Advanced College of Engineering and Management

Kupondole, Lalitpur

(AFFILIATED TO TRIBHUBHAN UNIVERSITY)

A REPORT

On

Remote Desktop Viewing & Controlling Application

SUBMITTED BY:

Anil K.C (2008/BCT/507) Email:neal2blucky@gmail.com

Sachin Kushwaha(2008/BCT/535)Email:sachinkeshav@gmail.com

Som NathBhandari(2008/BCT/538) Email:Sombhandari88@gmail.com

Tekendra Bista(2007/BCT/240) Email:texabd@gmail.com

November, 2011

Advanced College of Engineering and Management

Kupondole, Lalitpur

(AFFILIATED TO TRIBHUBHAN UNIVERSITY)

Remote Desktop Viewing & Controlling Application

SUBMITTED TO:

Department of Computer & Electronics

SUBMITTED BY:

Anil K.C (2008/BCT/507)
Sachin Kushwaha (2008/BCT/535)
Som Nath Bhandari (2008/BCT/538)
Tekendra Bista (2007/BCT/240)

Lalitpur, Nepal
November, 2011

Abstract

This project “Remote Desktop Viewing & Controlling Application”, a derivative of Remote Administration Application, is a network based project. The main objective of this project is to develop an application that can control remote computer through network. There are three preliminary components i.e. Server, Network and Client. Server acts as a Master while client is similar to Slave. But, the server can control the client only if it is permitted by the client for remote access. The basic idea behind the project is to capture the screen from the client and send to server. In response, mouse events and key events are captured from the Server and exchanged those events between Server and Client via network where same desktop can be tuned as Server or Client according to user’s will. This project also includes the facility of the interaction between server and client exchanging text messages termed as Text Chat.

Acknowledgement

We would like to express our gratitude to all those who gave us the support to complete this project. We are deeply indebted to our Project Coordinator **Er. Badri Adhikari** of Advanced College of Engineering and Management (ACEM) whose help, stimulating suggestions and encouragement helped us to develop this project.

We are equally grateful to **Er. Pranita Upadhya**, the HOD of Computer and Electronics Department, and all the College Board members for their valuable help. We are also grateful to our teacher **Er. Shyan Kirat Rai** for providing us with his precious time for our queries

We would also like to express our gratitude to the Department of Computer and Electronics of ACEM for providing us an environment to work with full potential and energy.

List of Abbreviations

1.	API	Application Program Interface
2.	AWT	Abstract Window Toolkit
3.	FOSS	Free and Open Source Software
4.	GUI	Graphical User Interface
5.	HOD	Head Of Department
6.	IP	Internet Protocol
7.	JDK	Java Development Kit
8.	LAN	Local Area Network
9.	NIO	Network Input / Output
10.	OS	Operating System
11.	RD	Remote Desktop
12.	RMI	Remote Method Invocation
13.	TCP/IP	Transmission Control Protocol/ Internet Protocol
14.	UDP	User Datagram Protocol
15.	VNC	Virtual Network Connection

Table of Contents

List of Figures

1 Introduction

Remote Desktop Administration refers to a method of controlling any computer from any remote location. Remote location may refer to a computer in the next room or one on the other side of the world. It may be legal or illegal remote administration.

This project, Remote Desktop Viewing and Controlling Application aims at administering any computer that is connected in network from remote location. The person sitting on the server can view and control the client's computer. Here controlling refers to taking control of mouse and keyboard.

The server module needs to be run on the computer which acts as server. The server then starts and binds the TCP port and keeps listening to any incoming connection for the client to connect. The client module is then run on the computer which acts as client. The client is then started and captures the screen-shot and sends to the server over TCP through network. The server then gets the client screen-shot, captures the keyboard and mouse event and sends them back to the client over TCP through network for the same event to take place on the client side. All the handling of mouse and keyboard event on the client side is done by "*Robot Class*". The server and client process to view and control the remote desktop is termed as "Remote Access".

This project is able to exchange text between the Server and Client i.e. the users can interact also. This can be helpful part of the software to achieve objective of this project efficiently. The server and client interaction is termed as "Text Chat". This part of program have similar module as Remote Access. The server module starts and binds the TCP port and keeps listening to the client. Similarly, the client starts and attempts to connect to the specified address provided by the user. Here, we make two different connections for Text Chat and Remote Access as the data transfer rate is limited by the Kryonet API we have used in this project.

