Mekelle University – Mekelle Institute of Technology

Network Administrator Tool

A Thesis

By

1. **Berhane Teklu**
2. **Haftom Desbele**
3. **Meresa Gebrehiwot**
4. **Yigaalem Gidey**

Under the guidance of Mr. Fisha Mezgebe and Mr. Tesfamicael Welderufael

Department of **Computer Science and Engineering**

Submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Science in **Computer Science and Engineering and Information Technology(Engineering)**

June 17, 2016

Certificate

The undersigned have examined the thesis entitled **Network Administrator Tool** presented by **Berhane Teklu, Haftom Desbele, Meresa Gebrehiwot and Yirgaalem Gidey** a candidate for the degree of **Bachelor of Science in Computer Science and Engineering and Information Technology(Engineering)** and hereby certify that it is worthy of acceptance.

Advisor's Name and Signature	Date
Examiner's Name and Signature	Date
Examiner's Name and Signature	Date

Acknowledgement

We would like to extend our heartiest thanks with a deep sense of gratitude and respect to all those who provides us immense help and guidance during the development period.

We would like to thank our Project advisor **Mr. Fiseha Mezgebe and Mr. Tesfamicael Welderufael** for guidance, encouragement, understanding and insightful support in the development process. We have been greatly benefited from their regular critical reviews and inspiration throughout our work. We would like to express our sincere thanks to our Head of Department **Mr. Kifle Brhane** for his constant encouragement, suggestions and moral support throughout the duration of this project.

Last but not the least we would like to mention here that we are greatly indebted to each and everybody who has been associated with our project at any stage but whose name does not find a place in this acknowledgement.

Abstract

Network Administrator Tool is a tool for remote administration purposes. It is fast, secure, comfortable to use and platform independent. Now days many organizations use manpower for file transfer, machine shutdown, sharing information. This approach consumes time, energy and money. Most of the recent programs, which provide remote services, have problems like cost, plat form dependency, require expert knowledge. This project deals with the various functionalities of the client systems connected through a network. This enhances the work efficiency of the administrator and also reduces the physical work strain. It can also be used to reduce the unnecessary power consumption in an organization. Network Administrator Tool provides remote service to its entire client over the network. It is developed using Java(Swing) for Front end and XML as Back end. It acts as a network administrator to its client to provide remote service like remote shutdown, remote logoff, remote file transfer, remote desktop view, remote desktop control and Remote Chatting.

Key words: Remote, Java, Xml, Swing

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Admin	**Administrator**
API	**Application Programming Interface**
JRE	**Java SE Runtime Environment**
Java SE	**Java Standard Edition**
Java VM	**Java Virtual Machine**
LAN	**Local Area Network**
MAN	**Metropolitan Area Network**
NAT	**Network Administrator Tool**
OSI	**Open System Interconnection**
RD	**Remote Desktop**
WAN	**Wide Area Network**
XML	**Extensible Markup Language**

CHAPTER ONE

GENERAL INTRODUCTION

1.1 Introduction

Computer network is a collection of computer systems and other devices that can send and receive data from one another communication system, since networking deals with data, managing the other remote computers is equally important in the congested network environment. In today's fast and growing world, every system is connected to network via internet or by a combination of LAN or WAN or MAN. [1][5]

By monitoring other computers on the network we know what others are doing on their system. In an environment where there is an admin, a user present, management and monitoring becomes the important aspect of the network system. The Admin should have control over the remote computer systems. This system provides many features, advantages and may have its applications in many organizations.

So we thought of developing a new type of system called as Network Administrator Tool, NAT software. The operations that can be performed by Network Administrator Tool are: -

1. Remote system log off, shut down and restart
2. Remote files transfer through TCP protocol
3. Remote Desktop Access and Control
4. Remote Chatting
5. Remote Pen Drive Detection

1.2 Statement of the problem

Almost all PC-based systems, that are available at the present times, are all running through the Network based communication. [14] This system is mostly applied in Network centers, many offices, mainly in industries. It is obviously more time consuming to control all these systems connected in a network manually.

- ✓ In the present generation systems, there is a need for the Admin to go all around the network in order to terminate any system that is left non terminated.
- ✓ The Admin has to take all the trouble of going to a particular system to access a file that is needed by him/her.
- ✓ The processes that are running in a particular system can be viewed only in that system itself by using the present generation software's.
- ✓ Existing systems are platform dependent.
- ✓ Existing systems are expensive to purchase and deploy.

1.3 Objectives

1.3.1 General Objective

The objective of this thesis is to develop Network Admin Tool for LANs. NAT application provides remote services to its entire client over the network.

1.3.2 Specific Objectives

- ✓ To remotely shut down, restart, and log off client computer systems
- ✓ To remotely view client desktop
- ✓ To enable file sharing between clients
- ✓ Pen Drive Detection at client systems
- ✓ Chatting app for client systems with chat history

1.4 Scope

Our application, NAT software, has a very wide area of usage. As the number of users increase the scope of our project goes on. There is no limit for this application this can be used to any type of organization by anyone. Each and every module in this project is individual and independent from others. As a result, the Admin's work will be minimized. The System is done in a modular fashion, so any changes can be done easily.

1.5 Significance

The need for time-effective and cost-effective solutions has been the basic advantage of the project. Flexibility, availability, fault tolerance, massive processing power and economic feasibility are also among the advantages of our software application. This application can be the most important tool to monitor activities of employees in a particular office or department and the work running on clients' terminal specific lab work section.

1.6 Limitation

We have to consider some assumptions for proper implementation of NAT software. All IP addresses must fall in the same range and firewall should be turned off for intranet. NAT application is not applicable for remote systems which are disconnected from the network or if they are shutdown.

1.7 Organization

This documentation contains reports on all the processes related to the development of our proposed system. The structure of the document would be as follows:

In chapter 2, an overview and detailed explanation of the background and literature review will be given. Chapter 3 presents the design and implementation of the system. In chapter 4, the results and discussions of the thesis along with prepared screenshots for each module will be discussed. Chapter 5 deals with the conclusion part. In the last chapter, future work and recommendation will be given.

CHAPTER TWO

BACKGROUND AND LITERATURE REVIEW

2.1 Background of the Study

2.1.1 Java

Java is an object oriented, multi thread programming language. It is designed to be small, simple and portable across different platforms as well as operating systems. [2]

source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file contains bytecodes that are the machine language of the Java VM. The java launcher tool then runs the application with an instance of the Java VM.

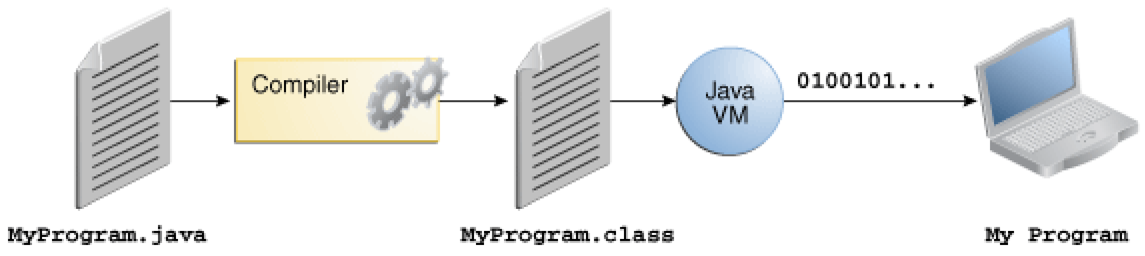


Fig 1: Overview of Software Development process

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, Linux, or Mac OS.

The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms. The Java platform has two components that insulate the program from the underlying hardware.

The Java VM: - basics for the Java platform and is ported onto various hardware-based platforms.

The Java API: - is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces these libraries are known as packages

Why java?

Java was designed to meet the real-world requirements of creating interactive, networked programs. To achieve this, Java supports multithreaded programming, which allows user to write programs that perform many function simultaneously. The Java run-time system enables the user to construct smoothly running interactive systems. One of the main problems is the execution speed of the program. Since Java is architecture neutral it generates bytecode that resembles machine code and are not specific to any processor.

2.1.2 Swings

Swing is a set of classes that provides more powerful and flexible components than are possible with the Abstract Window Toolkit. Even familiar components such as buttons have more capabilities in swing. For example, a button may have both an image and a text string associated with it. Also the image can be changed as the state of the button changes.

2.1.3 XML

XML is a simple text based language which we have used to store data in plain text format. It is a markup language in which we can define our own tags that supports data storage. [8]

Why XML?

XML is human readable language. Because it uses simple text format that is easily understandable. XML is also extensible by providing custom tags that can be created and used very easily. There is any need of a server system to store the data. We can store our data on the local disk using XML.

2.1.4 Java Socket Programming

Java provides the best option for networking through java socket programming. And TCP/IP model is used for communicating devices in network. [7]

TCP/IP model is the enhanced version of OSI model. OSI model is nothing but a standard reference model for data Communication's TCP/IP model is relevant with all platforms. It has five layers.

1. Application Layer

This layer contains the applications, like our NAT application. The applications running in application layer is communicating with the transport layer by giving and receiving data with transport layer.

2. Transport Layer

This layer is dealing with end to end communication. Transport layer contains two protocols, TCP and UDP. However, we have selected TCP for our project. Because it provides connection oriented, reliable communication between two end points and Java can use TCP for networking. Though Java uses UDP for networking as well, UDP provides unreliable communication. The java.net package contains the necessary components for networking.

3. Internet layer

This layer contains the Internet Protocol. It does the routing, packing and addressing of data.

4. Data Link Layer

This layer connects with external network. This layer sends data to external network. It receives data from external network and giving the data to the internet layer.

5. Physical layer

The physical layer consists of the basic networking hardware transmission technologies of a network.

However, Java do not need to be concerned with the details of how data will be routed within a LAN. Definitely, Java does not provide access to the lower-level data link protocols used by LANs. The application layer contains our application and communicates with transport layer, no concern on the other lower layers, which are, data link layer, internet layer, and physical layer.

Computer systems connected to a LAN have their own unique physical or hardware address. This assists other devices on the network in delivering data packets to the correct location or destination. Java provides support for TCP, which can bind network devices together. No matter what type of LAN is used if one is used at all software can be written for it in Java providing it supports TCP/IP.

2.2 Literature Review

2.2.1 TeamViewer

TeamViewer is one of the leading product for remote access and file transfer. This tool has many advantages as well as some disadvantages. All the commercial licenses include features such as file transfer between computer, chat, remote support of unattended servers and computers, and multiplatform support. Disadvantages are Business packages are quite expensive to buy and the program needs fast continuous internet connection otherwise is would be troublesome and frustrating to use. Commercial clients purchase a lifetime license instead of paying a monthly fee. This may not be a good option for customers that do not want to make a large payment or that would like the option to discontinue the software without taking such a large loss. The free version does lack many of the nice features found in the paid versions. [10]

2.2.2 Remote Desktop Protocol (RDP)

Remote Desktop Protocol (RDP) is a proprietary protocol developed by Microsoft, which provides a user with a graphical interface to connect to another computer over a network connection. The user employs RDP client software for this purpose, while the other computer must run RDP server software. Microsoft currently refers to their official RDP client software as remote desktop connection. It allows a user to remotely log into a networked computer running the terminal services server. RDC presents the desktop interface (or application GUI) of the remote system, as if it were accessed locally. [13]

2.2.3 LANVisor

The LANVisor is one of the remote desktop monitoring system that allows you to view and record desktop activities of PCs connected to your network. You will be able to monitor what is happening on user's PCs and record screenshots of all their actions. The program is to be used on the Local Area Network and is used to monitor and record user activity. The LANVisor system is compatible with remote control software, which allows you to control the mouse and keyboard of a computer connected to the network. Some weakness of this software are it is platform dependent it works only in windows and it is still expensive. [11] [12]

CHAPTER THREE

DESIGN AND IMPLEMENTATION

3.1 Overview

Network Admin is a crucial job to monitor and manage systems in a LAN. LAN has different platforms and resources so if the Admin wants to monitor or share the resources on the remote systems in the LAN, operating systems doesn't have compatibility.

