

# Design And Implementation of LAN Commander Applicable For Efficient Network Administration Solution Using JAVA Programming

<sup>1</sup>Dr Rajamohan Parthasarathy, <sup>2</sup>Dr Leelavathy Rajamanickam, <sup>3</sup>Mr Seow Soon Loong,  
<sup>4</sup>Ms Preethy Ayyappan, <sup>5</sup>Hazim Al Saidawi

**Abstract** - The defining factor of our today world is communication technology and particularly, the Internet; that vast multitude of computer networks that came together to become a meaningful whole. Our world of now relies upon those networks, and the expansion and scaling of those networks is always underway, trying to keep ahead of the ever-growing needs of large companies and organizations. As these networks grow, the number of technical challenges that maintaining them grows as well. Depending on the size, a single company may house dozens or hundreds or even more subnets, and all must be kept in check, to make sure that the organization's work goes ahead unhindered. The latter lines basically describe how the job of "Network Administrator" came to be overcome. They are the first and perhaps only line of defense against unforeseen circumstances that may hinder the operations of an organization and although it seems straightforward on paper, the sheer amount of network end devices makes their task seem impossible at a glance. This paper aims to facilitate the operation of Computer Networks, by allowing Network Administrators to remotely manage endpoints through means of a Java Software called "LAN Commander". This function will allow for various features, such as Remote Desktop control and Network Monitoring.

**Keywords** - Local Area Network (LAN) Commander, Remote Administration Tool (RAT), Network Monitoring Tool, LAN Chat Server, LAN Chat Client, LAN Scanner, Java Remote Method Invocation (RMI).

## I. INTRODUCTION

### I.I. Background

A variety of tools were developed to tackle the issues of Network Administrators, and most of these tools seem to provide useful functions, such as remote desktop, chat, file transfer and other functions, some tools were more focused on the network aspect of the Network Administrator's work, providing the means to monitor certain aspects of the network in real-time, and thus improving the problem detection and response time of the administrator.

### I.II. Problem Statement and Scopes

The tools that were used to fulfill the remote administration tasks were apparently unsafe, as reports surfaced of these tools being used as backdoors into unsuspecting users' computers, allowing the developer to abuse these tools to further their own agenda. From the perspective of a future Network Administrator, the security of the network they are entrusted to manage is a key imperative. This research works aims to create an application that enables network administrators to monitor and control certain aspects of their network and its end devices, by using java network

programming.

### **I.III. Proposed System**

The LAN Commander Java-Based Software solution seeks to provide a software capable of both remote control and network monitoring functions. The program shall provide both a server module and a client module, with the server module being on the administrator's PC, and the client module deployed on the client PC, one for each client endpoint. The administrator module shall have a security login feature to prevent unauthorized access, and both modules shall be capable of communicating via at least chat messages, and for the administrator module to be able to send files to the client modules that enables remote control functions together with network monitoring functions, The remote desktop control allows as a method to diagnose a target PC on the network with accurate status of the current network and provide quick text communication with employees to resolve simple issues that do not require access to the desktop remotely. The work to develop a software solution called "LAN Commander" using Java Network Programming and its most notable features will include remote desktop control, File Transfer and network monitoring.

## **II. LITERATURE REVIEW**

This section of the report attempts to define the existing technologies that this project is based upon, and provides examples of these technologies in order to better explain the relevance of the proposed software (Wang Ping et al., 2007).

### **II.I. Existing Technologies**

#### **II.I.I. The Remote Administration Tool (RAT)**

RAT is a software tool that enables the user to control another computer remotely. A RAT is not a Trojan horse, since the purpose of RAT's is to only control another person's computer after approval. RATs can also be used for remotely control their own computers within a local area network or have full control over their home computer from the workplace (Rupal D. Bhatt et al., 2013). A RAT consists of two parts: A client and a server. The user (master) who wants to control another computer (slave) uses a client, which is usually an interface-based application, while "slave" runs a program called server. The server is a program that is usually run in a hidden mode, to make the whole process smoother.

#### **II.I.II. The Network Monitoring Tool (NMT)**

The field of network management has become a vital aspect in modern computer networks. Today's networks tend to be heterogeneous, comprising of a variety of computers, hubs, switches, bridges, routers and various other network devices from different manufacturers. The task of monitoring and managing network resources has therefore become more taxing and complex. Network administrators are consequently faced with the challenge of ensuring that customer satisfaction is guaranteed through constant monitoring and management of network resources. Network Monitoring Tools provide a variety of functionalities, such as status monitoring, information about connected hosts, Simple ICMP requests (Ping). These tools may also measure current connection bandwidth, intercept packets throughout the network.

### **II.II. Existing Solutions**

While the inspiration of LAN Commander is not entirely an original idea, as several existing programs have features similar to the proposed solution, none of these tools seem to provide the same level of convenience and features that except the network administrators (Debra L S et al., 2001)

### II.II.I. JrDesktop (Java Remote Desktop)

A cross-platform software for remote desktop control, remote assistance and desktop sharing. jrDesktop is used to remote control to a target pc on a local network. While jrDesktop does have a good implementation of remote-control using java, it lacks a lot of functions, such as the ability to communicate with the client, which could save a lot of time while attempting to identify the issue, its development was also shutdown many years ago (Abhishek K G et al.,2018).

### II.II.II. njRAT

njRAT is a Remote Access Trojan (RAT) application that may run in the background and silently collect information about the system, connected users, and network activity. njRAT may attempt to steal stored credentials, usernames and passwords and other personal and confidential information. njRAT may allow an attacker to install additional software to the infected machine. njRAT is one of many trojan variants of the Java-Based Remote Access tools (njRAT, 2013).

### II.II.III. Real Network Monitor (RNM)

RNM is a professional, freeware application designed to monitor internet traffic information and statistics. RNM allows for real-time monitoring of incoming and outgoing network traffic and logging of daily bandwidth usage in addition to upload and download speeds. The user can either use GUI for monitoring or use the desktop overlay in order to multitask while monitoring more efficiently. RNM has an internal website filter, designed to access the hosts file, which exists in both Linux and windows, in order to block access to certain websites (Abdullah A M et al., 2015) Other notable features include a Gateway Finder, Interface lookup and a built-in traffic database that stores monthly total for bandwidth usage-

## II.III. Alternative Solutions Compared to Proposed Solution

Table 1. Comparison between Existing Solutions and LAN Commander

<i>Key Issues / Solutions</i>	<i>LAN Commander</i>	<i>jrDesktop</i>	<i>njRAT</i>	<i>Real Network Monitor</i>
Chat Communication	Yes	No	Yes	N/A
Remote Desktop Control	Yes	Yes	Yes	N/A
LAN IP Discovery	Yes	N/A	No	No
Real-Time Network Speed	No	N/A	N/A	Yes
Windows Command Line	Yes	No	Yes	No
Send Files / File Manager	Yes	Yes	Yes	N/A
Security	Trusted	Old	Trojan	Trusted

After reviewing and comparing the previous solutions, it is clear that none of the existing solutions match the specifications of the solution. The work will also provide an extra layer of security for network administrators by providing the open-source code to allow further improvement to the security provided by the proposed solution..

