize(preferredSize);
        }
}
contentPane.add(panel2);
panel2.setBounds(870, 20, 225, 90);
//======== panel3 ========
{
panel3.setBorder(new TitledBorder(null, "Log", TitledBorder.LEADING,
   TitledBorder.TOP, null, new Color(51, 0, 153)));
        panel3.setLayout(null);
        //======== scrollPane1 ========
        {
                //---- textArea1 ----
                textArea1.setFont(new Font("Dialog", Font.BOLD, 14));
                textArea1.setForeground(new Color(7, 148, 46));
                textArea1.setWrapStyleWord(true);
                textArea1.setBackground(Color.black);
                textArea1.setLineWrap(true);
                scrollPane1.setViewportView(textArea1);
        }
        panel3.add(scrollPane1);
        scrollPane1.setBounds(10, 20, 200, 440);
        { // compute preferred size
                Dimension preferredSize = new Dimension();
                for(int i = 0; i < panel3.getComponentCount(); i++) {
                Rectangle bounds = panel3.getComponent(i).getBounds();
                 preferredSize.width = Math.max(bounds.x + bounds.width,
                 preferredSize.width);
                preferredSize.height = Math.max(bounds.y + bounds.height,
                preferredSize.height);
                 }
                 Insets insets = panel3.getInsets();
                 preferredSize.width += insets.right;
                 preferredSize.height += insets.bottom;
                 panel3.setMinimumSize(preferredSize);
                 panel3.setPreferredSize(preferredSize);
        }
}
contentPane.add(panel3);
panel3.setBounds(870, 115, 220, 480);
{ // compute preferred size
        Dimension preferredSize = new Dimension();
        for(int i = 0; i < contentPane.getComponentCount(); i++) {
        Rectangle bounds = contentPane.getComponent(i).getBounds();
```

```
                    preferredSize.width = Math.max(bounds.x + bounds.width,
                    preferredSize.width);
                    preferredSize.height = Math.max(bounds.y + bounds.height,
                    preferredSize.height);
                    }
                    Insets insets = contentPane.getInsets();
                    preferredSize.width += insets.right;
                    preferredSize.height += insets.bottom;
                    contentPane.setMinimumSize(preferredSize);
                    contentPane.setPreferredSize(preferredSize);
                }
                setSize(1110, 640);
                setLocationRelativeTo(getOwner());
                // JFormDesigner - End of component initialization  //GEN-
END:initComponents
        }
 public static void main(String[] args) {
    try {
      new  MSDP().setVisible(true);
    } catch (Exception ex)  {
      ex.printStackTrace();
    }
     }

 public void AddNode(Node node) {
     try  {
         boolean isNodeAdded=false;
         if(totalCluster==0) {
           totalCluster++;
         ArrayList<Node> list= new ArrayList();
         node.setClusterName("Cluster"+totalCluster);
           list.add(node);
           listHashMap.put("Cluster"+totalCluster,list);
           isNodeAdded=true;
         }
       //Create the sheet
       // listHashMap.s
         if(!isNodeAdded) {
           Set key= listHashMap.entrySet();
                 Iterator i = key.iterator();
                 // Display elements
                 while(i.hasNext()) {
                   Map.Entry me = (Map.Entry)i.next();
                       ArrayList<Node> list=listHashMap.get(me.getKey());
                        if(list.size()<clusterSize)  {
                          node.setClusterName("Cluster"+totalCluster);
                           list.add(node);
                           isNodeAdded=true;
                           break;
                        }
```

```
                }
                if(!isNodeAdded) {
                   totalCluster++;
                   ArrayList<Node> list= new ArrayList();
                   node.setClusterName("Cluster"+totalCluster);
                   list.add(node);
                   listHashMap.put("Cluster"+totalCluster,list);
                   isNodeAdded=true;
                }
        }