3.2 Proposed System

Network Admin Tool (NAT) proposed to provide different remote services to the entire client over the network where the application is installed. These remote services are remote shutdown, remote restart, remote logoff, remote file transfer, remote desktop sharing and remote chatting.

If these remote operation requests, such as remote shutdown, remote restart and remote log off arrive from server, then it parses the request and provides service to its corresponding client.

File transfer operation among clients is done by receiving the request and recognizing the file name. If a user/client wants to transfer file, it sends request to the receiver. After the recipient side replies ok message, the client sends the file. If recipient rejects file transfer, the operation will be stopped.

Remote Desktop (RD) aims to enable the admin to gain access to remote windows desktop in the network. It is a tool we can use to access from anywhere in the network without requiring any native client. RD provides the mechanism by Network admins remotely connects to desktop/client machine.it monitors and views the client machine remotely. It is a tool enabling to know what is doing in computer and handles the basic problem of the client.

Remote Chatting (RC) is a small application we can use to facilitate communication between different hosts on the same LAN with chat history stored on local hard disk. It does not require a central server and uses very little bandwidth. RC provides user login that handles multiple users at the same time and also provides them to send and receive private and public messages. Chatting aims to pass data to and from applications over the local network which makes the data synchronization simpler. It allows users across the network to exchange data in real time.

Pen drive detection defines the detection of the pen drive port. When the external device is added to the client systems, then the server will get a message from that particular client system with IP address. This helps the server to detect or to get information about that.

The proposed system overcomes the disadvantages of the existing systems.

- ✓ Using the NAT software, the admin can control the overall operations of the remote system from the admin’s system itself.
- ✓ Enabling the admin terminate the operations on the remote systems.

3.3 Feasibility Study

Every project can be feasible if provided with unlimited resources and infinite time. However, the development of software application is affected by the scarcity of resources and difficult delivery rates. We have assumed three key considerations that are involved in our feasibility analysis.

3.3.1 Technical Feasibility

The NAT software is developed using the java environment. The reason for using java, as the development platform is that, java is an Object Oriented Language which handles most of the networking concepts. Since java is a platform independent language, the class files can be executed on any operating system easily.

3.3.2 Economic Feasibility

This is the most frequently used method for evaluating the effectiveness of a system. It is also called as a cost analysis. The NAT project, which is used to control all the remote systems in a network, requires resources such as the software and hardware components that support the Remote Control through java project effectively. Since all the clients are usually connected to the server in any organization, it reduces the cost factor. Hence there is no need of further physical connection to be established between the server and the client.

3.3.3 Operational Feasibility

The NAT project is a user-friendly tool developed in order to make the operations of the Admin much better. It will be easy for the Admin to handle all the systems in the network from the server itself which helps in increasing the operational efficiency of the Admin.

3.4 Requirement Description

3.4.1 Functional Requirements

- ✓ Private and public chat
- ✓ Chat history saved in XML format at local disk
- ✓ Private file transfer
- ✓ Remote Desktop View, mouse and key emulation
- ✓ Remote control such as Shutdown, Restart and log off
- ✓ Remote Pen drive detection when it plugged and unplugged

3.4.2 Non-Functional Requirements

- ✓ The system responds in less than seconds.
- ✓ It displays error messages when there is incorrect data or unauthorized users.
- ✓ The system is secure for unauthorized user.
- ✓ Multiple users can access at the same time.
- ✓ The system has user friendly and interactive user interface.

3.5 System Design

3.5.1 Design Methodology

The main focus of our system design is on the specifying which modules are needed for NAT system, clear specification of each module and also on how these modules are interconnected. Our system design considers the properties of software design like consistency, completeness, simplicity and understandability.

3.5.2 Data Flow Diagram

Top level DFD: -

- ✓ To understand interaction of NAT with the users
- ✓ To understand the final output that is generated in two parts of it

Part 1: Client system

Part 2: Server system

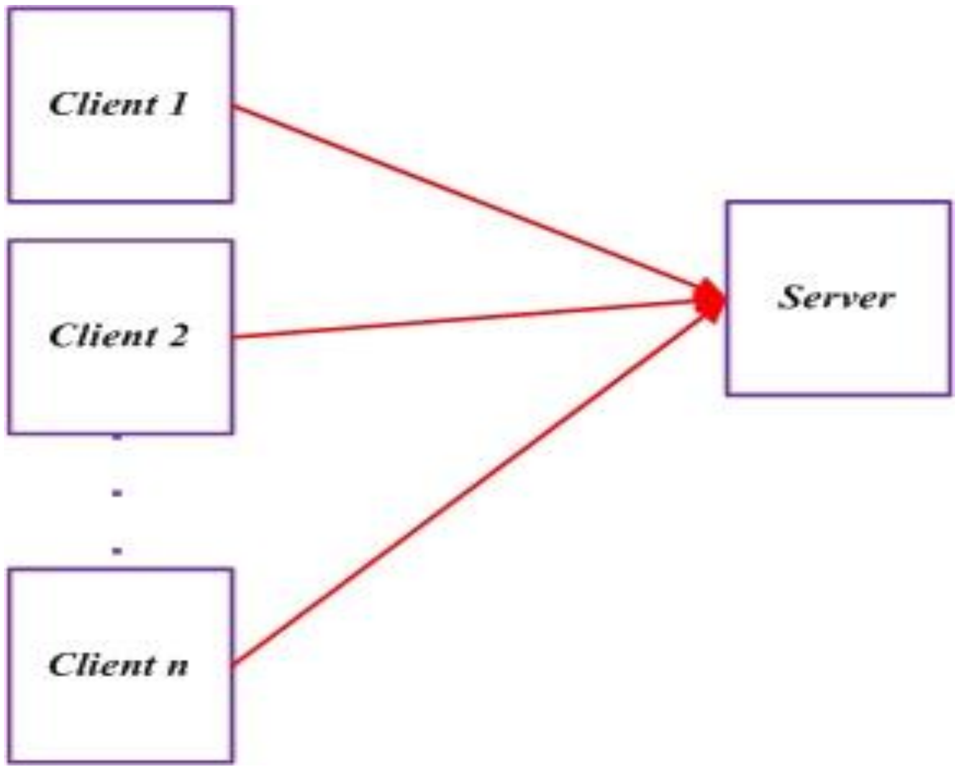


Fig 2: Top level DFD

Low level DFD

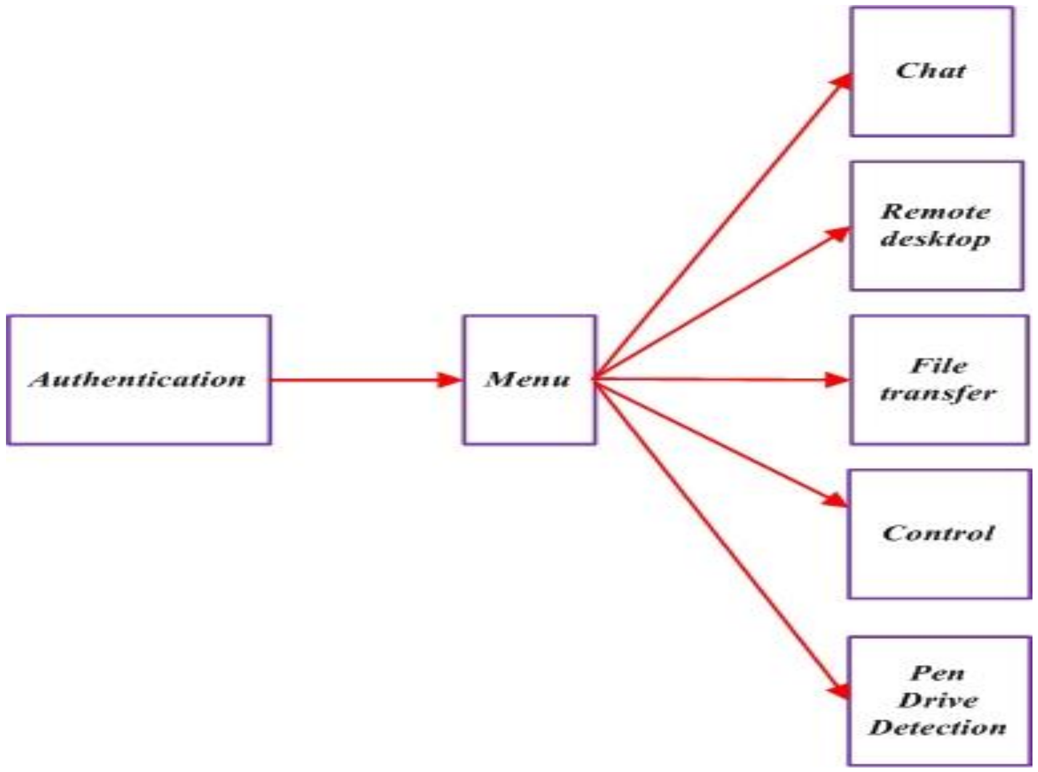


Fig 3: Low level DFD

3.5.3 Class Diagram

NAT class diagram is a structure with both variables and methods that shows a set of classes, interfaces and collaborations and their relationships. This is used to model our object oriented system and to give static view of our system. The class diagram of our project is shown below. Each block contains class name, variables and methods.

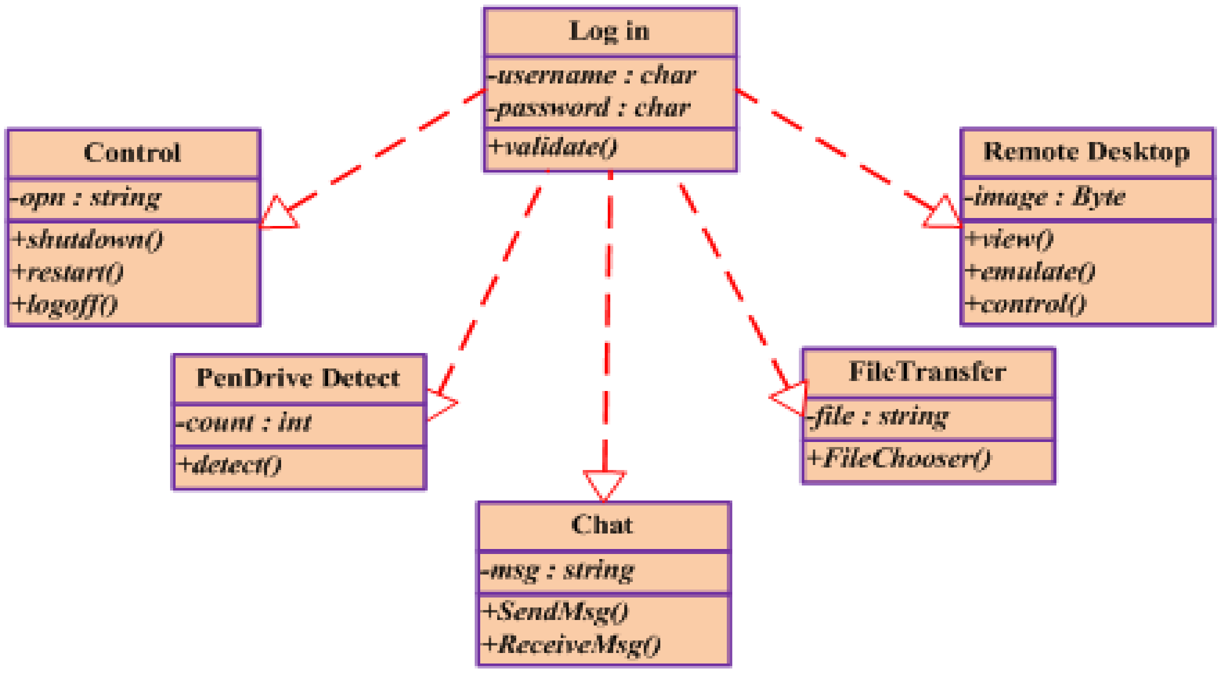


Fig 4: Class diagram for NAT

3.5.4 Use case Diagram

The use case diagram for NAT software shows use cases and actors and their relationship. Use case describes a set of sequence of actions. The actor represents real world object.