### III. METHODOLOGY, ANALYSIS AND DESIGN

#### III.I. Functional Requirements

##### III.I.I. Administrator

Connect to a remote endpoint with the proper credentials, Perform Network monitoring functions, Send to and receive messages from the endpoint's operator, Send Files to the connected remote computer and search for other hosts on the network.

##### III.I.II. Employee

Send messages, error reports to connected administrator.

#### III.II. Software Requirements

**III.II.I. Operating System:** Windows XP or later with JRE (Java Runtime Environment).

**III.II.II. Environment: Java SE:** Java Platform, Standard Edition, is used to develop and test java applications in a windows environment. Provides the necessary runtime for java applications in windows (Gallardo Raymond, 2013).

##### III.II.III. NetBeans IDE

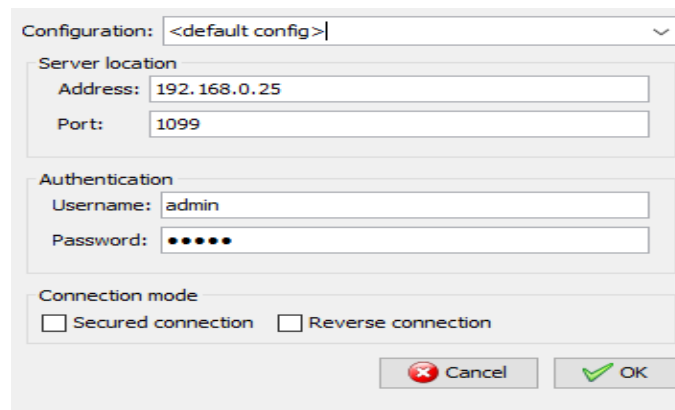
NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications).

#### III.III. Design

The design aspects of the program are still preliminary, and may change slightly or drastically in the future (Robert K Wysocki, 2009) The proposed software is made of several functions, for the sake of simplicity we will split the design preview based on the proposed functions:

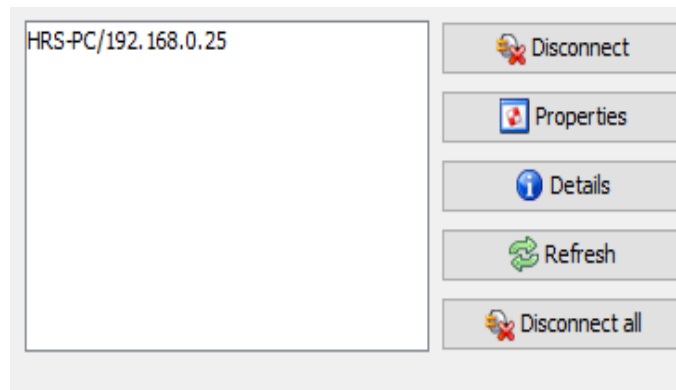
##### III.III.I. Remote Desktop

The proposed software is planned to contain the ability to connect to other devices based on their respective IP address. A port designated for that function can be specified manually and a username and/or a password requirement can be made to ensure the security of the system. Here is a preview of our concept of the connection setup screen.



The image shows a configuration dialog box for a remote desktop connection. It has a title bar with a dropdown menu for 'Configuration' set to '<default config>'. The dialog is divided into three sections: 'Server location' with fields for 'Address' (192.168.0.25) and 'Port' (1099); 'Authentication' with fields for 'Username' (admin) and 'Password' (masked with dots); and 'Connection mode' with two checkboxes: 'Secured connection' and 'Reverse connection', both of which are currently unchecked. At the bottom right, there are 'Cancel' and 'OK' buttons.

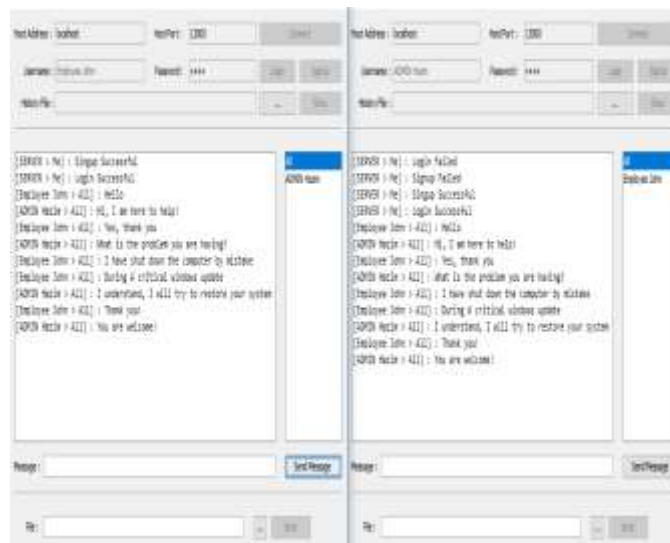
**Fig. 1. Outgoing Remote Desktop Connection Concept**



**Fig. 2. Connection Management Interface**

### III.III.II. Chat and File Transfer

It is possible to include 1 on 1 or multi-party chat and file transfer. Chat function was proposed in order to help the admin identify problems quickly by asking the employee at the endpoint to provide an insight towards the circumstances surrounding the malfunction. File transfer function also helps to send files directly to the endpoint. Those files may include certain diagnostic software or other necessary programs to restore functionality to the target PC, it would save time and bandwidth as opposed to downloading those files from the internet, since the LAN interface is usually much faster in matters of file transfer/download.



**Fig. 3. LAN Chat Interface Example**

### III.IV. Network Monitoring Functions

#### III.IV.I. Network Discovery and port scan

The ability to discover other hosts on the network, as well as a basic port scan to determine if there are any ports open on the target host. This information is sometimes important in helping to detect intrusions on the network, or possible security vulnerabilities.



Fig. 4. Network Discovery and Port Scanner

### III.IV.II. Ping other hosts

Sends an ICMP packet to the target host in order to measure latency, useful to detect and measure abnormal latency in the network.

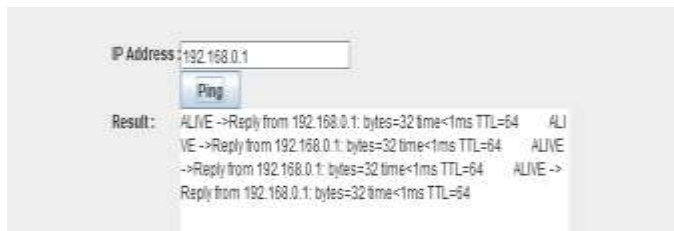


Fig. 5. Ping Module Concept

### III.IV.III. Traceroute

Traceroute command helps to identify the number of hosts the packet goes through before reaching its destination. It helps to keep track of performance issues such as increased latency due to an unnecessary number of hops, as well as unusual hop destinations which could mean that the data sent by the organization is being intercepted by an unwanted 3rd party.