    /*    Load Combo
     */
      findHead();
    } catch (Exception ex)  {
       ex.printStackTrace();
    }
}
    public void MoveNode(Node node,String clusterName) {
        try  {
           ArrayList<Node> moveClusterlist=listHashMap.get(clusterName);
           if(moveClusterlist.size()==clusterSize ) {
              JOptionPane.showMessageDialog(null,clusterName+" is Full!");
           }
           else {
              ArrayList<Node> currClusterlist=listHashMap.get(node.getClusterName());
              for(int i=0;i<currClusterlist.size();i++) {
               Node n=  currClusterlist.get(i);
               if(n.getNodeName().equalsIgnoreCase(node.getNodeName())) {
                  currClusterlist.remove(i);
                  listHashMap.put(node.getClusterName(),currClusterlist);
                  node.setClusterName(clusterName);
                  moveClusterlist.add(node);
                  listHashMap.put(clusterName,moveClusterlist);
                  break;
               }
              }
           }
           findHead();
        } catch (Exception ex)  {
           ex.printStackTrace();
        }
     }
     public void DeleteNode(Node node,String clusterName) {
        try  {

           {
              ArrayList<Node> currClusterlist=listHashMap.get(node.getClusterName());
              for(int i=0;i<currClusterlist.size();i++) {
               Node n=  currClusterlist.get(i);
```

```
            if(n.getNodeName().equalsIgnoreCase(node.getNodeName())) {
              currClusterlist.remove(i);
              listHashMap.put(node.getClusterName(),currClusterlist);
              break;
            }
           }
         }
         findHead();
       } catch (Exception ex)  {
         ex.printStackTrace();
       }
}
public void findHead()
{
     try  {
       int headCount=0;
       int headSize=0;
        Set key= listHashMap.entrySet();
              Iterator i = key.iterator();
              // Display elements
              while(i.hasNext()) {
                 headCount=0;
                 headSize=0;
                 Map.Entry me = (Map.Entry)i.next();
                 ArrayList<Node> list=listHashMap.get(me.getKey());
                 if(list!=null)
                 {
                 for(int j=0;j<list.size();j++) {
                  Node n=  list.get(j);
                  String []ser=n.getServiceName().split(";");
                  if(ser.length>headSize) {
                    headSize= ser.length;
                    n.setNodeType("Member");;
                    headCount=j;
                  }
                  else {
                     n.setNodeType("Member");;
                  }
                 }
                 if(list.size()>0)
                 {
                 Node n=  list.get(headCount);
                 n.setNodeType("Head");
                 }
                 }

              }

         key= listHashMap.entrySet();
              i = key.iterator();
```

```
        comboBox1.removeAllItems();
        comboBox1.addItem("--Select--");
             // Display elements
             while(i.hasNext()) {
                Map.Entry me = (Map.Entry)i.next();
                comboBox1.addItem(me.getKey());
             }
             } catch (Exception ex)  {
        ex.printStackTrace();
    }
}
public void FindServices(String service)
{
    ArrayList keylist=new ArrayList();
    try  {
     if(comboBox1.getSelectedItem()==null) {
        JOptionPane.showMessageDialog(null,"Please Create  Cluster !");
     }
      else if(comboBox1.getSelectedItem().toString().trim().equals("")) {
         JOptionPane.showMessageDialog(null,"Please Create  Cluster !");
      }
      else if(comboBox1.getSelectedItem().toString().equals("--Select--")) {
        JOptionPane.showMessageDialog(null,"Please Select Cluster !");
      }
     else
     {
       int headCount=0;
       int headSize=0;
       boolean found=false;
        Set key= listHashMap.entrySet();
              Iterator i = key.iterator();
              // Display elements
               keylist.add(comboBox1.getSelectedItem().toString());
              while(i.hasNext()) {
                                  Map.Entry me = (Map.Entry)i.next();

if(!comboBox1.getSelectedItem().toString().equalsIgnoreCase(me.getKey().toString()))
                  {
                   keylist.add(me.getKey());
                  }
                }
             for(int keys=0;keys< keylist.size();keys++)
             {
             ArrayList<Node> list=listHashMap.get(keylist.get(keys).toString());
                    found=false;
                    for(int j=0;j<list.size();j++) {
                      Node n=  list.get(j);
                      String []ser=n.getServiceName().split(";");
                      for(int k=0;k<ser.length;k++) {
                         if(ser[k].equalsIgnoreCase(service)) {
```