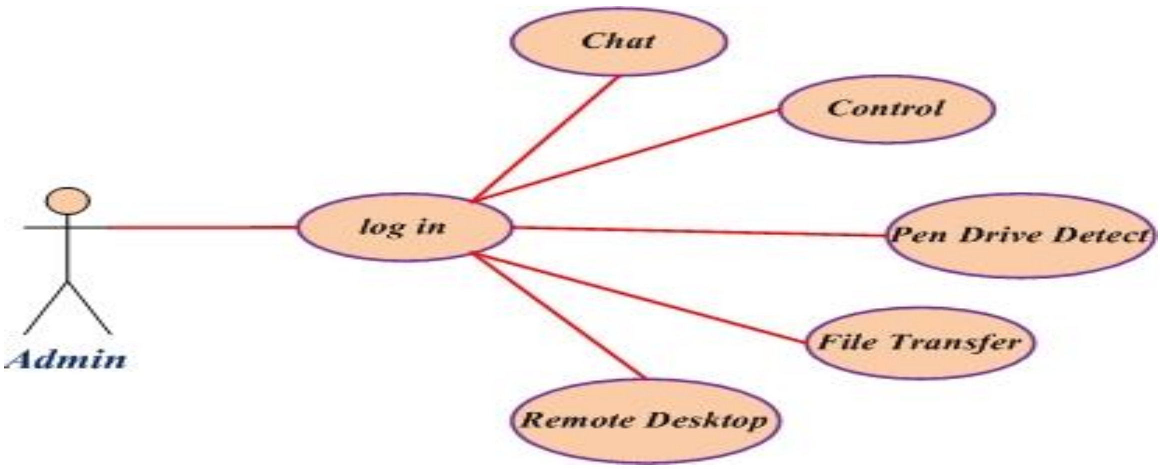


Fig 5: Use case diagram for NAT

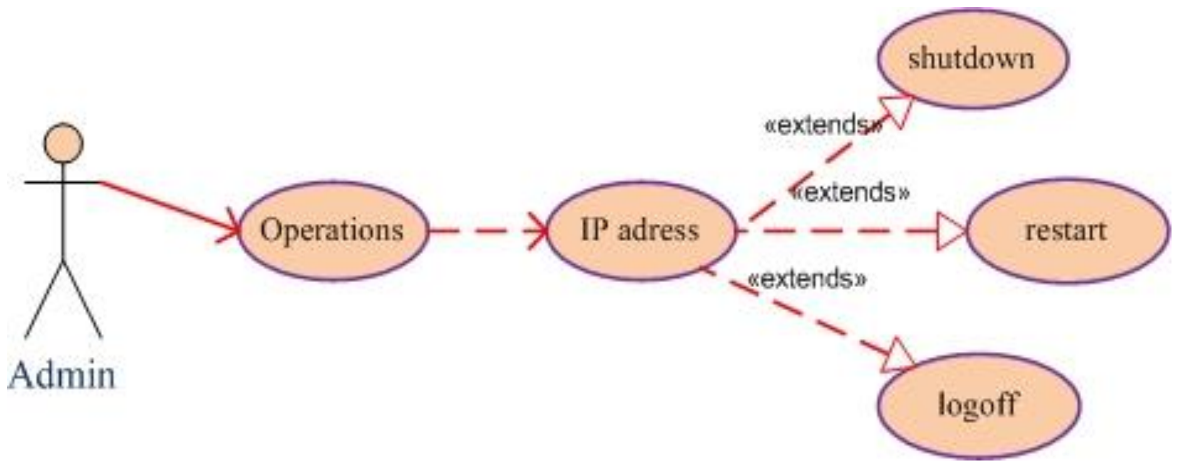


Fig 5: Use case diagram for Control module

3.5.5 Description of Use cases

Name:	Identifier:	Description:
Control Client Operation	UC1	Used to shutdown, restart, log off client
Detect Pen Drive	UC2	Used to detect pen drive on client
View Client	UC3	Used to view client desktop
Emulate key & mouse	UC4	Used to control client desktop
Chat with Client	UC5	Used for messaging among clients
Transfer File	UC6	Used to share file among clients

Table 1: Description of NAT use cases

These use cases will be described briefly below.

UC1	
Participating Actor	Server
Pre-condition	The log in form displayed on screen. The Actor has logged on NAT and has an authorization for controlling
Flow of events	1. Server connected with clients 2. The system displays IP list of clients 3. Server chooses operation 3.1. Shutdown, or 3.2. Restart, or 3.3. Log off
Post condition	Operation executed at client system

Table 2: Use case Name: Control client operation

UC2	
Participating Actor	Server
Pre-condition	The log in form displayed on screen. The Actor has logged on NAT and has an authorization for detecting pen drive
Flow of events	1. Server connected with clients 2. Server gets information from client 3. Information sent to server display
Post condition	Pen drive at client system will be detected by server

Table 3: Use case Name: Pen Drive Detection

UC3	
Participating Actor	Server
Pre-condition	The log in form displayed on screen. The Actor has logged on NAT and has an authorization for viewing client desktop
Flow of events	1. Server connected with clients 2. Client send desktop information [image format] 3. Server stores clients' information on internal frame
Post condition	Server views clients' desktop information

Table 4: Use case Name: View Client

UC4	
Participating Actor	Server
Pre-condition	The log in form displayed on screen. The Actor has logged on NAT and has an authorization for emulating
Flow of events	1. Server connected with clients 2. Client send desktop information [image format] 3. Server stores clients' information on internal frame 4. Server views clients' desktop information 5. Server controls clients' desktop using clients' mouse and key emulation
Post condition	Server emulates the clients' mouse and key

Table 5: Use case Name: Emulate Key & mouse

UC5	
Participating Actor	Client
Pre-condition	The log in form displayed on screen. The Actor has connected with server and logged on NAT and has an authorization for chatting
Flow of events	1. NAT provides list form of clients 2. Client selects name from the list 3. NAT provides chat form for the client 4. Client send message for other client/s on the list 5. Client receives message from other client/s
Post condition	Message saved to chat history

Table 6: Use case Name: Chat with client

UC6	
Participating Actor	Client
Pre-condition	The log in form displayed on screen. The Actor has connected with server and logged on NAT and has an authorization for file transfer
Flow of events	1. NAT provides list form of clients 2. Client selects name from the list 3. NAT provides chat form for the client 4. Client browses file from to transfer 5. Sender transfer file for other client/s on the list 6. Receiver approves file transfer 6.1. Accept 6.2. Reject 7. Receiver shares file from sender (if it accepts)
Post condition	Operation saved to chat history

Table 7: Use case Name: Transfer File

3.5.6 Sequence Diagram

It is an interaction diagram which focuses on the time ordering of messages. Sequence diagram for NAT software is like a table showing objects on X-axis and messages ordered in increasing time along Y-axis. The objects are ordered from left to right. That means the object that initiates the interaction is placed at the left, then increasingly more sub-routine objects at the right. The message sent and received by objects is placed along the Y-axis in order of increasing time from top to the bottom. Therefore, the reader will have a clear visual understanding to the flow of operations over time.

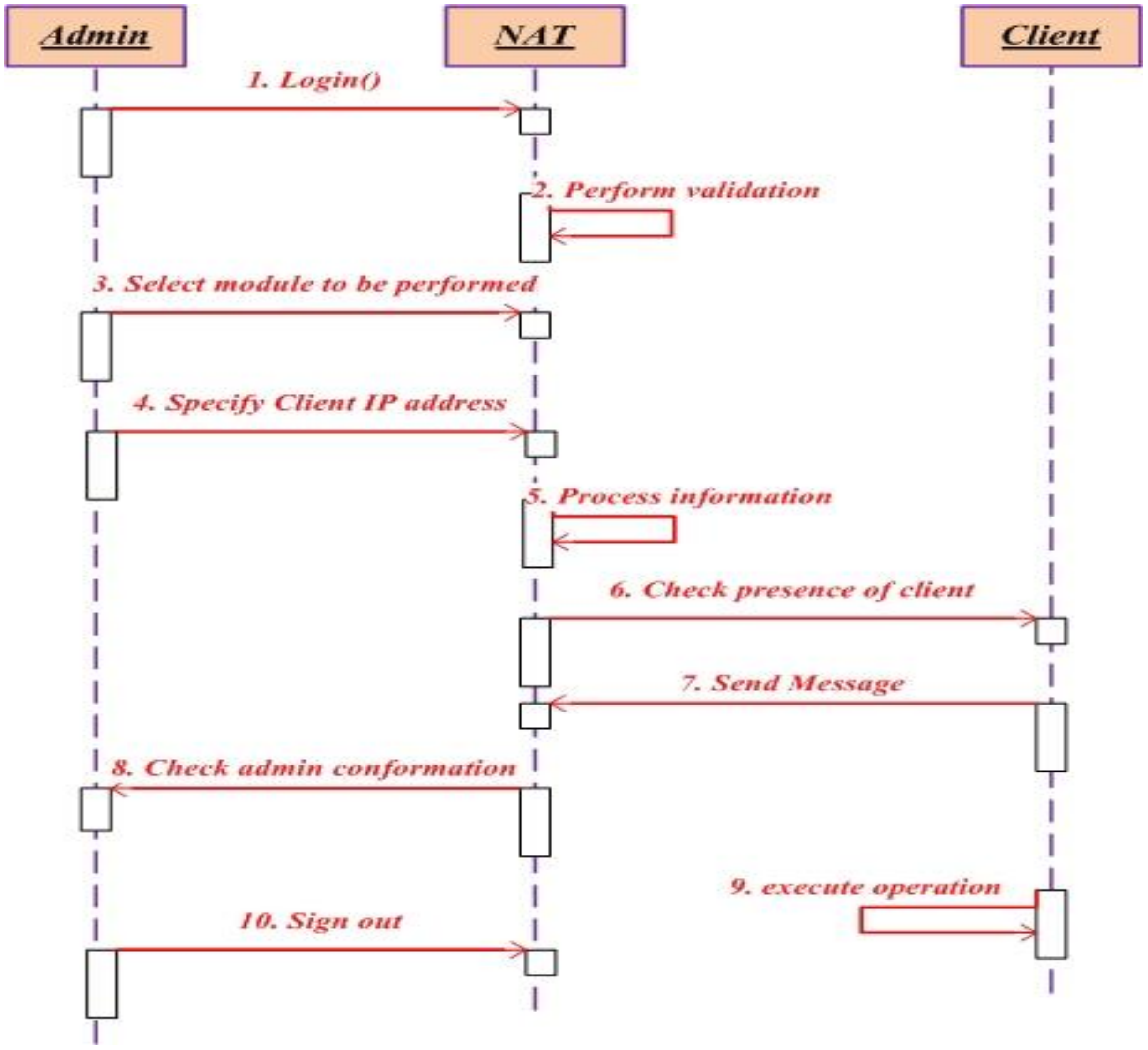

Admin
NAT
Client
1. Login()
2. Perform validation
3. Select module to be performed
4. Specify Client IP address
5. Process information
6. Check presence of client
7. Send Message
8. Check admin conformation
9. execute operation
10. Sign out