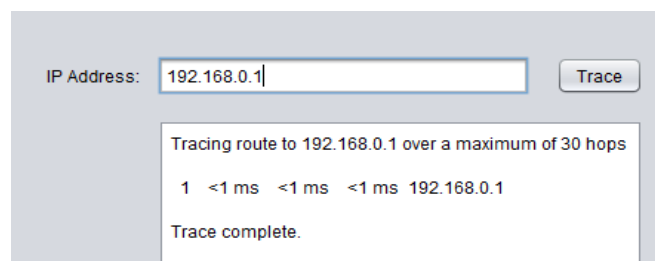


Fig. 6. Traceroute Module Concept

## IV. IMPLEMENTATION

**IV.I. Main Menu:** has clearly-labeled buttons that provide access to the features of the program.



**Fig. 7. Main Menu**

**IV.II. Host module:** Includes configuration options for the server and has the following features:

**IV.II.I. Connection Mode**

**IV.II.I.I. Manual:** Connect using an IP address

**IV.II.I.II. Auto Detect (Multihomed Server):** [Beta]: Uses multiple networks to achieve the best connection.

**IV.II.II. Connection Security**

**IV.II.II.I. SSL (Secure Socket Layers):** [Beta]: Provides an extra layer for security for a possible future Internet Connection Capable implementation of LAN Commander.

**IV.II.III. Host Address**

**IV.II.III.I. Address:** The IP address of the Client goes here. A feature was implemented to read the address automatically from the current active network adapter, thus reducing the complexity of the process.

**IV.II.III.II. Port:** The port to be used by the connection. A default value is assigned to this field to simplify the connection process.

**IV.II.IV. Authorization**

This feature allows for an extra layer of security by requiring a password to be able to use the Hosting feature.

**IV.II.IV.I. Username & Password:** to facilitate the operation of the software in its current testing phase, these fields are automatically filled.

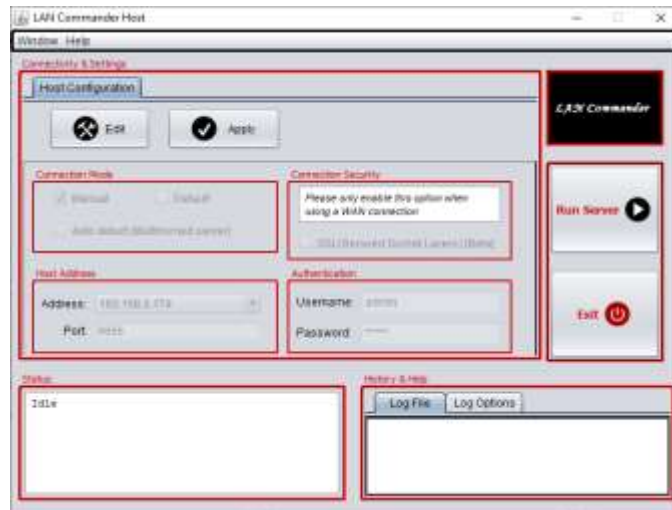
**IV.II.IV.II. Edit/Apply button:** Allows for any necessary adjustment of settings.

**IV.II.IV.III. Menu Bar:** Allows for extra control of the window, and to view information about the program.

**IV.II.IV.IV. Side Buttons:** Run Server: Run/Stop the server & Exit: Closes the module.

**IV.II.IV.V. Status Area:** Displays the current configuration and running status of the host server.

**IV.II.IV.VI. Log Area:** Log: Keeps track of important events. Useful for diagnostic purposes. Log Options:  
Allows for management of logs.



**Fig. 8. Host Module i**

### IV.III. Admin Module

**IV.III.I. Top Buttons:** These buttons provide control over the functions of the remote desktop module. Including the ability to Connect/Disconnect, Toggle control mode between View only/Full Control, file transfer\* [Beta], Start Host module and Exit.

**IV.III.II. Menu Bar:** Includes the same functions as in top buttons. Allows for extra control of the window, and to view information about the program.

**IV.III.III. User Account Information:** Allows the admin to enter his password to access a list of previously added hosts. The password can be changed here, and the table can be hidden while the admin is away from his station.

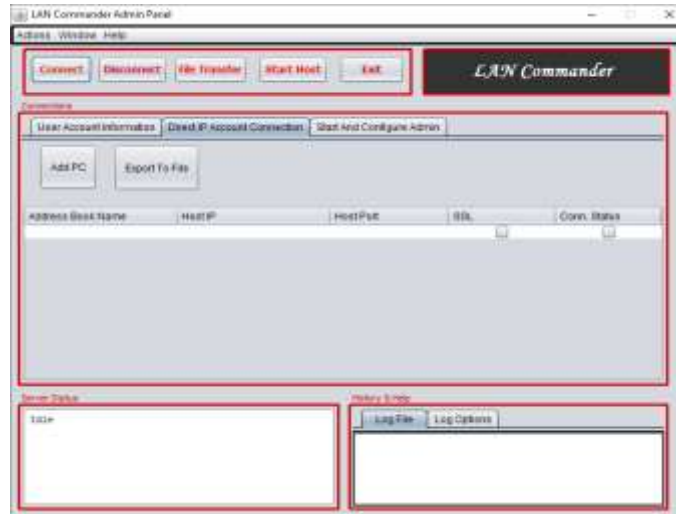


**Fig. 9. Admin Module 1 of 3**

**IV.III.IV. Direct IP Connection:** This section allows the admin to add a new pc to his list. Once added the remote connection will start immediately.

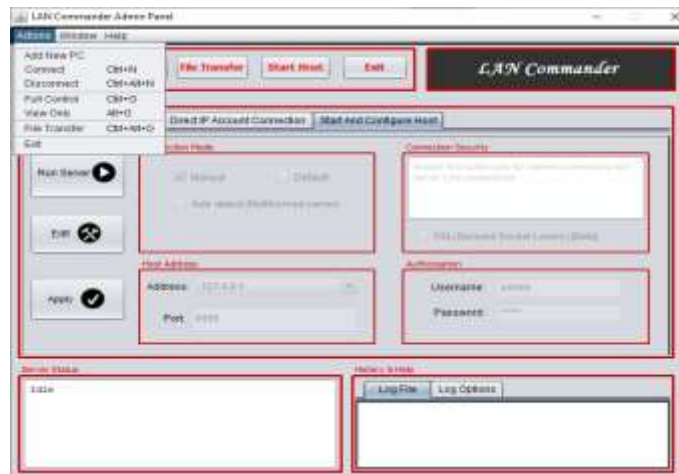


**IV.III.V. Export to file (beta):** Export the table contents to a text file (employee.txt) for future reference



**Fig. 10. Admin Module 2 of 3**

**IV.III.VI. Start and Configure Host:** Built-in Host module.



**Fig. 11. Admin Module 3 of 3**

**IV.III.VII. Status Area:** Displays the current configuration and running status of the host server.

**IV.III.VIII. Log Area:** Log: Keeps track of important events. Useful for diagnostic purposes and Log Options allows for management of logs.