```
                    JOptionPane.showMessageDialog(null,"Service Found");
                      textArea1.append("      ********************** \n");
                      textArea1.append("Service Found in the Node
                      "+n.getNodeName()+"\n");
                      textArea1.append("Cluster Name "+n.getClusterName()+"\n");
                      textArea1.append("      ********************** \n");
                      found=true;
                    }
                  }
                }
                if(!found) {
                  textArea1.append("Service Not Found  in "+keylist.get(keys)+" \n");
                }
              }
            }
        } catch (Exception ex)  {
          ex.printStackTrace();
        }
    }
        private JMenuBar menuBar1;
        private JMenu menu1;
        private JMenuItem menuItem1;
        private JMenuItem menuItem2;
        private JMenuItem menuItem4;
        private JMenuItem menuItem5;
        private JMenuItem menuItem7;
        private JMenu menu2;
        private JMenuItem menuItem3;
        private NodePanel panel1;
        private JPanel panel2;
        private JComboBox comboBox1;
        private JPanel panel3;
        private JScrollPane scrollPane1;
        public  JTextArea textArea1;

}
```

Node Creation:

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.*;
import javax.swing.border.*;
public class NodeCreation extends JDialog {
   MSDP objMSDP=null;
   Random r=new Random();
        public NodeCreation(Frame owner) {
                super(owner);
```

```
            initComponents();
    }
    public NodeCreation(Frame owner,MSDP objMSDP) {
    super(owner);
    this.objMSDP=objMSDP;
    initComponents();
}
    public NodeCreation(Dialog owner) {
            super(owner);
            initComponents();
    }
    private void textField2FocusLost(FocusEvent e) {
    getUID();      }
public void getUID() {
  try {
    textField3.setText(""+textField1.getText().toString().hashCode());
    textField4.setText("Un Known Member");
  } catch (Exception ex) {
    ex.printStackTrace();
  }
}
 public Node getServices(Node n) {
  try {
    ArrayList<String> list=new ArrayList();
    String services=n.getServiceName();
    String[] temp=services.split(";");
          if(temp==null) {
      list.add(services);
    }
    for(int i=0;i<temp.length;i++) {
      list.add(temp[i]);
    }
    n.setServiceList(list);
          return n;
   } catch (Exception ex) {
    ex.printStackTrace();
   }
   return null;
  }
      private void button1ActionPerformed(ActionEvent e) {
        try {
          if(!CommonFunction.isNumber(textField5.getText()) ) {
          JOptionPane.showMessageDialog(null,"Please enter the  Number!");
        }
        if(textField2.getText().equalsIgnoreCase("")) {
          JOptionPane.showMessageDialog(null,"Please enter the the Number!");
          return;
        }
        else
        {
```

```
Node n=new Node();
n.setNodeName(textField1.getText() );
n.setServiceName(textField2.getText() );
n.setNodeUID(textField3.getText() );
    n.setNodeType(textField4.getText() );
    n.setTimeInterVal(Integer.parseInt(textField5.getText()) );
 int x=r.nextInt(300)+r.nextInt(100)+50;
 int y=r.nextInt(300)+r.nextInt(100);
 Rectangle rect=new Rectangle(x,y,40,40);
 n.setRect(rect);
   n.setStartPoint(new Point(x,y));
  n= getServices(n);
 objMSDP.NodeList.add(n);
   objMSDP.AddNode(n);
 JOptionPane.showMessageDialog(null,textField1.getText()+" Node Created ! ");
 objMSDP.textArea1.append("      ********************** \n");
 objMSDP.textArea1.append(textField1.getText()+" Node Created ! \n");
 objMSDP.textArea1.append("Services Name "+textField2.getText()+" \n");
 objMSDP.textArea1.append("UID  "+textField3.getText()+" \n");
 objMSDP.textArea1.append("Node Type "+textField4.getText()+" \n");
 objMSDP.textArea1.append("Node Interval "+textField5.getText()+" \n");
 objMSDP.textArea1.append("      ********************** \n");
 }
  } catch (Exception ex) {
    ex.printStackTrace();
  }
 }
private void button2ActionPerformed(ActionEvent e) {
 try {
  textField1.setText("");
  textField2.setText("");
  textField3.setText("");
  textField4.setText("");
  textField5.setText("");
} catch (Exception ex) {
  ex.printStackTrace();
}
}
private void initComponents() {
        // Component initialization
        panel1 = new JPanel();
        label2 = new JLabel();
        label1 = new JLabel();
        textField1 = new JTextField();
        textField2 = new JTextField();
        label3 = new JLabel();
        textField3 = new JTextField();
        textField4 = new JTextField();
        label4 = new JLabel();
        label5 = new JLabel();
```

```
textField5 = new JTextField();
button1 = new JButton();
button2 = new JButton();
//======== this ========
setTitle("Node Creation");
setBackground(Color.lightGray);