Fig 6: Sequence diagram for NAT

3.5.7 Activity Diagram

It shows the flow from one activity to another activity of NAT. An activity is on progressing and non-atomic execution with in a state machine. It is a collection of vertices and arcs

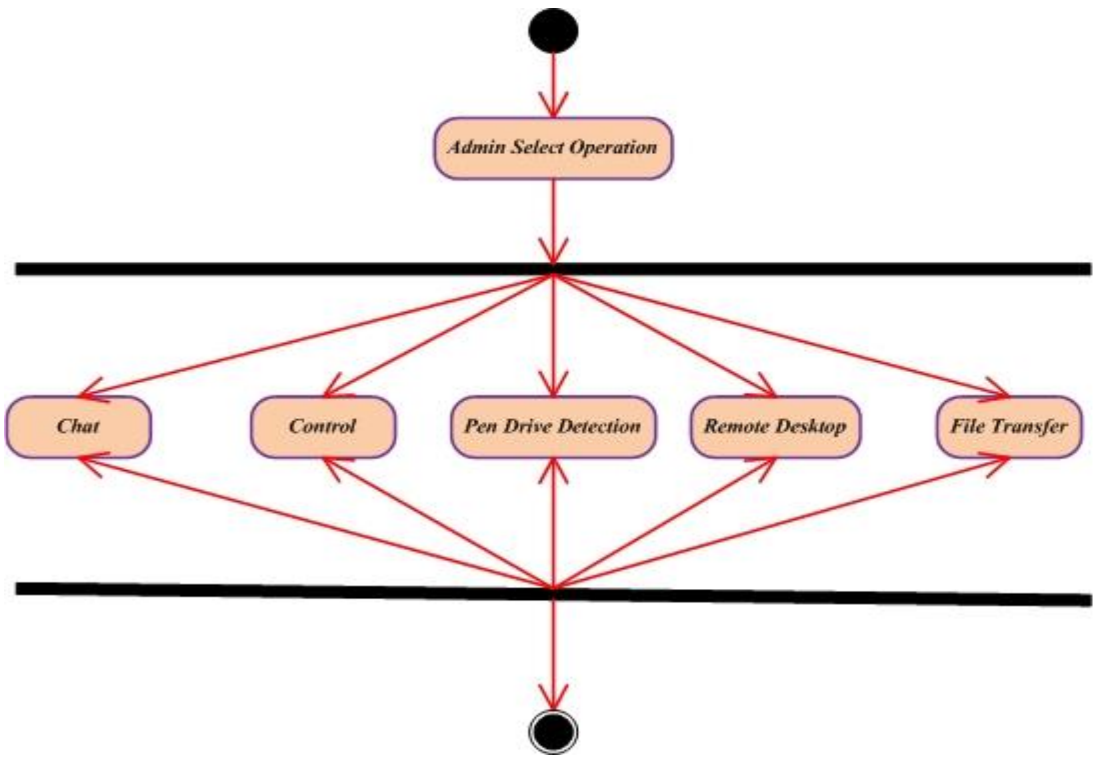


Fig 7: Activity diagram for NAT

3.5.8 Flow Chart Diagram

It represents the work flow or process by showing the steps as boxes of various kinds, and their order by connecting them with arrows. It is considered as a solution model to a given problem. We used flow charts in analyzing, designing and managing a process or a program in different fields.

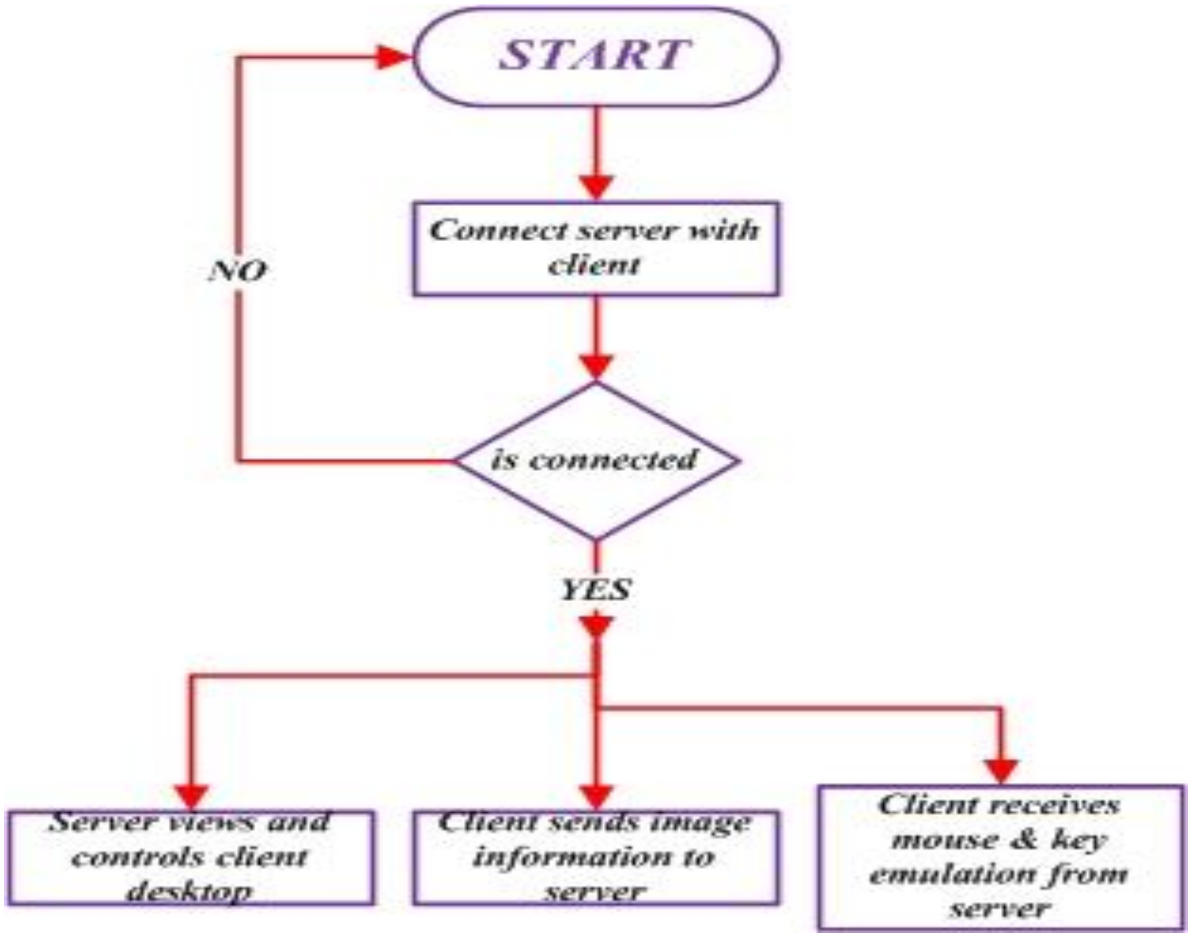

START
Connect server with client
NO
is connected
YES
Server views and controls client desktop
Client sends image information to server
Client receives mouse & key emulation from server

Fig 8: Flow Chart for remote desktop [view and control]

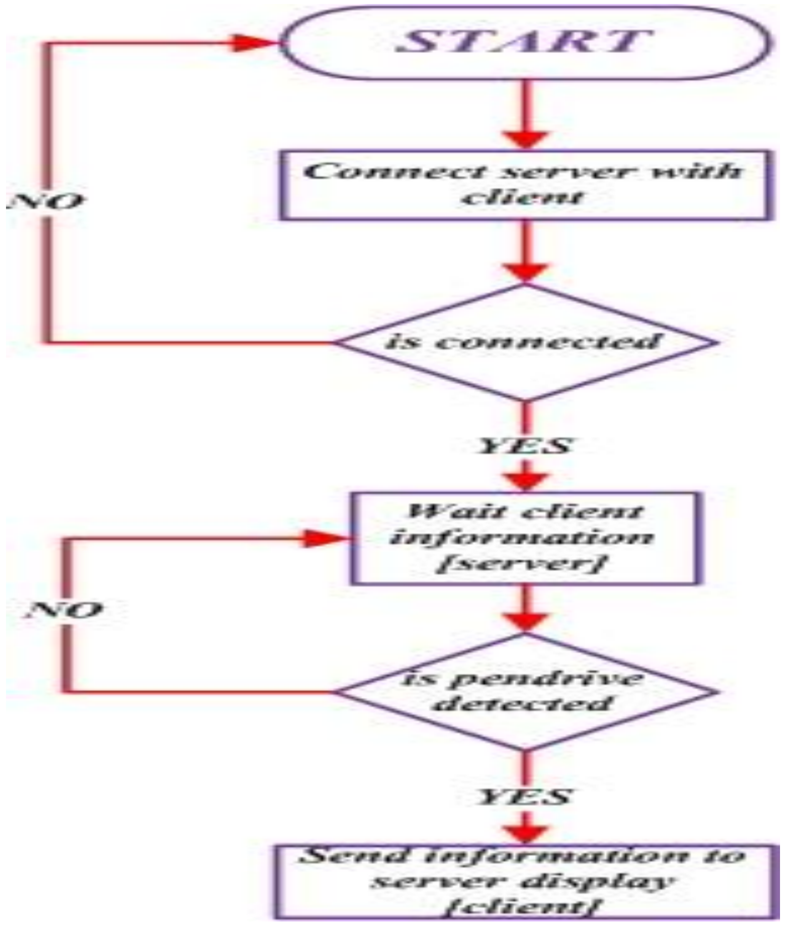

START
Connect server with client
is connected
NO
YES
Wait client information [server]
is pendrive detected
NO
YES
Send information to server display [client]

Fig 9: Flow Chart for Pen Drive Detection

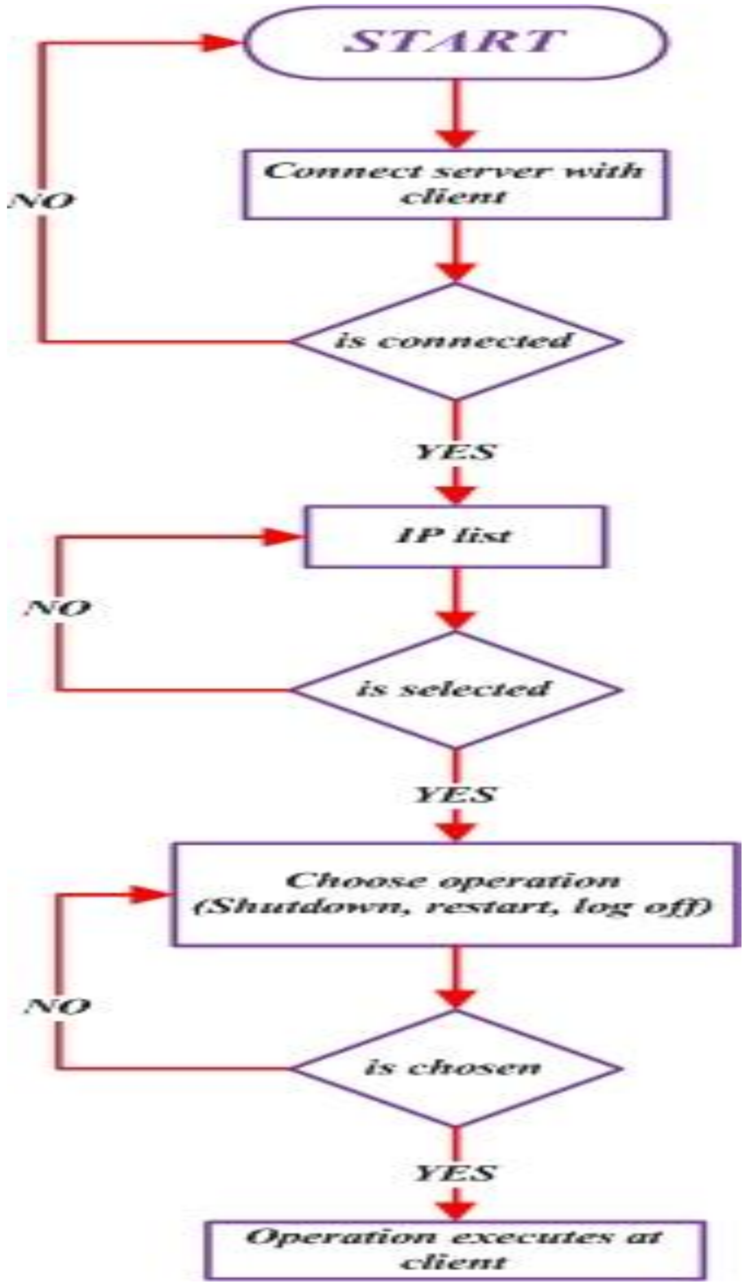

START
Connect server with client
NO
is connected
YES
IP list
NO
is selected
YES
Choose operation (Shutdown, restart, log off)
NO
is chosen
YES
Operation executes at client