**IV.III.IX. Remote Host Viewer:** The Interface that displays the screen of the remote computer. It has a number of buttons to facilitate the operation of the remote viewer, as well as a zoom feature.



Fig. 12. Remote Host Viewer 1

#### IV.III.X. LAN Scanner

**IV.III.X.I. IP scanner:** detects reachable IP address on a network within a certain IP range.

**IV.III.X.II. Port Scanner:** scans for open ports in a specific host address.

**IV.III.X.III. Ping:** Sends a ping to a designated IP address. Able to resolve hostnames to DNS.

**IV.III.X.IV. Traceroute:** Trace the route that a packet travels from the current network to its destination.

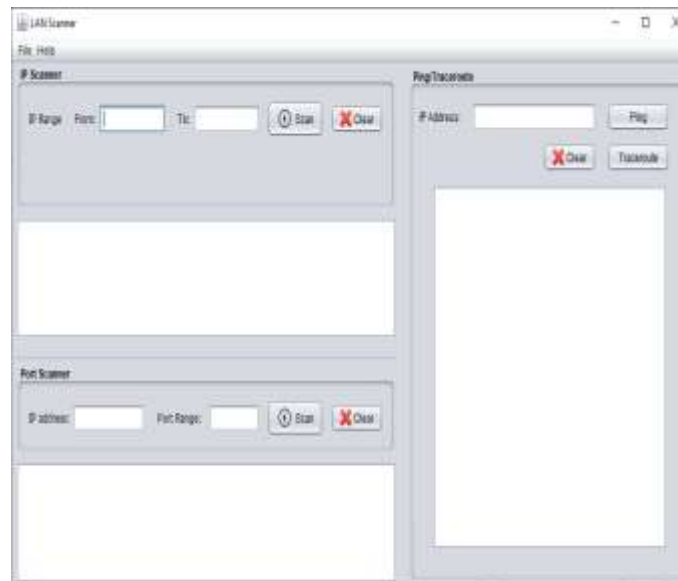
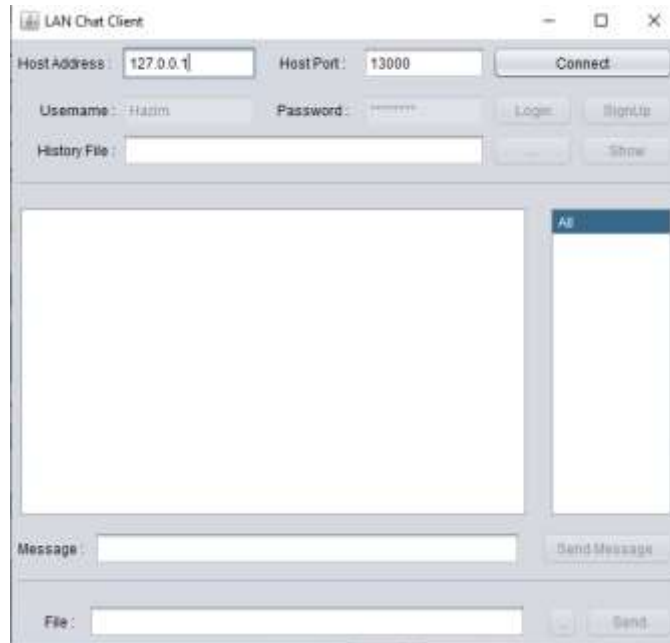


Fig. 13. LAN Scanner 1

#### IV.III.XI. LAN Chat Client

Allows the employee at a remote station to reach-out to the Network Admin when there is an issue. Facilitates remote communication between both parties in environments where other means of chat (i.e. Social Media) are restricted by corporate policy, or in other cases where the Network itself is isolated from the Internet. The process of using this module for the task such as the employee attempts to connect to the server, the employee enters their name and a password, If the employee had already registered, they should click sign-in, otherwise they should click sign-

up, once the previous step is complete, the employee will be able to chat with the admin and the file sending feature allows both parties to send and receive files.



**Fig. 14. LAN Chat Client**

#### **IV.III.XII. LAN Chat Server**

This separate module allows the admin to host a chat server, where other clients (Employees) can connect to report problems or request assistance. This server should remain running at all times.



**Fig. 15. LAN Chat Server**

#### **IV.IV. Coding**

Once the program is launched, the Main Menu Appears, this is done by the way of setting the main.java class inside the package hosting the menu frame as the main class for the program. Creating a new JFrame (MainMenu.java). The main menu theme is set as the nimbus theme by adding the following code to the main args method of Main.java:

```
try {  
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  
        if ("Nimbus".equals(info.getName())) {  
            javax.swing.UIManager.setLookAndFeel(info.getClassName());  
            break;  
        }  
    }  
}
```

- Create a new JFrame Opwi\_Fr.

```
MainMenu OpWi_Fr = new MainMenu();
```

- Set the default bounds for the main menu window

```
Toolkit tk = Toolkit.getDefaultToolkit();  
ySize = (int) tk.getScreenSize().getHeight();  
xSize = (int) tk.getScreenSize().getWidth();  
OpWi_Fr.setBounds((xSize/2) - 250, (ySize/2) - 200, 465, 340);  
OpWi_Fr.setResizable(false); //prevent resizing
```

- Set the menu (JFrame) as visible:

```
OpWi_Fr.setVisible(true);
```

Using JButtons to communicate with other JFrames:

```
// import packageName.JFrameClassName  
import LAN.RemoteHost.HostMainWindow;  
import LAN.Administrator.AdminMainWindow;  
import ui.ChatFrame;  
import LAN.Scanner.LanScanFrame;
```

Open an instance of the required JFrame code as below.

```
// JFrameClassName.MainMethod(null);  
ChatFrame.main(null);
```

Once the window appears, several options appear to the user as buttons. These JButtons contain arguments that launch the desired module. Their method of operation is as follows. Import the class that belongs to the JPanel that contains the desired functionality. Enter the following arguments into the JButtons' actionPerformed method:

```
NewWindow.main(null); //creates a JFrame from the NewWindow JFrame
```

LAN Commander follows a flexible modular approach. Each module was developed separately, then all modules were combined in a single package. The following modules are included: Modules that operate the Remote Viewer (LAN.Viewer Prefix), Modules that operate the chat function (socket – ui), Modules that operate the LAN Scanner function (LAN.Scanner). The following sub-functions are included that Logging & Keyboard workaround.

