RESEARCH ARTICLE OPEN ACCESS

Remote Nodes Management Using RMI

Saurabh Malgaonkar*, Swarnalata Bollavarapu*, Tejas Hirave**
*(Computer Engineering Department, Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai, India)
** (Computer Engineering Department, Shah and Anchor Kutchhi Engineering College, Mumbai University, Mumbai, India)

ABSTRACT
We have focused upon the design and development of a remote desktop management system using Remote Method Invocation, since it has many advantages in terms of remote computation, software optimization, data access and storage services that do not require end-user knowledge of the physical location and configuration of the system. In this type of a system we have two types of users - clients and the server. The server can control all the activities of the clients and also impose certain restrictions on the clients. The clients can use the data or shared resources, software and information that are provided by the server to computers and other devices as a utility over a network. The major elements of desktop management are assets management, software deployment, patch management, remote desktop sharing, active directory reports and user logon reports. Remote desktop sharing enables users to access and use their own desktop from other machines or locations. This system is focused upon the real time events.
Keywords- cloud hosting services, distributed computing, remote control, remote monitoring, networking, real-time applications

I. INTRODUCTION

Computer networks is a collection of computers and other devices that can send and receive data from one another communication system, since networking deals with data, hence it becomes necessary to focus on the security. Managing the other remote computers is equally important in the congested network environment [1]. Management consists of monitoring, by monitoring the other computers on the network we know what others are doing on their system.

This real time application does not involve any end user (client) activity. Only user of the application is administrator sitting at server. This application can be applied to monitor activities of employee in a particular department or the work running on client's terminal specific lab work session.

In today's fast and growing world, every system is connected to network via internet or by a combinations of LAN (local area network) or by a WAN (wide area network) or by a MAN (metropolitan area network). In an environment where there is an administrator, a user present, management and monitoring becomes the important aspect of the network system. Administrator should have control over the remote computers.

This system provides many features and advantages and may have its applications in corporate firms and many other organizations. The need for time-effective and cost-effective solutions has been the basic motive behind the development of this project.

We have studied about the cloud services and have analyzed its advantages over the normal networks and hosting services. For this system we have preferred cloud based hosting because of the following advantages: flexibility, availability, fault tolerance, massive processing power and economic feasibility.

II. DESIGN

A. Product Function Overview

This real time application mainly deals with the administrator access from the server. It includes security and monitoring aspect of the network. It involves grabbing the screen of multiple client terminals and showing multiple client screens at server side. The main search and retrieve operation involves the discovery [2] [3] of the nodes of a system along with the grabbing of their information for their inclusion in the server list.

B. User Characteristics

There will be only one type of user in application –administrator. He/She is the only person to keep watch of networked client terminals from the server side only. He can do function like:

1. Add client.
2. Remove client.
3. Administrator message to the client.
4. Watching screen of single client terminals.

5. Simultaneously keeping watch of multiple clients.
6. Time flexibility for screen updates.
7. Performing remote operations.

C. General Characteristics

The client terminal to be watched should have client program installed over there.

1. Inputs and Outputs:

Input: Administrator does not have to enter any data. Just by clicking over options he can perform desired operations.

Output: Single terminal screen or multiple clients' screens mapped.

2. External Interface Requirements:

A user-friendly GUI has been developed which provides effective screens that an administrator can easily use.

D. Performance Constraints

In this application, the client screen changes are updated at specified interval of time but if multiple clients are sending their screens simultaneously, then there can be chances of increased network traffic. To overcome this problem, message can be passed from server to client about sending its screen after specified interval. It may be possible that on a networked terminal, there is no work going on & still if clients send its screen at each interval, traffic will be increased but this is also properly addressed. The deployment of the server is on Google cloud hosting service [6] that enables to handle and perform many operations simultaneously without any delays perfectly on a real time basis.

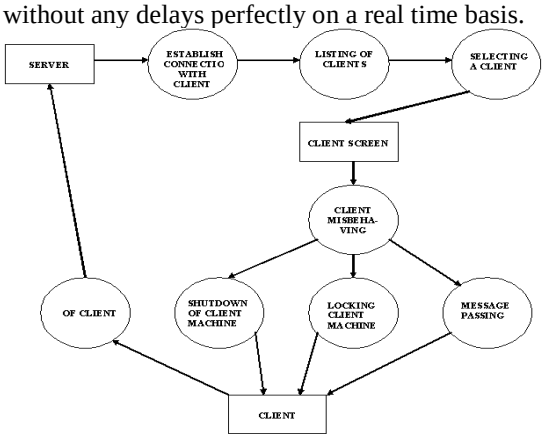

Fig 1: Functional Workflow of the system

III. IMPLEMENTATION

A. RMI

The Java Remote Method Invocation Application Programming Interface (API), or Java RMI [5], is a Java [4] API that performs the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java objects and distributed garbage collection.