Container contentPane = getContentPane();
contentPane.setLayout(null);
//======== panel1 ========
{
        panel1.setBorder(new TitledBorder("Node Info"));
        panel1.setBackground(new Color(185, 178, 136));
        panel1.setLayout(null);
        //---- label2 ----
        label2.setText("Services Name ");
        label2.setFont(new Font("Arial", Font.BOLD, 11));
        panel1.add(label2);
        label2.setBounds(55, 85, 105, label2.getPreferredSize().height);
        //---- label1 ----
        label1.setText("Node Name");
        label1.setFont(new Font("Arial", Font.BOLD, 11));
        panel1.add(label1);
        label1.setBounds(60, 50, 85, 19);
        panel1.add(textField1);
        textField1.setBounds(215, 50, 120,
        textField1.getPreferredSize().height);
        //---- textField2 ----
        textField2.addFocusListener(new FocusAdapter() {
                @Override
                public void focusLost(FocusEvent e) {
                        textField2FocusLost(e);
                }
        });
        panel1.add(textField2);
        textField2.setBounds(205, 85, 130, 20);
        //---- label3 ----
        label3.setText("UID");
        label3.setFont(new Font("Arial", Font.BOLD, 11));
        panel1.add(label3);
        label3.setBounds(55, 115, 65, label3.getPreferredSize().height);
        //---- textField3 ----
        textField3.setEditable(false);
        textField3.setBackground(Color.white);
        panel1.add(textField3);
        textField3.setBounds(145, 110, 190,
        textField3.getPreferredSize().height);
        //---- textField4 ----
        textField4.setEditable(false);
        textField4.setBackground(Color.white);
        panel1.add(textField4);
```

```
    textField4.setBounds(195, 140, 140,
    textField4.getPreferredSize().height);
     //---- label4 ----
     label4.setText("Node Type");
     label4.setFont(new Font("Arial", Font.BOLD, 11));
     panel1.add(label4);
     label4.setBounds(60, 145, 70, label4.getPreferredSize().height);
     //---- label5 ----
     label5.setText("Time Interval");
     label5.setFont(new Font("Arial", Font.BOLD, 11));
     panel1.add(label5);
     label5.setBounds(55, 180, 90, label5.getPreferredSize().height);
     panel1.add(textField5);
     textField5.setBounds(225, 175, 100, 20);
     //---- button1 ----
     button1.setText("Add");
     button1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  button1ActionPerformed(e);
            }
     });
     panel1.add(button1);
     button1.setBounds(125, 235, 90, button1.getPreferredSize().height);
     //---- button2 ----
     button2.setText("Cancel");
     button2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  button2ActionPerformed(e);
            }
     });
     panel1.add(button2);
     button2.setBounds(235, 235, 90, button2.getPreferredSize().height);
     { // compute preferred size
            Dimension preferredSize = new Dimension();
            for(int i = 0; i < panel1.getComponentCount(); i++) {
            Rectangle bounds = panel1.getComponent(i).getBounds();
            preferredSize.width = Math.max(bounds.x + bounds.width,
            preferredSize.width);
             preferredSize.height = Math.max(bounds.y + bounds.height,
             preferredSize.height);
            }
            Insets insets = panel1.getInsets();
            preferredSize.width += insets.right;
            preferredSize.height += insets.bottom;
            panel1.setMinimumSize(preferredSize);
            panel1.setPreferredSize(preferredSize);
      }
}
contentPane.add(panel1);
panel1.setBounds(5, 10, 470, 325);
```

```
        { // compute preferred size
                Dimension preferredSize = new Dimension();
                for(int i = 0; i < contentPane.getComponentCount(); i++) {
                Rectangle bounds = contentPane.getComponent(i).getBounds();
                preferredSize.width = Math.max(bounds.x + bounds.width,
                preferredSize.width);
                preferredSize.height = Math.max(bounds.y + bounds.height,
                preferredSize.height);
                }
                Insets insets = contentPane.getInsets();
                preferredSize.width += insets.right;
                preferredSize.height += insets.bottom;
                contentPane.setMinimumSize(preferredSize);
                contentPane.setPreferredSize(preferredSize);
        }
        setSize(495, 380);
        setLocationRelativeTo(getOwner());
    }
     private JPanel panel1;
    private JLabel label2;
    private JLabel label1;
    public  JTextField textField1;
    public  JTextField textField2;
    private JLabel label3;
    public  JTextField textField3;
    public  JTextField textField4;
    private JLabel label4;
    private JLabel label5;
    public  JTextField textField5;
    private JButton button1;
    private JButton button2;

}
```