#### **IV.IV.I. Java RMI (Harsh M et al., 2013) & (William, 2001)**

The Java RMI API was originally designed by Ann Wollrath, Roger Riggs, and Jim Waldo at Sun Microsystems Laboratories. Its distributed object model allows the programmer to build a complex distributed system in an efficient, maintainable, and object-oriented way. Due to certain limitations during the development, the Server for the chat function cannot reside in the same directory as the chat client, Thus the server was removed from the main program, and is now an independent program. The program uses RMI to operate the remote control and screen sharing function.

#### **Program Components & Features**

The remote viewer class **Viewer.java** in the package

##### **IV.IV.I.I. LAN.Viewer.RMI**

Extends Thread Class in java, allowing for multiple threads at once. public class Viewer extends Thread. Loads the configuration from the Host Module (username – pass - ...etc.), Contains Methods that control various states of

the remote connection (Connect – Disconnect – Stop), Contains code to modify the IP table data and Contains code to allow for the logging feature. The LAN.ViewerHelper package contains the following classes:

**IV.IV.I.II. Recorder.java**

Extends Thread Java class (Multi-Threading), Controls the remote screen output and Controls the Admin Input (View Only/Full Control).

**IV.IV.I.II. Screenplayer.java**

Handles the user's interactions with the screen (Mouse Clicks – Mouse scroll – Keyboard Key presses)

**IV.IV.I.II. ViewerData.java**

Sets Important values for the remote viewer.

**IV.IV.I.II. ViewerGUI.java**

The Jform that hosts the remote viewer. Contains the following elements:

**IV.IV.II. Control Buttons**

These buttons perform various functions that benefit the viewer's operator, includes the following:

**IV.IV.II.I. Play/Pause Button:** Starts or stops the remote screen.

**IV.IV.II.II. FullScreen Toggle button:** Toggles between full screen and window modes.

```
public void changeFullScreenMode() {
GraphicsDevice device = getGraphicsConfiguration().getDevice();
if (!device.isFullScreenSupported()) return;
fullScreenMode = !fullScreenMode;
if (fullScreenMode) { // Full screen mode
dispose();
setUndecorated(true);
device.setFullScreenWindow(this);
setVisible(true);
jBtnFullNormal.setIcon(new ImageIcon(main.NORMAL_SCREEN_ICON));
}
else { // Normal screen mode
dispose();
setUndecorated(false);
device.setFullScreenWindow(null);
setVisible(true);
jBtnFullNormal.setIcon(newImageIcon(main.FULL_SCREEN_ICON)); } }
```

**IV.IV.II.III. View Only/Full Control**

Toggles the admin ability to modify the remote screen.

**IV.IV.III. Remote host properties**

Displays useful information about the currently connected remote host.

#### **IV.IV.III.I. Connection Info**

Displays information about the remote connection, such as speed.

**IV.IV.III.II. Exit Button:** Closes the remote viewer.

**IV.IV.III.III. Zoom Feature:** Zooms the screen in or out. Available values range from 25% to 200%

#### **IV.IV.III.IV. Logging: LAN.Host.Logs & LAN.Admin.Logs**

The host module and the Admin module have a separate log package for each of them. These packages contain the necessary classes to generate an informative log file. These logs help keep track of the program's states and may provide useful insights into possible errors. They also contain an interface to view the logs. This interface is implemented in a manner similar to the way the main menu functions.

#### **IV.IV.IV. LAN Scanner:**

The package LAN.Scanner contains the necessary classes to implement the most important feature of this program. The LAN monitor. The monitor provides various functions, each class contains a part of the functionality:

##### **IV.IV.IV.I. IPrange.java**

This java class contains the necessary code to recognize the input IP, determine if its valid or not, and determine which network class it belongs to (A,B,C).

##### **IV.IV.IV.II. IPscanner.java**

This class determines whether a host on a network is reachable or not, by means of a simple ping test. If the test returned a successful ping, there is a reachable host. The IP range previously processed in IPrange.java, is used here to determine the start and end IP addresses for the test. After the test is finished, the result is displayed in the text area specific to the IPscanner.

```
for(int i = startIp; i < stopIp; i++)
{
    try
    {
        InetAddress address = InetAddress.getByName(subnet + i); System.out.println("Pinging: " + subnet + i);
        //scanProgressBar.setValue(i);//updating progress bar
        ping = address.isReachable(TIMEOUT);
        if(ping)
        {
            System.out.println(subnet + i + " ----> is alive!"+"\n");
            ipadd += subnet + i + " ----> is alive!\n";
            numOfHost++;
        }
        else
        {
            ipadd += subnet + i + "\n"; }
        //updating text area
```

```
        textArea.append(ipadd);  
        ipadd = "";  
    }
```

#### **IV.IV.IV.III. Portscanner.java**

Scans for open ports by attempting to open a socket for each port in the scan range. The below code details how it works:

```
try  
{  
    InetAddress addr = InetAddress.getByName(ipAddr);  
    ipPorts = "";  
    for(int i = startPort; i < endPort; i++)  
    {  
        try  
        {  
            Socket s = new Socket();//trying to open connection  
            SocketAddress address = new InetSocketAddress(ipAddr, i);  
            s.connect(address, timeOut);  
            if(s.isConnected())  
            {  
                portCounter++;  
                ipPorts += "port " + i + " -----> open!!!" + "\n";  
                //we found open port  
                s.close();//closing port  
            } }  
        }  
    }  
}
```

#### **IV.IV.IV.IV. LANScanFrame.java**

A JFrame that provides interface to the classes mentioned above, as well as the following features:

Ping & Traceroute: Using the following code, a ping request can be sent to the CMD. The response is recorded directly in the specific JTextArea:

```
BufferedReader in;  
try{  
    Runtime r = Runtime.getRuntime();  
    Process p = r.exec("Ping" + " " + jTextField1.getText());  
    in = new BufferedReader(new InputStreamReader(p.getInputStream()));  
    String line;  
    if(p==null)  
        System.out.println("could not connect");  
    while((line=in.readLine())!=null){  
        System.out.println(line);  
        redirectSystemStreams(); }  
}
```

Similarly, traceroute (tracert) and any other CMD command may be substituted for ping in the code above.

**IV.IV.V. LAN Chat: Server:** The package *socket* contains the following classes:

**IV.IV.V.I. Database.java**

Contains the code necessary to implement the database feature. The database is stored in an xml file, and this class contains the necessary arguments to handle this type of file. The database stores login information and passwords.

Example: method to match login information with xml file:

```
public boolean checkLogin(String username, String password)
{
    if (!userExists(username)) return false;
    try {
        File fXmlFile = new File(this.filePath);
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.parse(fXmlFile);
        doc.getDocumentElement().normalize();
        NodeList nList = doc.getElementsByTagName("user");
        for (int temp = 0; temp < nList.getLength(); temp++) {
            Node nNode = nList.item(temp);
            if (nNode.getNodeType() == 1) {
                Element eElement = (Element)nNode;
                if ((getTagValue("username", eElement).equals(username)) && (getTagValue("password",
                eElement).equals(password))) { return true;
                    } } }
                System.out.println("Hippie");
                return false;
            }
        }
```

**IV.IV.V.II. Message.java**

Stores variables related to the messaging functionality.