Fig 10: Flow Chart for Control

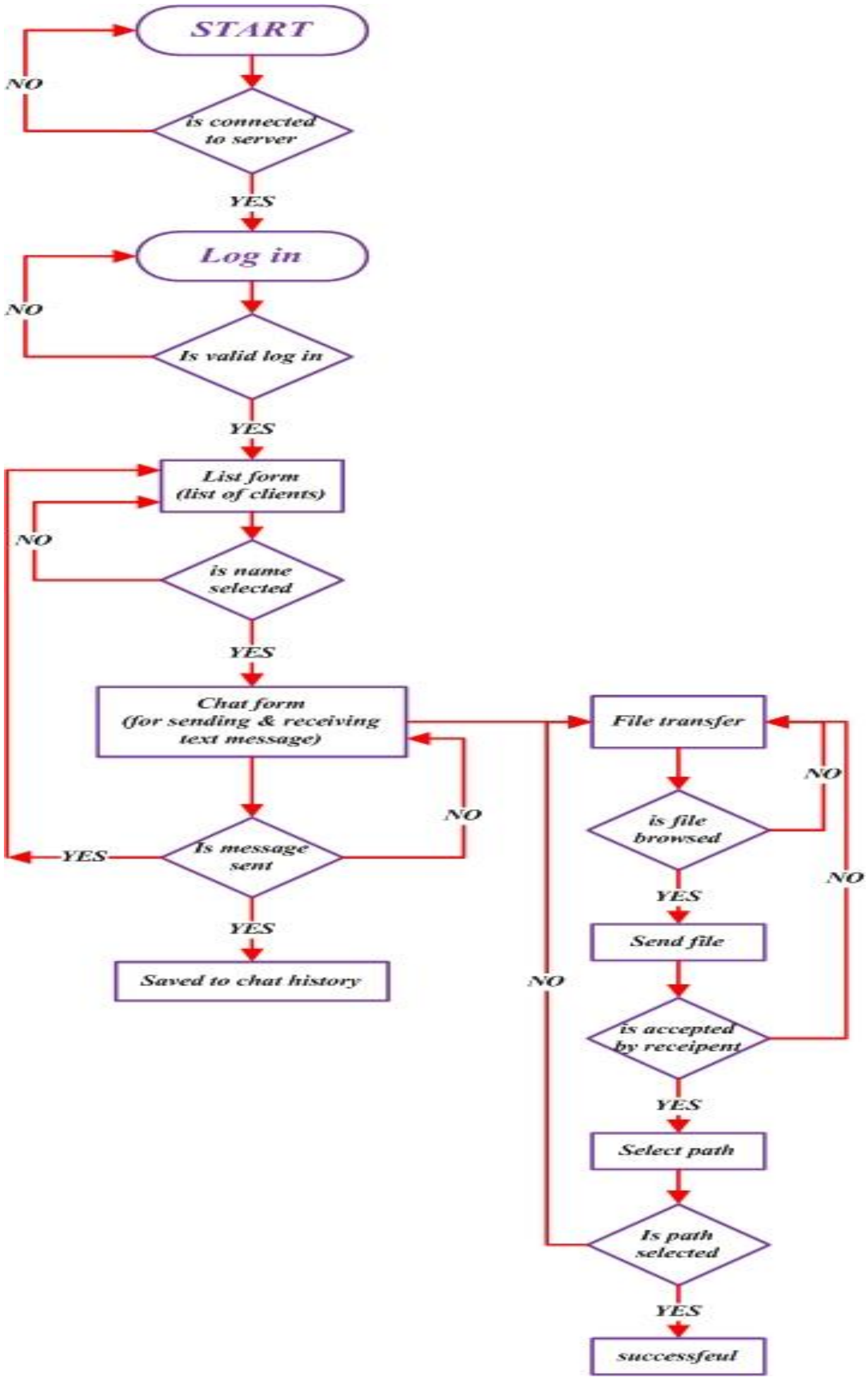


Fig 11: Flow Chart for Chat and File transfer

3.5.9 Deployment Diagram

NAT used deployment diagram for modeling the aspects of Object Oriented System. It shows the configuration of Runtime processing nodes and the components that live on them.

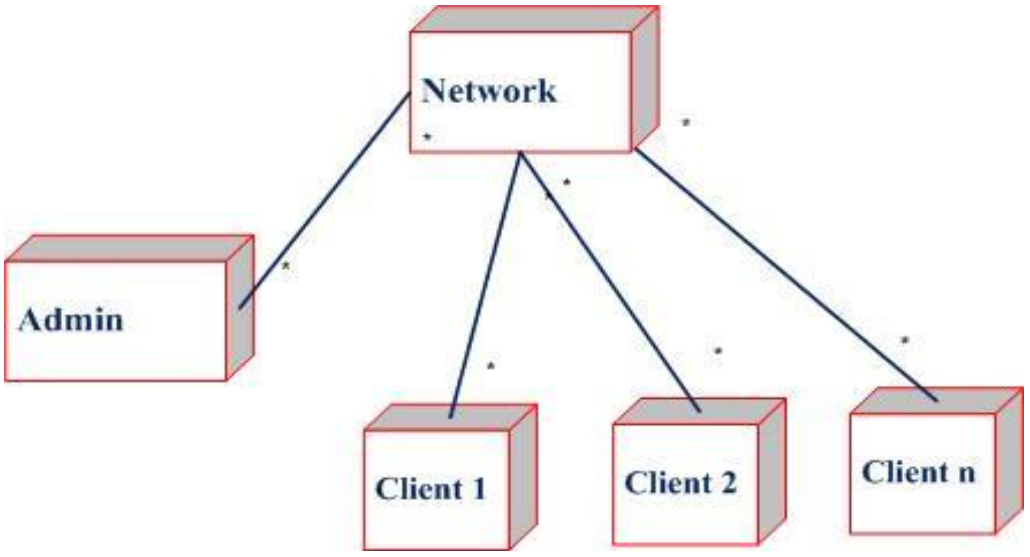


Figure 12: Deployment diagram for NAT

3.6 Development Tools

For developing the NAT software, we have used the following development languages:

- ✓ JAVA, SWINGS as front end programming language
- ✓ XML as back end data storage tools
- ✓ NetBeans as Integration Development Environment.
- ✓ Microsoft Office Visio as system design tool
- ✓ Microsoft Office Word for documentation too

3.6.1 System Specifications

Software Requirements	
Operating Platform	Any platform where JRE is installed
Front End	JAVA, SWINGS
Back End	XML as Data Storage tool

Open Specific ports: To prevent from firewall in Windows	
Port Number: 3000	For Remote Control
Port number: 4444	For Pen Drive Detection
Port number: 3200	For Remote Desktop
Port Number: 1300	For Remote Chatting
Hardware Requirements	
Processor	PENTIUM-4 or +, 128MB RAM, 10 GB HDD
RAM	256 (Minimum)
HADR DISK	20 GB (Minimum)
LAN	ENABLED

Table 8: System Specifications for NAT

3.7 System Implementation

3.7.1 Remote Desktop

We have used Robot Class of Java.awt to capture images and send to server using java.net.Socket Class.

3.7.2 Remote Control

Allows a server to shutdown, restart and log off to all connected clients. We Use java.net.Socket to send operations from server and Runtime.getRunTime.exec() method to execute server operations.

3.7.3 Remote Pen Drive Detection

The client program looks for any Pen Drive and sends to the Server if plugged or unplugged Pen drive is detected.

3.7.4 Remote Chat and File Transfer

We have used java.net.Socket to send Messages and files jFileChooser to show Files from client computer system.

The detailed implementation of each module is laid out in the Appendices section of the thesis.

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 Remote Desktop Module

We have tested the remote desktop module of the NAT software in Windows 8.1 and Ubuntu 14.04 (Linux). We have seen that it works well on both operating systems effectively. We make the delay to be small, in order to prevent network congestion, to improve the network performance and error free results.