1.1 Objectives

Remote Desktop Viewing & Controlling Application includes the following objectives:

- View Remote Desktop.
- Mouse Movement Control.
- Keyboard Control.
- Mouse Button control.
- Shutting down and rebooting.

1.2 Project Overview

This project, Remote Desktop Viewing & Controlling Application is built in Java platform. We have used Kryo and Kryonet APIs the connection of server and client. Kryonet API uses TCP and UDP port for the connection between server and client. However, we chose TCP port as a default port to establish the connection between server and client of this application.
Basically, there are three minimum components to make a Remote Desktop to work. They are:

1.2.1Remote Server

The part which awaits the client to connect in the port opened for connection is termed as Remote Server. This part receives the screenshot of the client and loads the image thus received in a frame. The frame then starts listening mouse and key events. The key and mouse events thus received are sent to the client computer through network.

1.2.2Remote Client

The core function of this part is to send the screenshots of the computer to the server part in certain interval and to implement the mouse and key events sent from the server part.

1.2.3Network

Network can be defined as the way of communication in this context. It is the medium through which the server and client can interact with each other. However, different kinds of network can be setup for remote desktop

i.e. LAN or internet. In this particular project we are going to establish the connection between the server and client through LAN.
The connection between the server and client must be authorized. For this purpose we have implemented simple way in which the client only, is able to connect to the server, feeding the IP address of the server in the client module. However, the server should be ready to receive the connection from the client beforehand.

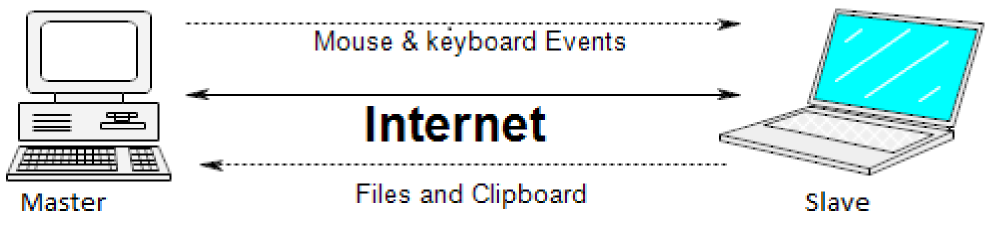


Figure 1: Interaction between Java Virtual Machine (JVM)

1.3 Screenshots

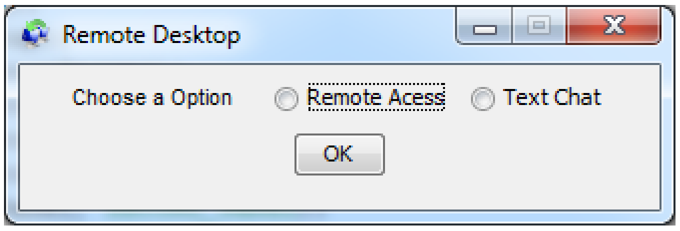
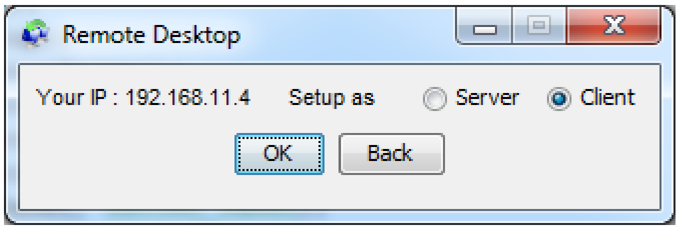


Figure 2: Remote Desktop startup GUI


Remote Desktop

Your IP : 192.168.11.4 Setup as Server [✓ Client]