**IV.IV.V.III. ServerThread.java & SocketServer.java**

The functionalities of these two classes are interchangeable, therefore, both will be explained under this section.

Some of the functions include the following such as setting the default IP and port for the connection, handling the socket connection, sends the response when a user tries to login, and announce the current user list, Implement java threading to allow for more than one user to connect at once.

**IV.IV.V.IV. ServerFrame.java**



Provides a simple interface for the server. Once the user has selected the database file, the option to start the server becomes available. This function is implemented to remind the admin about the database file before starting the server. Here is the code for jButton1, which is the “browse” button that opens the jfilechooser. A Boolean condition is implemented here to ensure that the user has selected a database file:

```
private void jButton2ActionPerformed(ActionEvent evt) {  
    this.fileChooser.showDialog(this, "Select");  
    java.io.File file = this.fileChooser.getSelectedFile();  
    if (file != null) {  
        this.filePath = file.getPath();  
if (isWin32()) this.filePath = this.filePath.replace("\\", "/");  
        this.jTextField3.setText(this.filePath);  
        this.jButton1.setEnabled(true);  
    } }  
}
```

**IV.IV.VI. Chat Client:** The package socket contains the following classes:

**IV.IV.VII. Download.java**

Contains the Public class “Download” which implements java runnable threads. This class handles the file sharing feature of LAN commander, here is an example on how it works:

```
public void run()  
{  
try {  
    this.socket = this.server.accept();  
System.out.println("Download:"+this.socket.getRemoteSocketAddress());  
    this.In = this.socket.getInputStream();  
    this.Out = new FileOutputStream(this.saveTo);  
    byte[] buffer = new byte[1024];  
    int count;  
while ((count = this.In.read(buffer)) >= 0)  
{ this.Out.write(buffer, 0, count);  
}  
this.Out.flush();  
this.ui.jTextArea1.append("[Application > Me] : Download complete\n");  
    if (this.Out != null) this.Out.close();  
    if (this.In != null) this.In.close();  
    if (this.socket != null) this.socket.close();  
}  
}
```

**IV.IV.VIII. History.java**

Similar to database.java class in the chat server, this class allows the client to store the chat history locally. Unlike the database, however, it does not store passwords. Example: Method FillTable to display the chat history in the chat

client:

```
public void FillTable(HistoryFrame frame)
{
DefaultTableModel model = (DefaultTableModel)frame.jTable1.getModel();
Try {
File fXmlFile = new File(this.filePath);
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(fXmlFile);
doc.getDocumentElement().normalize();
NodeList nList = doc.getElementsByTagName("message");
for (int temp = 0; temp < nList.getLength(); temp++) {
Node nNode = nList.item(temp);
if (nNode.getNodeType() == 1) {
Element eElement = (Element)nNode;
model.addRow(new Object[] { getTagValue("sender", eElement), getTagValue("content", eElement),
getTagValue("recipient", eElement), getTagValue("time", eElement) });
} } }
}
```

#### **IV.IV.VI.III. Message.java**

Stores variables related to the messaging functionality.

#### **IV.IV.VI.IV. SocketClient.java**

This class handles communication with the socket server. Here is an example of the implementation of Messaging in the app:

```
try {
Message msg = (Message)this.In.readObject();
System.out.println("Incoming : " + msg.toString());
if (msg.type.equals("message")) {
if (msg.recipient.equals(this.ui.username)) {
this.ui.jTextArea1.append("[ " + msg.sender + " > Me] : " + msg.content + "\n"); }
else {
this.ui.jTextArea1.append("[ " + msg.sender + " > " + msg.recipient + " ] : " + msg.content + "\n"); }
if (!(!msg.content.equals("bye")) && (!msg.sender.equals(this.ui.username))) {
String msgTime = new java.util.Date().toString();
Try
{ this.hist.addMessage(msg, msgTime);
DefaultTableModel table = (DefaultTableModel)this.ui.historyFrame.jTable1.getModel();
table.addRow(new Object[] { msg.sender, msg.content, "Me", msgTime });
}
}
```

Messages are recognized as either a “message” which is the normal type of message, or as “login” type, which is the type of message that the chat client sends to the server upon login:

```
else if (msg.type.equals("login")) {
    if (msg.content.equals("TRUE")) {    this.ui.jButton2.setEnabled(false);this.ui.jButton3.setEnabled(false);
this.ui.jButton4.setEnabled(true);this.ui.jButton5.setEnabled(true);
this.ui.jTextArea1.append("[SERVER > Me] : Login Successful\n");
this.ui.jTextField3.setEnabled(false);this.ui.jPasswordField1.setEnabled(false);
    }
    else {
this.ui.jTextArea1.append("[SERVER > Me] : Login Failed\n"); }
Upload.java
```

This class handles the uploading of files. So while this class handles sending the file, the previously mentioned class Download.java handles the downloads.

```
try {
    byte[] buffer = new byte[1024];
    int count;
    while ((count = this.In.read(buffer)) >= 0) {
        this.Out.write(buffer, 0, count); }
    this.Out.flush();
this.ui.jTextArea1.append("[Appication > Me] : File upload complete\n");
this.ui.jButton5.setEnabled(true);this.ui.jButton6.setEnabled(true);
    this.ui.jTextField5.setVisible(true);
    if (this.In != null) this.In.close();
    if (this.Out != null) this.Out.close();
    if (this.socket != null) this.socket.close();
    }
}
```

**IV.IV.VII. Host Module:** Provides the interface necessary to start a remote-control server. Implements the following features:

#### **IV.IV.VII.I. IP address Recognition**

The program automatically loads the current IP address from the default network adapter in Windows, and thus eliminates the complexity of locating and entering the IP address.

```
jComboBoxLocalAdrs=new javax.swing.JComboBox (InetAdrUtility.getLocalIPAddresses());
```

#### **IV.IV.VII.II. Authentication**

User Name and password required to operate the server. For testing purposes, the user name and password are filled by default. Example: Method to load configuration from the config file

```
HostServerConfig.loadConfiguration();
```

```
jComboBoxLocalAdrs.setSelectedItem(HostServerConfig.server_address);
```

```
jTextFieldPort.setText(String.valueOf(HostServerConfig.server_port));
```

```
jTextFieldUsername.setText(HostServerConfig.username);    jPasswordField.setText(HostServerConfig.password);
```

```
jCheckBoxSSLEnabled.setSelected(HostServerConfig.ssl_enabled);
jCheckBoxMultihomedEnabled.setSelected(HostServerConfig.multihomed_enabled);
jCheckBoxDefaultAdr.setSelected(HostServerConfig.default_address);
    if (jCheckBoxMultihomedEnabled.isSelected())        jCheckBoxMultihomedEnabledActionPerformed(null);
    else
jCheckBoxManualAdrActionPerformed(null);}