Node Move:

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import javax.swing.*;
public class NodeMove extends JDialog {
   MSDP objMSDP;
        public NodeMove(Frame owner) {
                super(owner);
                initComponents();
        }
    public NodeMove(Frame owner,MSDP objMSDP) {
        super(owner);
```

```
        this.objMSDP=objMSDP;
        initComponents();
}       public NodeMove(Dialog owner) {
                super(owner);
                initComponents();
        }
        private void rd1buttonClick(ActionEvent e) {

          if(radioButton2.isSelected() ) {
            radioButton2.setSelected(false);
          }
           comboBox2.setEnabled(true);
        }
      public void  LoadCombo() {
        try  {
          Set key=objMSDP.listHashMap.entrySet();
                Iterator i = key.iterator();
            comboBox1.removeAllItems();
          comboBox1.removeAllItems();
            comboBox1.addItem("--Select--");
          comboBox2.addItem("--Select--");
                // Display elements
                while(i.hasNext()) {
                  Map.Entry me = (Map.Entry)i.next();
                  comboBox2.addItem(me.getKey());
                  ArrayList<Node>
currClusterlist=objMSDP.listHashMap.get(me.getKey());
                  for(int j=0;j<currClusterlist.size();j++) {
                    Node n=  currClusterlist.get(j);
                      comboBox1.addItem(n.getNodeName());
                  }
                }
        } catch (Exception ex)  {
          ex.printStackTrace();
        }
      }
          private void rd2buttonClick(ActionEvent e) {
            if(radioButton1.isSelected() ) {
              radioButton1.setSelected(false);
            }
            comboBox2.setEnabled(false);
          }
          private void button1ActionPerformed(ActionEvent e) {
          try  {
              boolean find=false;
              Set key=objMSDP.listHashMap.entrySet();
                    Iterator i = key.iterator();
                    // Display elements
                    while(i.hasNext()) {
                      Map.Entry me = (Map.Entry)i.next();
```

```
                comboBox2.addItem(me.getKey());
                    ArrayList<Node>
              currClusterlist=objMSDP.listHashMap.get(me.getKey());
                for(int j=0;j<currClusterlist.size();j++) {
                 Node n=  currClusterlist.get(j);
if(n.getNodeName().equalsIgnoreCase(comboBox1.getSelectedItem().toString() ))
                  {
                    if(radioButton1.isSelected() ) {
                      objMSDP.MoveNode(n,comboBox2.getSelectedItem().toString());
                      JOptionPane.showMessageDialog(null,"Node Move to
"+comboBox2.getSelectedItem().toString());
                       find=true;
                       break;
                     }
                    else {
objMSDP.DeleteNode(n,comboBox2.getSelectedItem().toString());
                        JOptionPane.showMessageDialog(null,"Node Deleted!");
                        find=true;
                       break;
                     }
                   }
                  }
                  if(find) {
                    break;
                  }
                 }
               } catch (Exception ex)  {
                 ex.printStackTrace();
               }
               }
               private void initComponents() {
                       // Component initialization
                       label1 = new JLabel();
                       comboBox1 = new JComboBox();
                       label2 = new JLabel();
                       radioButton1 = new JRadioButton();
                       radioButton2 = new JRadioButton();
                       label3 = new JLabel();
                       comboBox2 = new JComboBox();
                       button1 = new JButton();
                       //======== this ========
                       setTitle("Node Movement");
                       setResizable(false);
                       Container contentPane = getContentPane();
                       contentPane.setLayout(null);
                       //---- label1 ----
                       label1.setText("Nodes");
                       contentPane.add(label1);
                       label1.setBounds(40, 40, 95, label1.getPreferredSize().height);
                       contentPane.add(comboBox1);
```