Figure 3: Client-Server tuning option UI

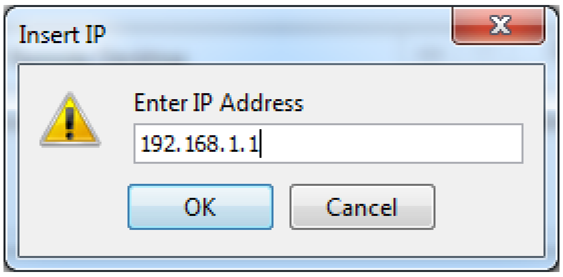
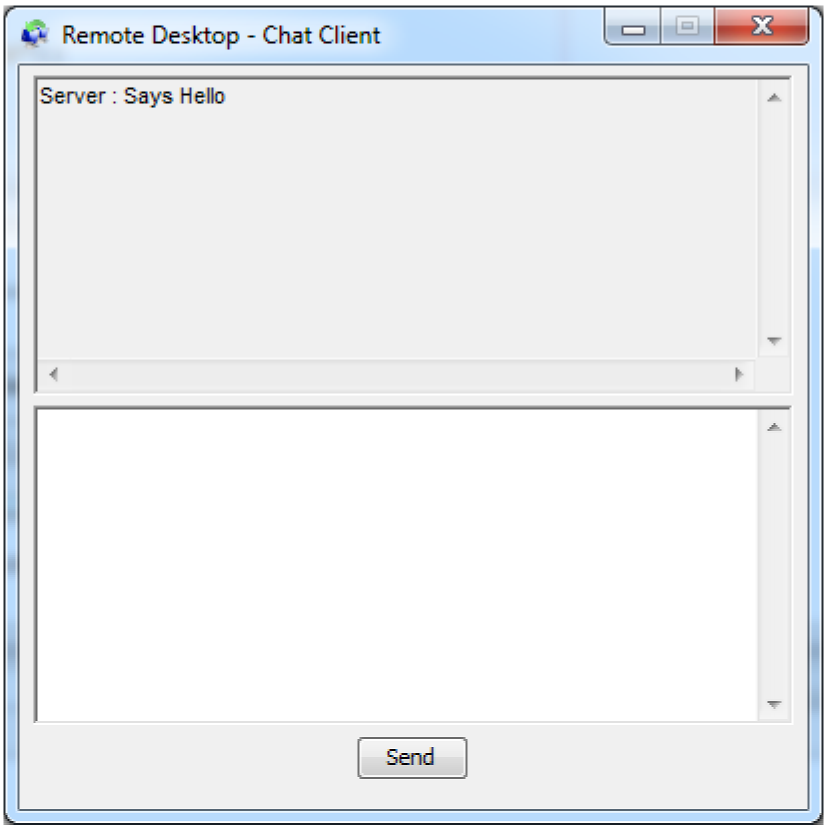
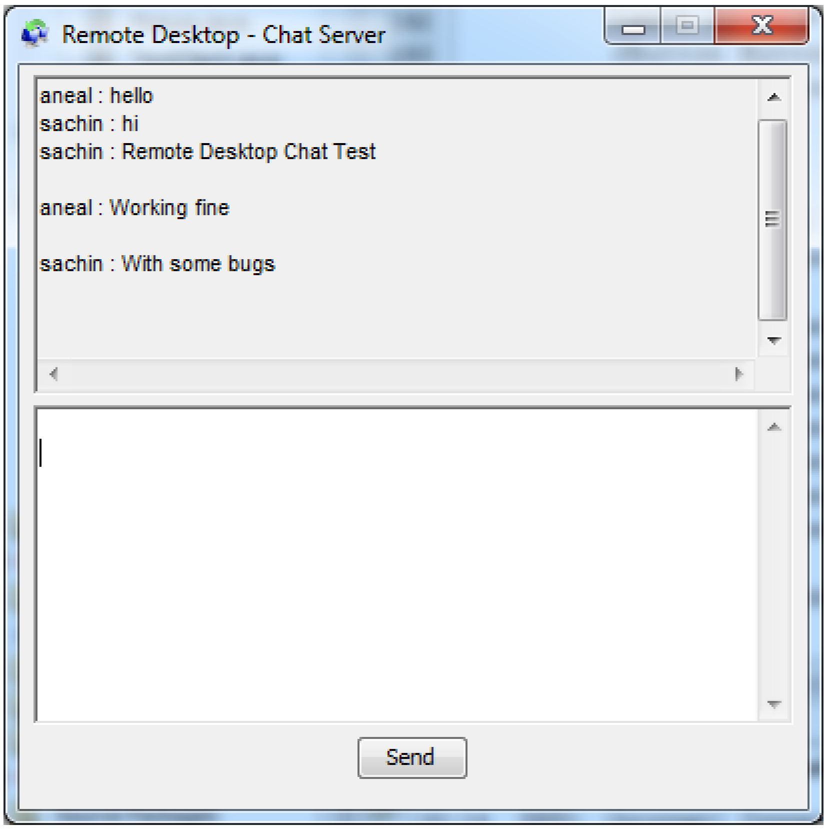