```

#### **IV.IV.VII.III. Connection Mode**

Manual: By means of entering the IP address and port number (uses a single connection).  
Multihoming: [Beta]

Use multiple networks at once. (auto detect)

#### **IV.IV.VII.IV. Logging**

Display and record logs: Example: Logging events when the server is run.

```
private void runHostServerToggleButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (ServerAdmin.isRunning()) {
        ServerAdmin.Stop();
runHostServerToggleButton.setToolTipText("Server Stopped!"+"\nRun Again?");
// shows log info on console and adminLogInfoTextArea
        if(isLogEnableHost == Boolean.TRUE)
        {
AppLogInfoForHost.logger.info("Server Stopped");
        } }
    else {
        ServerAdmin.Start();
runHostServerToggleButton.setToolTipText("Server Running!\nWant to Stop ?");
// shows log info on console and adminLogInfoTextArea
if(isLogEnableHost == Boolean.TRUE){ AppLogInfoForHost.logger.info("Server Started"); }
}
}

```

**IV.IV.VIII. Admin Module:** Provides the control interface for all remote desktop related functions and has the following:

#### **IV.IV.VIII.I. Security Feature**

Password protected access to the list of client PCs

```
private void passOKButtonActionPerformed(java.awt.event.ActionEvent evt) {
    passHelper = UserPasswordField.getText();
    System.out.println(passHelper);
if (passHelper.equals(UserAccInfoPassword.passWord))
    {
        infoTablePanel.setVisible(true);
        infoTableSeeInstScrollPane.setVisible(false);
        passOKButton.setEnabled(false);
        passCancelButton.setEnabled(false);
        UserPasswordField.setText("");
    }
}

```

```
// shows log info on console and adminLogInfoTextArea
if (AdminMainWindow.isLogEnable == Boolean.TRUE) {
    AppLogInfoForAdministrator.logger.info(" User Account Information has been displayed . ");
    } }
else {
    JOptionPane.showMessageDialog (this, "Wrong Passward!" + "\n"+"Try Again", "ERROR",
OptionPane.ERROR_MESSAGE);
return; }
```

#### **IV.IV.VIII.II. Add New PCs to the list**

After entering the correct IP address, a remote machine will be added to the table and the Remote connection will start Immediately.

```
private void addPcOKButtonActionPerformed(java.awt.event.ActionEvent evt) {
for(int i=0;i<=AdminMainWindow.tableDispIndex;i++) {
if( AdminMainWindow.directAccTable1.getValueAt(i, 1).equals( jTextFieldIPAdr.getText() ) )
{
    JOptionPane.showMessageDialog(this, "This PC already exists in the table!" + "\n" + "Retype IP Address",
"ERROR", JOptionPane.ERROR_MESSAGE);
return;
} } dispose(); AdminMainWindow.startViewer(jTextFieldIPAdr.getText(),
Integer.parseInt(jTextFieldPort.getText()),
jTextFieldUsername.getText(), String.copyValueOf(jPasswordField.getPassword()),
jCheckBoxSSEnabled.isSelected());
```

#### **IV.IV.VIII.III. Integrated Host Module (For Testing Only)**

This module is accessible via its own tab in the admin module, the functions performed remain same as the host module (William E Perry, 2006).

**IV.IV.VIII.IV. Log tracking and control:** Via the Text Areas in the bottom of the JFrame.

### **V. TEST RESULTS AND EVALUATION**

**V.I. Test Scope and Testing Strategy:** The test scope will examine how each requirement is reflected in the final product result and the testing strategy includes several key points.

**V.II. Administrator:** Functions Reside in the Admin Module:

**V.III. Connect to a remote endpoint with the proper credentials:** Connect Using IP & Port Number, Connection Authentication: Using User Name and Password

**V.IV. Perform Network monitoring functions:** Port Scanner: Discovers open ports in a particular host, Traceroute-Ping: Connectivity and Reliability related tests.

**V.V. Send to and receive messages from the endpoint's operator:** Chat Module: Real-time text chat.

**V.VI. Send Files to the connected remote computer:** Send Files through Chat Interface.

**V.VII. Search for other hosts on the network:** IP Scanner: Discovers other endpoints on the network.

**V.VIII. Employee:** Employee Functions will reside in the Host Module.

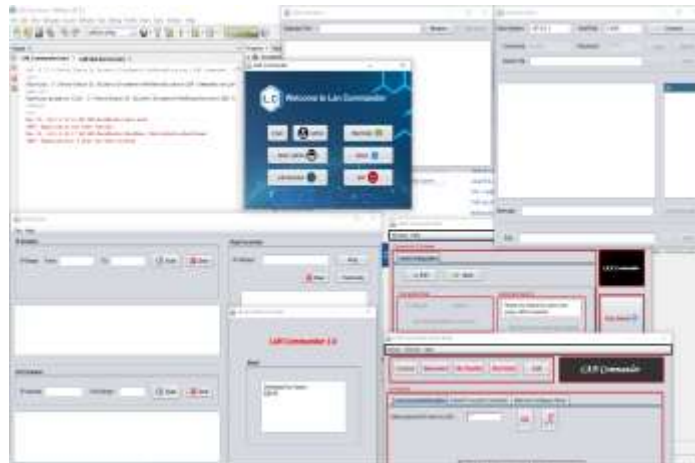
**V.IX. Send message to connected administrator:** Using the Chat Interface.

**V.X. Send error reports:** The Chat Server was designed to remain operational at all times, allow for multiple threads, and is independent from the main program. This Allows the employee to inform the admin at any time of errors when they occur, without needing external methods of communication.

**V.XI. Test Approach:** Run the Program in Its IDE development environment (NetBeans), test all features, starting with the ones specified in the test scope and record the results and create a Jar file using NetBeans and test the features again then record the results.

**V.XII. Release Control:** LAN Commander will have 3 versions, *Admin Version:* Admin Version Contains the Start Admin button in the main menu, *Host (Employee) Version:* Host Version Contains the Start Host button in the main menu and *Test Version:* Start Both button (Host + Admin) will only be included in the test version.