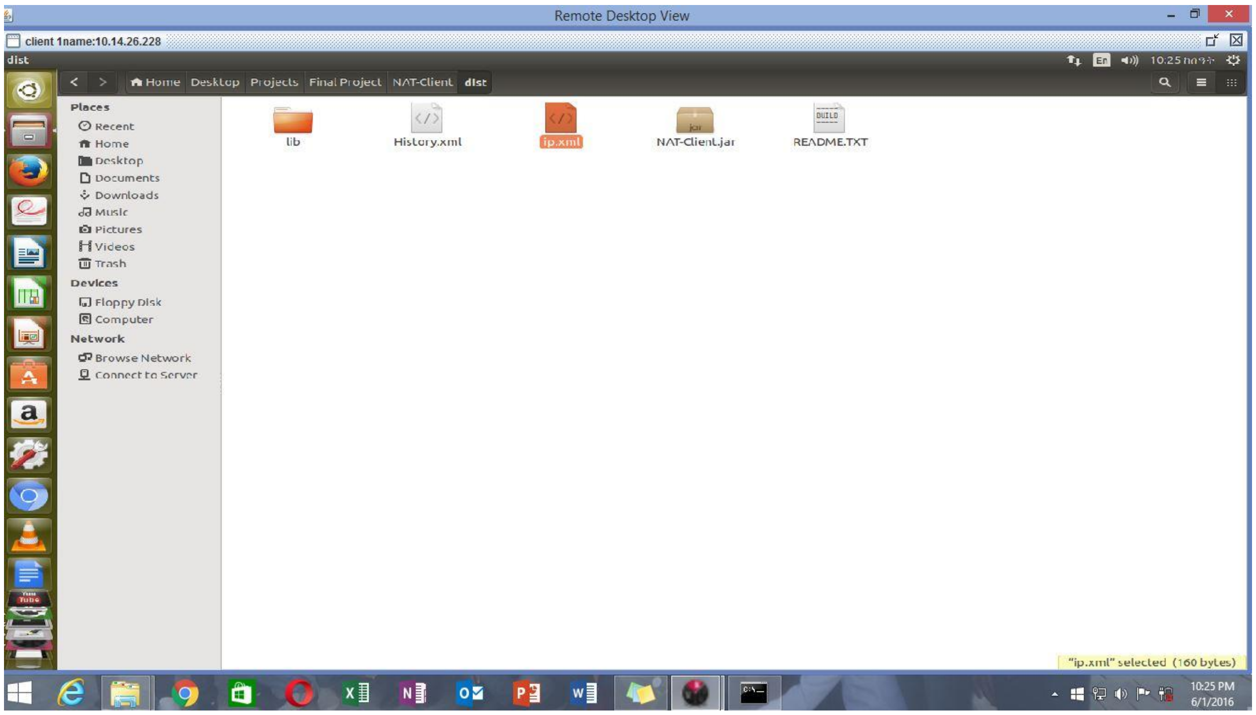


Fig 13: Remote Desktop View of Ubuntu 14.04 Client on Windows 8.1 server

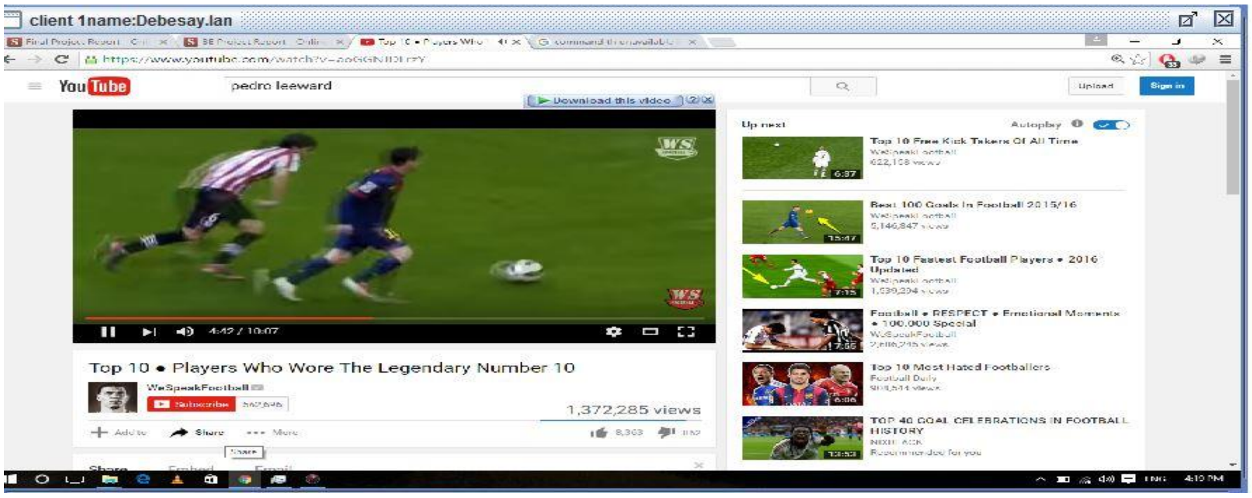


Fig 14: Remote Desktop View of Windows 10 Client on Windows 8.1 server

4.2 Remote Control Module

We have tested the remote control module of the NAT application by specifying available clients IP address. We had able to shutdown, restart, and logoff the remote clients for both Windows and Ubuntu (Linux).

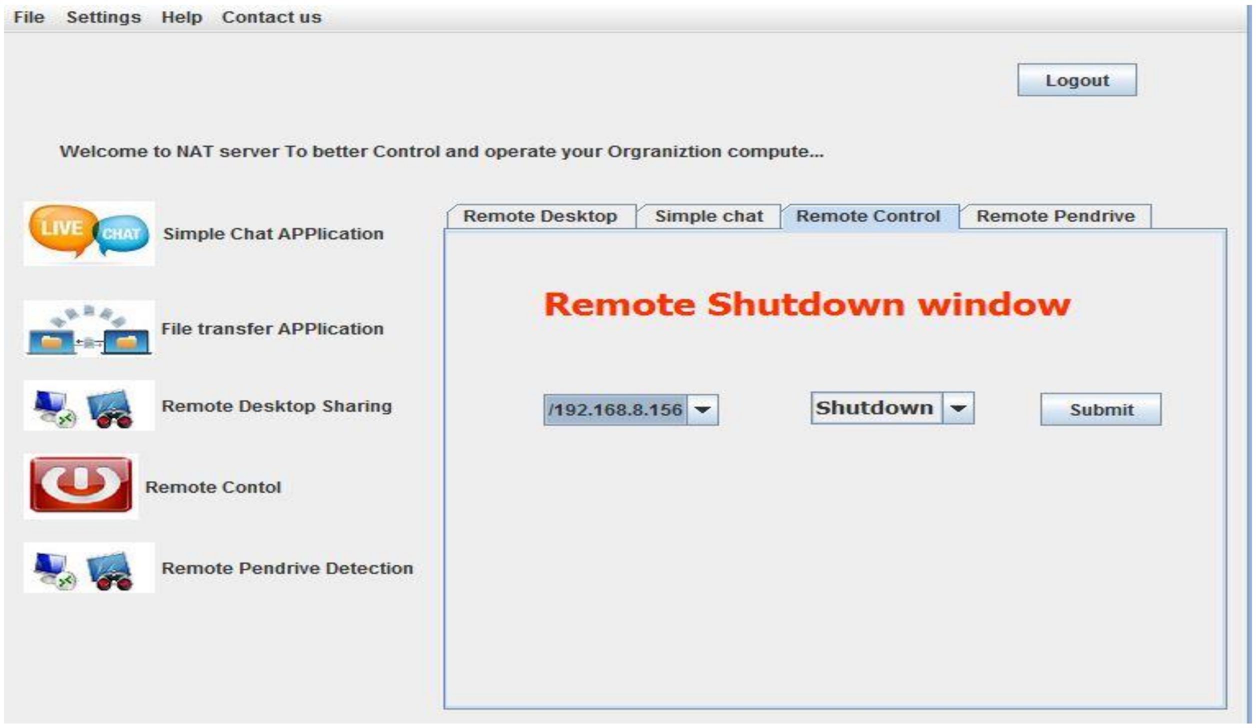
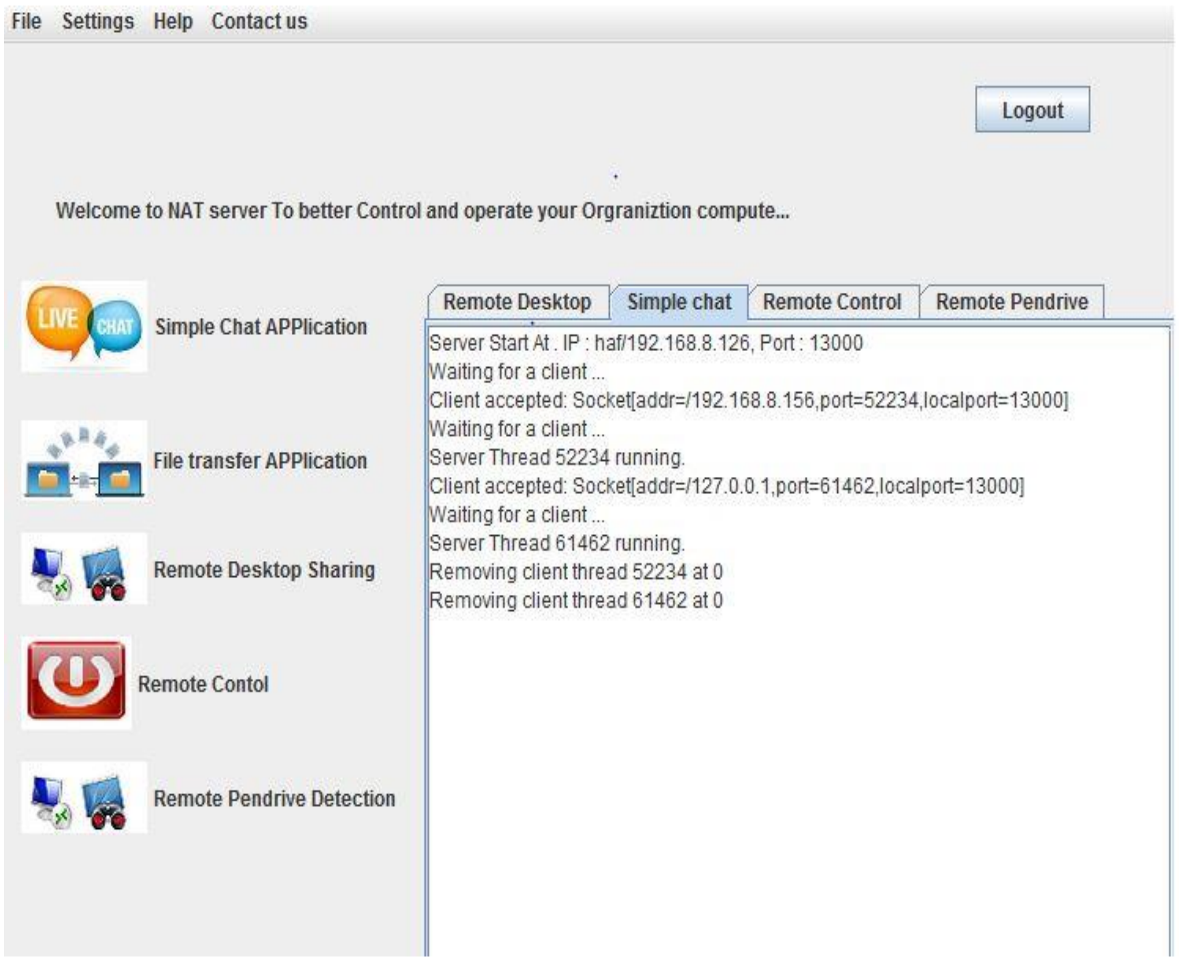


Fig 15: Remote control: Shut down the client with IP 192.168.8.156

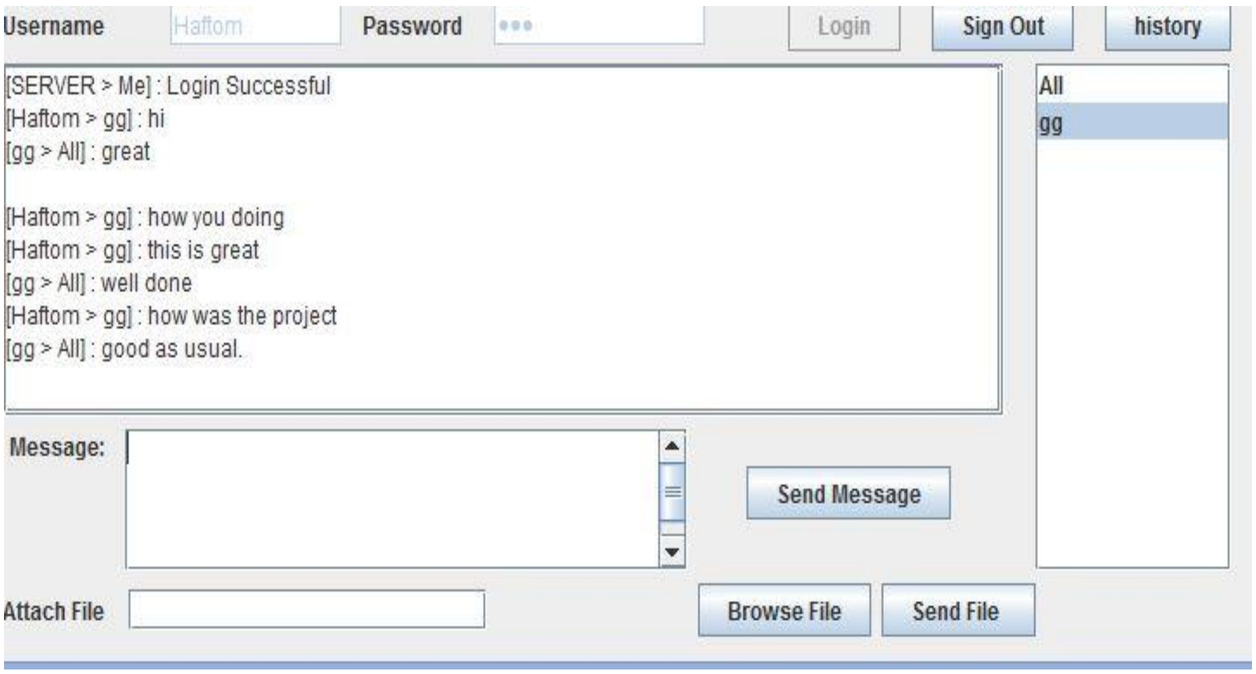
4.3 Chat and File Transfer Module

This module works properly for private(one-to-one) and public (with all) chat as well as file transfer among client systems. The chat history stored in the local directory of the client system successfully.



```
Server Start At . IP : haf/192.168.8.126, Port : 13000
Waiting for a client ...
Client accepted: Socket[addr=/192.168.8.156,port=52234,localport=13000]
Waiting for a client ...
Server Thread 52234 running.
Client accepted: Socket[addr=/127.0.0.1,port=61462,localport=13000]
Waiting for a client ...
Server Thread 61462 running.
Removing client thread 52234 at 0
Removing client thread 61462 at 0
```