1. The original implementation depends on Java Virtual Machine (JVM) class representation mechanisms and it thus only supports making calls from one JVM to another. The protocol underlying this Java-only implementation is known as Java Remote Method Protocol (JRMP).
2. In order to support code running in a non-JVM context, a CORBA version was later developed.

B. ROBOT CLASS

This class is used to generate native system input events for the purposes of test automation, self-running demos and other applications where control of the mouse and keyboard is needed. The primary purpose of Robot is to facilitate automated testing of Java platform implementation.

C. SAMPLE CODES

1. *Server Code for sending message to client*

```
Public void message method ()
{
    String s;
     s = "MESSAGE" + "\n";
System.out.println("hi");
int selected[] = list.getSelectedIndices();
System.out.println("hi " + selected.length);
     String hostnames[] = new
String[selected.length];
System.out.println("hi  " + hostnames.length);
for (int i = 0; i <selected.length; i++)
 {
hostnames[i] = (String)
model.getElementAt(selected[i]);
       String ipaddress = hostnames[i];
System.out.println("hi  " + ipaddress);
try
{
          Socket s1 = new Socket(ipaddress, 2030);
System.out.println("hi");
DataOutputStream dos = new
DataOutputStream(s1.getOutputStream());
System.out.println("hi");
dos.writeBytes(s);
dos.flush();
System.out.println("hi");
       }
catch (Exception e)
{
System.out.println(e);
       }
     }
System.out.println("hi");
msgsendms=  new msgsend(this);
System.out.println("hi");
}
```

2. *Server code tologoff client*

```
public void logoffmethod(){
 String s;
      s = "LOGOFF"+"\n";
           System.out.println("hi");
           int selected[] = list.getSelectedIndices();
           System.out.println("hi "+selected.length);
           String hostnames[]= new
String[selected.length];
           System.out.println("hi  "+hostnames.length);
           for(int i = 0 ; i<selected.length ; i++)
                      {
           hostnames[i]=(String)model.getElementAt(s
elected[i]);
String ipaddress = hostnames[i];
                      System.out.println("hi
"+ipaddress);
           model.removeElement(hostnames[i]);
try        {
                      Socket       s3       =       new
Socket(ipaddress,2030);
System.out.println("hi");
                      DataOutputStream dos = new

           DataOutputStream(s3.getOutputStream());
                                 System.out.println("hi");
                                 dos.writeBytes(s);
                                 dos.flush();
                                 System.out.println("hi");
                      rmodel.addElement(hostnames[i]);
                      }
                      catch(Exception
e){System.out.println(e);}
                      }
}
```

3. *Server code to restart client*

```
public void restartmethod(){
 String s;
      s = "RESTART"+"\n";
           System.out.println("hi");
           int selected[] = list.getSelectedIndices();
System.out.println("hi "+selected.length);
String hostnames[]= new String[selected.length];
           System.out.println("hi  "+hostnames.length);
for(int i = 0 ; i<selected.length ; i++)
{
           hostnames[i]=(String)model.getElementAt(s
elected[i]);
           String ipaddress = hostnames[i];
                      System.out.println("hi
"+ipaddress);

           model.removeElement(hostnames[i]);
           try                {
                      Socket       s2       =       new
Socket(ipaddress,2030);
                      System.out.println("hi");
                      DataOutputStream dos = new
DataOutputStream(s2.getOutputStream());
                      System.out.println("hi");
dos.writeBytes(s);
                      dos.flush();
                      System.out.println("hi");
                      rmodel.addElement(hostnames[i]);
                      }
catch(Exception e){System.out.println(e);}
                      }
}
```

4. *Client code for sending its live screen to server*

```
public class ClientFirst
 {
   Socket socket = null;
public static void main(String[] args)
{
newClientFirst().initialize("192.168.1.4", 1122);
}
public void initialize(String ip, int port )
{
      Robot robot = null; //Used to capture the screen
      Rectangle rectangle = null; //Used to represent
screen dimensions
try
{
System.out.println("Connecting to server .....");
socket = new Socket(ip, port);
System.out.println("Connection Established.");
//Get default screen device
GraphicsEnvironmentgEnv=GraphicsEnvironment.ge
tLocalGraphicsEnvironment();
GraphicsDevicegDev=gEnv.getDefaultScreenDevice
();
//Get screen dimensions
          Dimension       dim       =
Toolkit.getDefaultToolkit().getScreenSize();
rectangle = new Rectangle(dim);
          //Prepare Robot object
robot = new Robot(gDev);
//ScreenSpyer sends screenshots of the client screen
newScreenSpyer(socket,robot,rectangle);
 //ServerDelegaterecievesserver commands and
execute them
newServerDelegate(socket,robot);
      }
catch (UnknownHostException ex)
 {
ex.printStackTrace();
}
catch (IOException ex)
{
ex.printStackTrace();
```

```
}
catch (AWTException ex)
        {
ex.printStackTrace();
        }
   }
}
```

IV. SCREENS

The server side was successfully deployed on the cloud from where it was used to monitor and manage clients of a particular network.