**V.XIII. Phase 1: Simple Run Test for all modules.**



**Fig. 16. Phase 1 Run Test**

**V.XIV. Phase 2: Confirm Functionality for each module**

**V.XIV.I. Chat function**

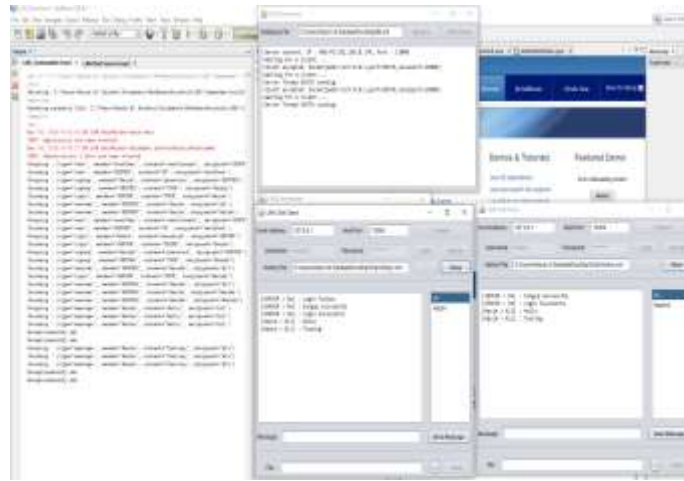


Fig. 17. Phase 2 Chat Test

#### V.XIV.II. LAN Scanner

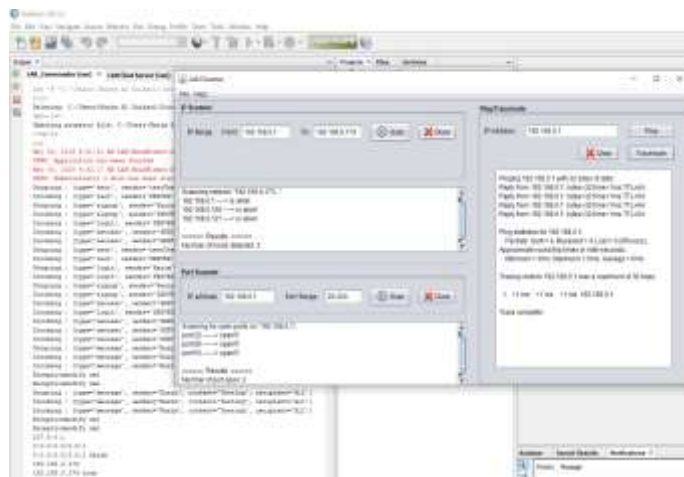
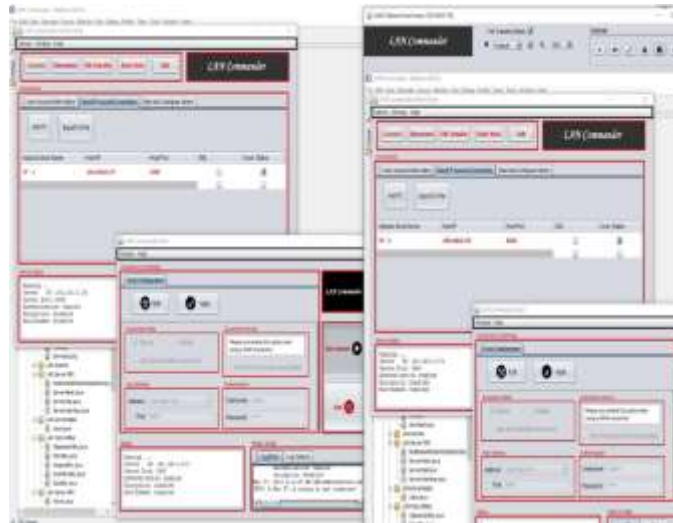


Fig. 18. Phase 2 LAN Scanner Test

#### V.XIV.III. Remote Desktop Control (Host – Admin)



**Fig. 19. Phase 2 Remote Desktop Test**

## **VI. CONCLUSION**

Easy to understand functions with useful tooltips, clearly defined sections, and buttons with expressive icons and it requires java runtime environment and an empty directory. No Installation required. This LAN Commander has little bit weakness part of restricted by firewalls (i.e) LAN Commander will require port forwarding on most routers in order to function. This is a limitation imposed by several factors, including the fact that programs that bypass this requirement are either more complex, with UPhP support, or are rootkits/malware, designed to operate beyond those boundaries. Developing java application using RMI and other challenging concepts proved to be a serious challenges with a technical level like Java Network Programming. Recommendations for further research such as find a way to implement voice chat, research the possibility of incorporating a reverse connection function and research advanced functions of RMI.

## **VII. ACKNOWLEDGMENT**

The authors would like to thank SEGi University Management, Director and staff of Research Innovation Management Centre (RIMC) & Institute of Graduate Studies (IGS) SEGi University, Head of School & Staff of School of Information Technology (SOIT) SEGi University, Dean & Staff of Faculty of Engineering and Built Environment (FOEBE) SEGi University.

## **REFERENCES**

- [1] Rupal D. Bhatt, Dr. D.B.Choksi. (2013), A Comparative Evaluation of Remote Administration Tools. International Journal of Advanced Research in Computer Science, 4(4), 235-240
- [2] Abhishek Kumar Gupta, Sagar Goswami, Ankur Nasa, Dr SC Gupta. (2018). Remote Desktop Monitoring and Controlling. International Research Journal of Engineering and Technology, 5(3), 2026-2027
- [3] Harsh Mittal, Manoj Jain and Latha Banda. (2013). Momonitoring Local Area Network Using Remote Method Invocation”, International Journal of Computer Science and Mobile Computing, 2(5), 50-55.



- [4] Abdullah Al Mamun, Sultan Anwar, Hassan Ali. (2015). Development of Monitoring Tools for Measuring Network Performances: A Passive Approach. *International Journal of Computer Science And Technology*, 6(4), 81-85.
- [5] Wang Ping, Wany Zheng, "IEEE. (2007). Design and Implementation of Open Computer Lab Monitoring and Management system. *Computer and modernization*, IEEE.11, 125-128.
- [6] njRAT Uncovered. (2013). General Dynamics Fidelis Cybersecurity Solutions. [www.fidelissecurity.com](http://www.fidelissecurity.com), 1-27. Retrieved from <https://paper.seebug.org/papers/APT/APT.../fta-1009---njrat-uncovered-1.pdf>
- [7] Debra Littlejohn Shinder. (2001). *Computer Networking Essentials*. Computer Networking Essentials (Cisco Press Core Series), Publisher: Cisco Press.
- [8] Gallardo Raymond. (2013). *The Java Tutorial: A Short Course on the Basics*. Publisher: Pearson, Fifth Edition.
- [9] William Grosso. (2001). *Java RMI*. Publisher: O'Reilly Media, First Edition.
- [10] William E Perry. (2006). *Effective Methods for Software Testing*. Publisher: Wiley Publishing inc, Third Edition.
- [11] Robert K Wysocki. (2009). *Effective Software Project Management*. Publisher: Wiley Publishing inc, Fifth Edition.
- [12] Learn How To RAT (njrat With Full Tutorial) - Science / Technology, Nairaland, Retrieved from
- [13] <https://www.nairaland.com/4343782/learn-how-rat-njrat-full>