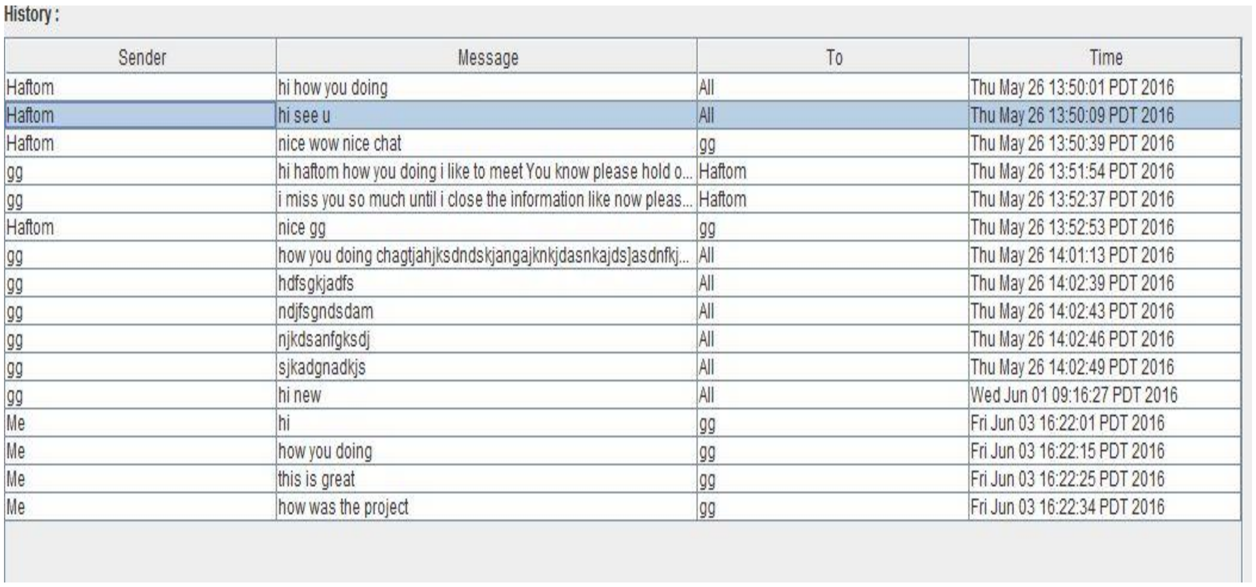
Fig 16: Chat module: list of multiple clients connected to server



[SERVER > Me] : Login Successful
[Haftom > gg] : hi
[gg > All] : great

[Haftom > gg] : how you doing
[Haftom > gg] : this is great
[gg > All] : well done
[Haftom > gg] : how was the project
[gg > All] : good as usual.

Fig 17: Chat and file transfer module: Clients can chat and attach file with single client and can chat with all clients

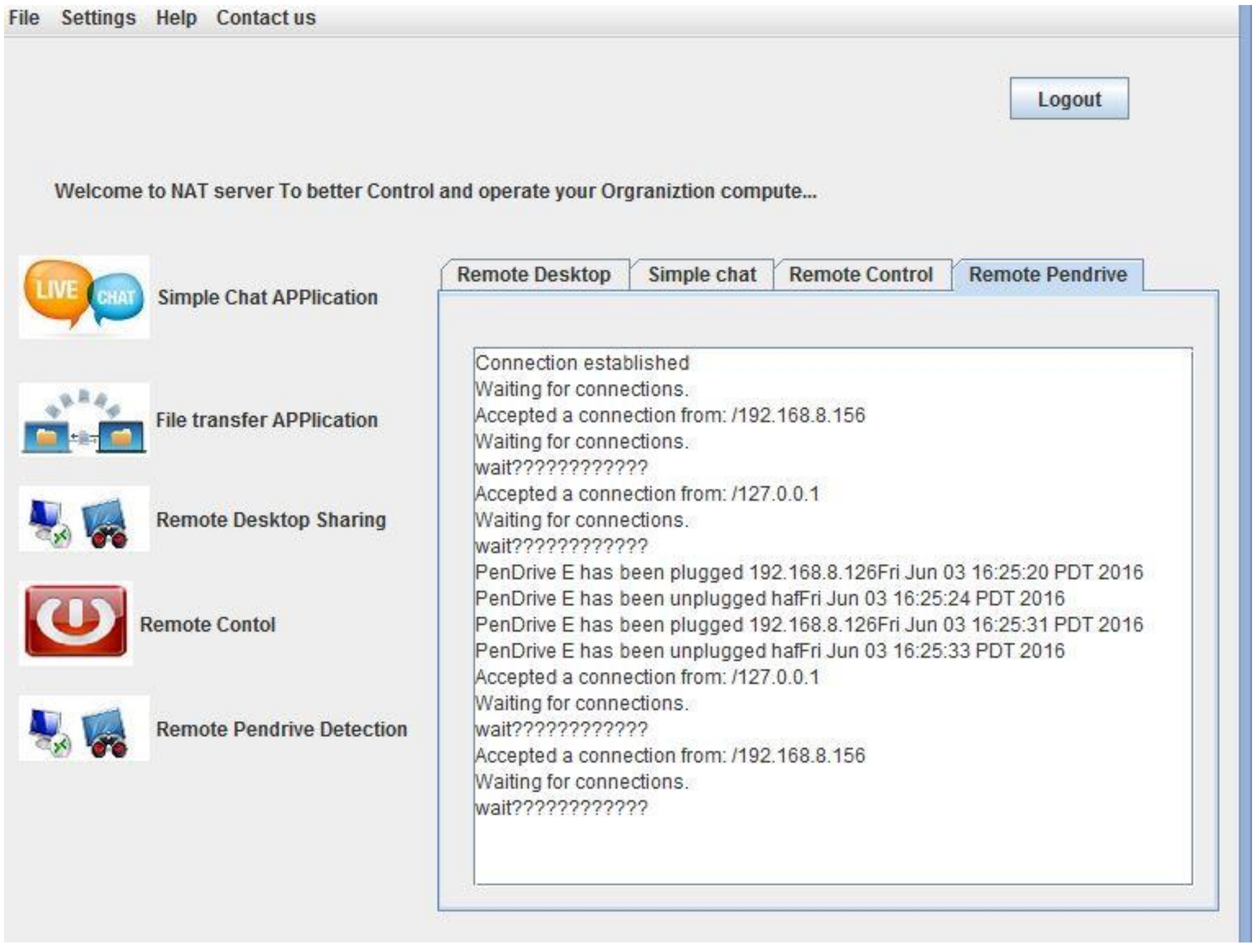

History :

Sender	Message	To	Time
Haftom	hi how you doing	All	Thu May 26 13:50:01 PDT 2016
Haftom	hi see u	All	Thu May 26 13:50:09 PDT 2016
Haftom	nice wow nice chat	gg	Thu May 26 13:50:39 PDT 2016
gg	hi haftom how you doing i like to meet You know please hold o...	Haftom	Thu May 26 13:51:54 PDT 2016
gg	i miss you so much until i close the information like now pleas...	Haftom	Thu May 26 13:52:37 PDT 2016
Haftom	nice gg	gg	Thu May 26 13:52:53 PDT 2016
gg	how you doing chagtjahjksdndskjangajknkjdasnkajds]asdnfkj...	All	Thu May 26 14:01:13 PDT 2016
gg	hdfsgkjadfs	All	Thu May 26 14:02:39 PDT 2016
gg	ndjfsgndsdam	All	Thu May 26 14:02:43 PDT 2016
gg	njkdsanfgksdj	All	Thu May 26 14:02:46 PDT 2016
gg	sjkadgnadkjs	All	Thu May 26 14:02:49 PDT 2016
gg	hi new	All	Wed Jun 01 09:16:27 PDT 2016
Me	hi	gg	Fri Jun 03 16:22:01 PDT 2016
Me	how you doing	gg	Fri Jun 03 16:22:15 PDT 2016
Me	this is great	gg	Fri Jun 03 16:22:25 PDT 2016
Me	how was the project	gg	Fri Jun 03 16:22:34 PDT 2016

Fig 18: Chat module: View saved Chat history from local disk

4.4 Pen Drive Detection Module

This module detected all the plugged in and/or plugged out pen drives in the client systems. For both Windows and Ubuntu (Linux).



```
Connection established
Waiting for connections.
Accepted a connection from: /192.168.8.156
Waiting for connections.
wait????????????
Accepted a connection from: /127.0.0.1
Waiting for connections.
wait????????????
PenDrive E has been plugged 192.168.8.126Fri Jun 03 16:25:20 PDT 2016
PenDrive E has been unplugged hafFri Jun 03 16:25:24 PDT 2016
PenDrive E has been plugged 192.168.8.126Fri Jun 03 16:25:31 PDT 2016
PenDrive E has been unplugged hafFri Jun 03 16:25:33 PDT 2016
Accepted a connection from: /127.0.0.1
Waiting for connections.
wait????????????
Accepted a connection from: /192.168.8.156
Waiting for connections.
wait????????????
```

Fig 19: Pen Drive Detection from Client with IP 192.168.8.126 with plugged and unplugged time

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In the present generation systems, the Admin has to go all around the network in order to terminate any system that is left non-terminated. The processes that are running in a particular system can be viewed only in that system itself by using the present generation software. NAT is very useful for monitoring client's activity and for solving the above problems.

Windows features of remote desktops and file sharing are platform dependent, requires expert knowledge and many steps to set up. Even though some soft wares are available on the market they are expensive and/or increases network congestion.

Network Administrator Tool can be considered as one of the best tools for remote administration purposes. It is fast, secure, comfortable to use and platform independent. This enables a network Admin to work on any computer in the local area network (Intranet) or connected to the Internet.

5.2 Recommendation

We have developed this NAT software application with client program and server program that will be installed separately and independently on both server and client machines. The five modules of our application are independent. But for simplicity and easily usage we have modularized them for providing great advantage for users of the software.

We would like recommend Offices in all areas and different organizations with multiple clients and multiple employees connected to LAN to use this effective and user friendly tool.