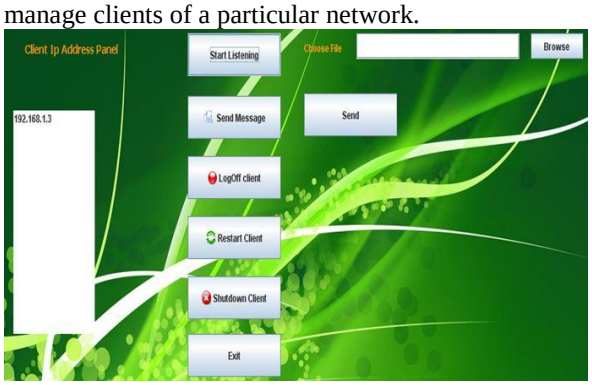

Fig 2: Server side interface

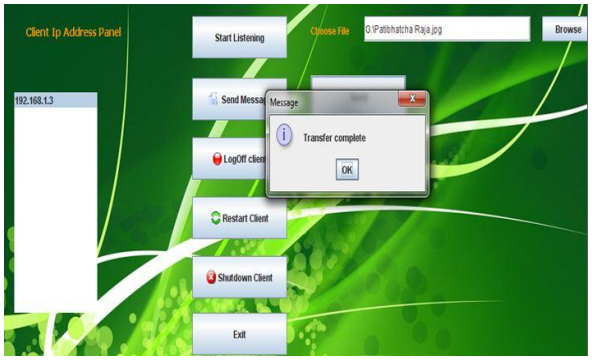

Fig 3: Successful file transfer operation

Successful operations were carried out using the server interface with very minute or almost no delays and the remote clients functioned properly.

V. CONCLUSION

This application mainly deals with the administrator access from the server. It includes security and monitoring aspect of the network remotely. It involves grabbing the screen of multiple client terminals and showing multiple client screens at server side. This system also provides facility of file transfer from server computer to client computer. These files are stored at the address specified by the client in its corresponding code. The server also has the ability to remotely control the entire available clients. The server can perform actions such as client restart, client logoff & client shutdown.

The both client and server applications are very lightweight (size <1MB combined), hence very flexible and easy to use.

VI. Future Work

"Remote cloud based monitoring" has many features but there may be many facilities that can be added to enhance the working of our system. System Design is creative, it is almost impossible to create a finished system with the first plan. Hence there is always a scope for improvements.

1) Creation of log file: To retrieve the information store at client side and have information regarding all the operations done in the past.
2) Lock: To disable the inputs from mouse and keyboard of remote client.
3) Benchmarking: We are currently hosting this at cloud, but we need to work upon the time based each operation result for the various type of hosting platforms (cloud, dedicated, Linux-based…) available.

ACKNOWLEDGEMENTS

Our sincere thanks to Sakshi "Geeta" Surve, who is an assistant professor at the Thadomal Shahani Engineering College, Computer Engineering department, Mumbai University, Mumbai, India; for her constant support, guidance and feedback for implementing this system. Her expert knowledge of distributed computing and cloud computing has always been a source of motivation and inspiration to us.

REFERENCES

[1] Kenneth P. Birman (2005), "Reliable distributed systems".
[2] Zupeng Li, Daoyin Huang, Jianhua Huang, "Research of Peer Discovery Method in Peer-to-Peer Network" ,12th IEEE Conference on distributed computing and networking, pp. 37-42, 2008.
[3] L. Alima, A. Ghodsi, and S. Haridi. A framework for structured peer- to-peer overlay networks in Global Computing, 223–249, 2011.
[4] "Java ", http://www.java.com/en/, December 18, 2013.
[5] "Remote Method Invocation", http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html, January 19, 2013.
[6] "Google Cloud Platform", https://cloud.google.com/, March 23, 2013.

 **Saurabh Malgaonkar** is a post graduate (Masters in Computer Engineering) pass out student of the Thadomal Shahani Engineering College, Computer Engineering Department.
His research interests are networking, distributed systems and computing, artificial intelligence, data mining and in particular the intersection of these fields.

 **Tejas Hirave** is a post graduate (Masters in Computer Engineering) pass out student of the Thadomal Shahani Engineering College, Computer Engineering Department.
His research interests are P2P systems, database systems, data mining, data structures and networking.

 **Ms.Swarnalata Bollavarapu** has received M.E. (Computer Engg.) from Thadomal Shahani Engineering College in 2011. She has more than 8 years of teaching experience. She is currently working as Assistant professor in Department of Computer Engineering at MPSTME, NMIMS University, Mumbai. Her areas of interest includes Information Security, Distributed Computing and Cloud Computing.
.