```
comboBox1.setBounds(110, 35, 120, comboBox1.getPreferredSize().height);
//---- label2 ----
label2.setText("Type");
contentPane.add(label2);
label2.setBounds(40, 80, 37, label2.getPreferredSize().height);
//---- radioButton1 ----
radioButton1.setText("Move");
radioButton1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                rd1buttonClick(e);
        }
});
contentPane.add(radioButton1);
radioButton1.setBounds(new Rectangle(new Point(110, 80),
radioButton1.getPreferredSize()));
//---- radioButton2 ----
radioButton2.setText("Delete");
radioButton2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                rd2buttonClick(e);
        }
});
contentPane.add(radioButton2);
radioButton2.setBounds(175, 80, 65, radioButton2.getPreferredSize().height);
//---- label3 ----
label3.setText("Cluster");
contentPane.add(label3);
label3.setBounds(30, 125, 46, label3.getPreferredSize().height);
contentPane.add(comboBox2);
comboBox2.setBounds(105, 125, 120, 25);
//---- button1 ----
button1.setText("Ok");
button1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                button1ActionPerformed(e);
        }
});
contentPane.add(button1);
button1.setBounds(270, 35, 105, 115);
{ // compute preferred size
        Dimension preferredSize = new Dimension();
        for(int i = 0; i < contentPane.getComponentCount(); i++) {
        Rectangle bounds = contentPane.getComponent(i).getBounds();
         preferredSize.width = Math.max(bounds.x + bounds.width,
         preferredSize.width);
          preferredSize.height = Math.max(bounds.y + bounds.height,
          preferredSize.height);
        }
        Insets insets = contentPane.getInsets();
        preferredSize.width += insets.right;
```

```
                preferredSize.height += insets.bottom;
                contentPane.setMinimumSize(preferredSize);
                contentPane.setPreferredSize(preferredSize);
            }
            setSize(420, 240);
          LoadCombo() ;
        radioButton1.setSelected(true);
            setLocationRelativeTo(null);
        }
        private JLabel label1;
        private JComboBox comboBox1;
        private JLabel label2;
        private JRadioButton radioButton1;
        private JRadioButton radioButton2;
        private JLabel label3;
        private JComboBox comboBox2;
        private JButton button1;
}
```

Common Function:

```
public class CommonFunction {
   public CommonFunction() {
   }
  public static boolean isNumber(String num) {
     try  {
         Double.parseDouble(num) ;
        return true;
     } catch (Exception ex)  {
        System.out.println(ex);
     }
     return false;
   }
 }
```

Node:

```
import java.awt.Point;
import java.awt.Rectangle;
import java.util.ArrayList;
public class Node {
   public String NodeName;
   public String  ServiceName;
   public String  NodeUID;
   public String  NodeType;
   int  TimeInterVal;
   public  Rectangle rect;
   public  Point  startPoint;
   public  Point  endPoint;
   public ArrayList ServiceList;
   public int Cluster;
    public String ClusterName;
```

```
public Node() {
}
public void setNodeName(String nodeName) {
   this.NodeName = nodeName;
}
public String getNodeName() {
   return NodeName;
}
public void setServiceName(String serviceName) {
   this.ServiceName = serviceName;
}
public String getServiceName() {
   return ServiceName;
}
public void setNodeUID(String nodeUID) {
   this.NodeUID = nodeUID;
}
public String getNodeUID() {
   return NodeUID;
}

public void setNodeType(String nodeType) {
   this.NodeType = nodeType;
}
public String getNodeType() {
   return NodeType;
}
public void setTimeInterVal(int timeInterVal) {
   this.TimeInterVal = timeInterVal;
}
public int getTimeInterVal() {
   return TimeInterVal;
}
public void setRect(Rectangle rect) {
   this.rect = rect;
}
public Rectangle getRect() {
   return rect;
}
public void setStartPoint(Point startPoint) {
   this.startPoint = startPoint;
}
public Point getStartPoint() {
   return startPoint;
}

public void setEndPoint(Point endPoint) {
   this.endPoint = endPoint;
}
public Point getEndPoint() {
```