5.3 Future Works

- ✓ We will include group chat feature
- ✓ We will extend our software to enable video conferencing and audio chatting
- ✓ We will add multiple file transfer
- ✓ We will include remote desktop sharing feature
- ✓ Taking measure actions to detected Pen Drive

References

1. Reilly, David, and Michael Reilly. *Java network programming and distributed computing*. Addison-Wesley Professional, 2002.
2. Naughton, Patrick, and Herbert Schildt. "The complete reference java 2." (2003).2. Core java2 (Advanced Features) -Horstmann Cornell SUN Micro Systems.
3. Deitel, Paul, and Harvey Deitel. *Java How to program*. Prentice Hall Press, 2011.
4. Harold, Elliotte. *Java network programming*. " O'Reilly Media, Inc.", 2004.
5. Nyatega, Charles Okanda. "Simple File Transfer System using GUI and Socket Programming for Window Operating System." *International Journal of Engineering Research and Technology*. Vol. 3. No. 10 (October-2014). ESRSA Publications, 2014.
6. Malgaonkar, Saurabh, Swarnalata Bollavarapu, and Tejas Hirave. "Remote Nodes Management Using RMI."
7. "java socket programming tutorial". [Online]. Available: http://www.coderpanda.com/java-socket-programming-tutorial/. [Accessed:16- June-2016]
8. "XML Databases". [Online]. Available: http://www.tutorialspoint.com/xml/xml_databases.htm [Accessed: 16- June - 2016]
9. Kappel, Gerti, Elisabeth Kapsammer, and Werner Retschitzegger. "XML and Relational Database Systems-A Comparison of Concepts." *International Conference on Internet Computing (1)*. 2001.
10. "TeamViewer Review". [Online]. Available: http://www.remoteaccess.org/teamviewer-review/ [Accessed:16- June- 2016]
11. "LanVisor". [Online]. Available: http://wiki.appvisor.org/LANVisor [Accessed: 16- June - 2016]
12. "Remote Desktop Monitoring System". [Online]. Available: http://www.lanvisor.com/ [Accessed: 16- June - 2016]
13. "Remote Desktop Protocol". [Online]. Available: https://en.wikipedia.org/wiki/Remote_Desktop_Protocol [Accessed: 16- June - 2016]
14. Rockoff, Todd E., and Michael Groves. "Design of an Internet-based system for remote Dutch auctions." *Internet Research* 5.4 (1995): 10-16 page

APPENDICES

Appendix A: Remote Desktop Module Sample Java Code

// code for Screen capture and Send.

```
package remoteclient;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.Socket;
import javax.swing.ImageIcon;

//This class is responisble for sending sreenshot every predefined duration
class ScreenSpyer extends Thread {
    Socket socket = null;
    Robot robot = null; // Used to capture screen
    Rectangle rectangle = null; //Used to represent screen dimensions
    boolean continueLoop = true; //Used to exit the program

    public ScreenSpyer(Socket socket, Robot robot,Rectangle rect) {
        this.socket = socket;
        this.robot = robot;
        rectangle = rect;
        start();
    }
    public void run(){
        ObjectOutputStream oos = null; //Used to write an object to the streem
        try{
            //Prepare ObjectOutputStream
            oos = new ObjectOutputStream(socket.getOutputStream());
            /*
             * Send screen size to the server in order to calculate correct mouse
             * location on the server's panel
             */
            oos.writeObject(rectangle);
        }catch(IOException ex){
            ex.printStackTrace();
        }
       while(continueLoop){
            //Capture screen
            BufferedImage image = robot.createScreenCapture(rectangle);
            /* I have to wrap BufferedImage with ImageIcon because BufferedImage class
             * does not implement Serializable interface
             */
            ImageIcon imageIcon = new ImageIcon(image);
            //Send captured screen to the server
            try {
              // System.out.println("before sending image");
               oos.writeObject(imageIcon);
               oos.reset(); //Clear ObjectOutputStream cache
               //System.out.println("New screenshot sent");
            } catch (IOException ex) {
              ex.printStackTrace();
            }
            //wait for 100ms to reduce network traffic
            try{
               Thread.sleep(100);
            }catch(InterruptedException e){
               e.printStackTrace();
          }  }  }
```

Appendix B: Remote Control Module Sample Java Code

```
package control;
import java.io.*;
import java.net.*;
import javax.swing.JOptionPane;
public class controlClient {
    Socket s;
    BufferedReader br1;
    public void initialize(String ip, int port) {
        {


            try {
                s = new Socket(ip, port);
                BufferedReader br = new BufferedReader(new
InputStreamReader(s.getInputStream()));
                PrintWriter out = new PrintWriter(s.getOutputStream(), true);
                out.println("i am ready");
                String s = br.readLine();
                String shutdownCommand = "";
                String restartCommand = "";
                String logoffCommand = "";
                String osName = System.getProperty("os.name");
                if (s.equals("shutdown")) {
                    if (osName.startsWith("Win")) {
                        int q = JOptionPane.showConfirmDialog(null, "Do you really want to shutdown
", "Shutdown", JOptionPane.YES_NO_OPTION);
                        if (q == 0) {
                            shutdownCommand = "shutdown.exe -s -t 60";
                        } else if (osName.startsWith("Linux") || osName.startsWith("Mac")) {
                            shutdownCommand = "shutdown -h now";
                        } else {
```

```
                System.err.println("Shutdown unsupported operating system ...");
                }
              Runtime.getRuntime().exec(shutdownCommand);
            }
          } else if (s.equals("restart")) {
            if (osName.startsWith("Win")) {
              restartCommand = "shutdown.exe -r";
            } else if (osName.startsWith("Linux") || osName.startsWith("Mac")) {
              restartCommand = "shutdown -r now";
            } else {
              System.err.println("Restart unsupported operating system ...");
            }
            Runtime.getRuntime().exec(restartCommand);
          } else if (s.equals("logoff")) {
            if (osName.startsWith("Win")) {
              logoffCommand = "shutdown.exe -l";
            } else if (osName.startsWith("Linux") || osName.startsWith("Mac")) {
              logoffCommand = "shutdown -l now";
            } else {
              System.err.println("logoff unsupported operating system ...");
             }
            Runtime.getRuntime().exec(logoffCommand);
          }
        } catch (Exception e) {
          e.printStackTrace();}
    }
  }
}
```

Appendix C: Pen Drive Detection Module Sample Java Code

```
package pendrive;
import java.io.*;
import java.net.*;
public class FindDrive {
    public static String msg() {
        String[] letters = new String[]{"A", "B", "C", "D", "E", "F", "G", "H", "I"};
        File[] drives = new File[letters.length];
        boolean[] isDrive = new boolean[letters.length];
        for (int i = 0; i < letters.length; ++i) {
            drives[i] = new File(letters[i] + ":/");
            isDrive[i] = drives[i].canRead();
        }
        System.out.println("FindDrive: waiting for devices...");
        while (true) {
            for (int i = 0; i < letters.length; ++i) {
                boolean pluggedIn = drives[i].canRead();
                try {
                    InetAddress address = InetAddress.getLocalHost();
                    if (pluggedIn != isDrive[i]) {
                        if (pluggedIn) {
                            return ("PenDrive " + letters[i] + " has been plugged " +
address.getHostAddress());
                        }
                        if (!pluggedIn) {
                            return ("PenDrive " + letters[i] + " has been unplugged " +
address.getHostName());
                        }}
                } catch (UnknownHostException e) {
                    System.out.println("Un able to determine this host's address");
                } } } } }
```

Appendix D: Remote Chat and File Transfer Module Sample Java Code

```
Public void run() {
        boolean keepRunning = true;
        while(keepRunning){
            try {
                Message msg = (Message) In.readObject();
                System.out.println("Incoming : "+msg.toString());

                if(msg.type.equals("message")){
                    if(msg.recipient.equals(ui.username)){
                        ui.jTextArea1.append("["+msg.sender +" > Me] : " + msg.content + "\n");
                    }
                    else{
                        ui.jTextArea1.append("["+ msg.sender +" > "+ msg.recipient +"] : " +
msg.content + "\n");
                    }

                    if(!msg.content.equals(".bye") ){
                        String msgTime = (new Date()).toString();

                        try{
                            hist.addMessage(msg, msgTime);
                            DefaultTableModel table = (DefaultTableModel) ui1.jTable1.getModel();
                            table.addRow(new Object[]{msg.sender, msg.content, "Me", msgTime});
                        }
                        catch(Exception ex){
                        }
                    }
                }
                else if(msg.type.equals("login")){
                    if(msg.content.equals("TRUE")){
```

```
        ui.login.setEnabled(false);
        ui.send.setEnabled(true);
        ui.sendFile.setEnabled(true);
        ui.jTextArea1.append("[SERVER > Me] : Login Successful\n");
       ui.user.setEnabled(false);
        ui.pass.setEnabled(false);
    }
    else{
        ui.jTextArea1.append("[SERVER > Me] : Login Failed\n");
    }
}
else if(msg.type.equals("test")){
    ui.connect.setEnabled(false);
    ui.login.setEnabled(true);
   ui.user.setEnabled(true);
    ui.pass.setEnabled(true);
    ui.host.setEditable(false);
    ui.portnum.setEditable(false);


}
else if(msg.type.equals("newuser")){
    if(!msg.content.equals(ui.username)){
        boolean exists = false;
        for(int i = 0; i < ui.model.getSize(); i++){
            if(ui.model.getElementAt(i).equals(msg.content)){
                exists = true; break;
            }
        }
        if(!exists){ ui.model.addElement(msg.content); }
    }
}
```

```
                        else if(msg.type.equals("signout")){
                  if(msg.content.equals(ui.username)){
                     ui.jTextArea1.append("["+ msg.sender +" > Me] : Bye\n");
                     ui.host.setEnabled(true);

                     ui.host.setEditable(true);
                     ui.portnum.setEditable(true);

                     for(int i = 1; i < ui.model.size(); i++){
                        ui.model.removeElementAt(i);
                     }

                     ui.clientThread.stop();
                  }
                  else{
                     ui.model.removeElement(msg.content);
                     ui.jTextArea1.append("["+ msg.sender +" > All] : "+ msg.content +" has signed
out\n");
                  }
               }
               else if(msg.type.equals("upload_req")){

                  if(JOptionPane.showConfirmDialog(ui, ("Accept '"+msg.content+"' from
"+msg.sender+" ?")) == 0){

                     JFileChooser jf = new JFileChooser();
                     jf.setSelectedFile(new File(msg.content));
                     int returnVal = jf.showSaveDialog(ui);

                     String saveTo = jf.getSelectedFile().getPath();
                     if(saveTo != null && returnVal == JFileChooser.APPROVE_OPTION){
```

```
                    Download dwn = new Download(saveTo, ui);
                    Thread t = new Thread(dwn);
                    t.start();
                    //send(new Message("upload_res",
(""+InetAddress.getLocalHost().getHostAddress()), (""+dwn.port), msg.sender));
                    send(new Message("upload_res", ui.username, (""+dwn.port), msg.sender));
                }
                else{
                    send(new Message("upload_res", ui.username, "NO", msg.sender));
                }
            }
            else{
                send(new Message("upload_res", ui.username, "NO", msg.sender));
            }
        }
        else if(msg.type.equals("upload_res")){
            if(!msg.content.equals("NO")){
                int port  = Integer.parseInt(msg.content);
                String addr = msg.sender;
              Upload upl = new Upload(addr, port, ui.filechat, ui);
                Thread t = new Thread(upl);
                t.start();
            }
            else{
                ui.jTextArea1.append("[SERVER > Me] : "+msg.sender+" rejected file
request\n");
            }
        }
        else{
            ui.jTextArea1.append("[SERVER > Me] : Unknown message type\n");
        }
```

```
          }
          catch(Exception ex) {
              keepRunning = false;
              ui.jTextArea1.append("[Application > Me] : Connection Failure\n");
              ui.connect.setEnabled(true);
              ui.host.setEditable(true);
              ui.portnum.setEditable(true);
             for(int i = 1; i < ui.model.size(); i++){
             ui.model.removeElementAt(i);
              }
            ui.clientThread.stop();
            System.out.println("Exception SocketClient run()");
            ex.printStackTrace();
          }
      }
  }
```