```
    return endPoint;
  }
  public void setServiceList(ArrayList serviceList) {
    this.ServiceList = serviceList;
  }
  public ArrayList getServiceList() {
    return ServiceList;
  }
  public void setCluster(int cluster) {
    this.Cluster = cluster;
  }
  public int getCluster() {
    return Cluster;
  }
  public void setClusterName(String clusterName) {
    this.ClusterName = clusterName;
  }
  public String getClusterName() {
    return ClusterName;
  }
}
```

Node Panel:

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Rectangle;
import java.util.ArrayList;
import javax.swing.JPanel;
public class NodePanel extends JPanel {
   MSDP objMSDP;
   public NodePanel() {
   }
   public NodePanel(MSDP objMSDP) {
   this.objMSDP=objMSDP;
   }
   public void paintComponent(Graphics g)
      {
        super.paintComponent(g);
     Graphics2D     g2d = (Graphics2D) g;
        if(objMSDP.isPaint)
        {
          drawRect(g2d);
        }
      }
      public void drawRect(Graphics2D     g2d ) {
        try {
```

```
g2d.setColor(Color.yellow);
Point p=null;
ArrayList <Node>list=objMSDP.listHashMap.get(objMSDP.currentCluster);
if(list!=null)
{
int len=list.size();
int j=0;
for(int i=0;i< len;i++ ) {
   Node n=list.get(i);
  // System.out.println(len+" I value"+i);
    if((i+1)==list.size()) {
     Node n1  =list.get(0);
     p=n1.getStartPoint();
     n.setEndPoint(p);
       objMSDP.NodeList.remove(i);
       objMSDP.NodeList.add(i,n);
    }
    else {
       Node n1  =list.get(i+1);
       p=n1.getStartPoint();
       n.setEndPoint(p);
       objMSDP.NodeList.remove(i);
       objMSDP.NodeList.add(i,n);
    }
        g2d.setColor(Color.yellow);
       g2d.fill3DRect(n.getRect().x,n.getRect().y,n.getRect().width,n.getRect().heigh
       t,true );
     g2d.setColor(Color.BLUE);
     g2d.setFont(new Font("Arial", Font.BOLD, 11));

     g2d.drawString(n.getNodeName()+"["+n.getNodeType()+"]",n.getRect().x+50,n.
     getRect().y+25);
      g2d.setColor(Color.BLACK);
      g2d.setFont(new Font("Arial", Font.BOLD, 11));
      g2d.drawString((i+1)+"",n.getRect().x+25,n.getRect().y+25);
  }
  g2d.setColor(Color.GREEN);
  for(Node n:list) {

  g2d.drawLine(n.getStartPoint().x,n.getStartPoint().y,n.getEndPoint().x,n.getEndPoi
  nt().y);
  }
/*   g2d.setColor(Color.yellow);
  g2d.drawRect(x,y,w,h);
  rectObj.put(i,rc);
  // g2d.setColor(Color.BLUE );
  g2d.setFont(new Font("Tahoma", Font.PLAIN, 18));
  g2d.drawString(String.valueOf(i),x+(w/2),y+(h/2));
 */
  }
```

```
      } catch (Exception ex) {
        ex.printStackTrace();
      } finally {
      }

    }
  public int findOutDirection(Point mousePoint) {
    Rectangle bounds=null;
    try {
         int len=objMSDP.NodeList.size();
         for(int i=0;i< len;i++ ) {
         Node n=objMSDP.NodeList.get(i);
         bounds= n.getRect();
         if (mousePoint.x >= bounds.x && mousePoint.y >= bounds.y
                && mousePoint.x <= (bounds.x + bounds.width)
                && mousePoint.y <= (bounds.y + bounds.height)) {
                 NodeCreation nc=  new NodeCreation(new Frame());
                 nc.textField1.setText(n.getNodeName());
                 nc.textField2.setText(n.getServiceName());
                   nc.textField3.setText(""+n.getNodeUID());
                   nc.textField4.setText(n.getNodeType());
                   nc.textField5.setText(""+n.getTimeInterVal());
                   nc.setVisible(true);
                 }
        }
      } catch (Exception ex) {
        ex.printStackTrace();
      }
      return 9;
    }
  }
```