Figure 4: IP input dialogue box in Client



Server : Says Hello

Figure 5: Chat window in Client side



aneal : hello
sachin : hi
sachin : Remote Desktop Chat Test

aneal : Working fine

sachin : With some bugs

Figure 6: Chat window in Server side

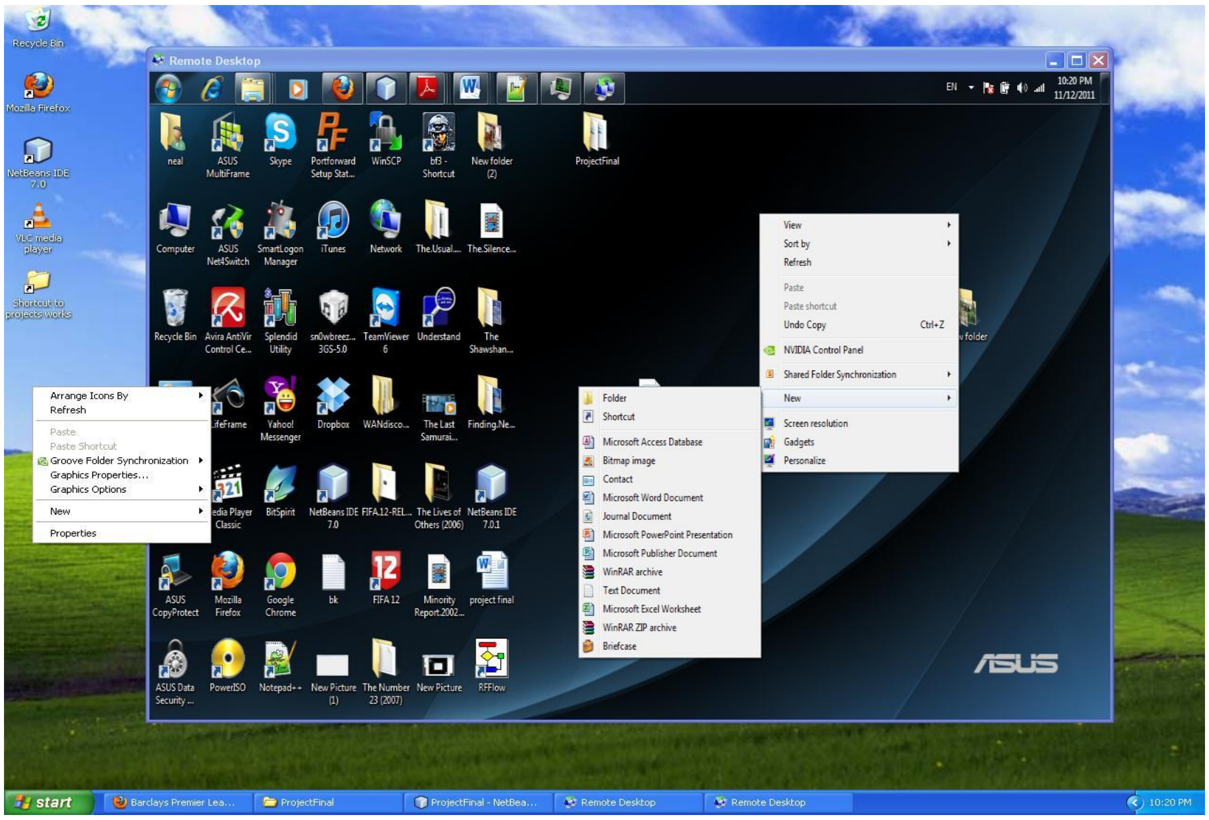


Figure 7: Client screenshot on Server

1.4 Technologies Used

S.No.	Technologies Used	Purpose
1.	JAVA Standard Edition [JDK 6 update 26]	Development Kit
2.	Java	Programming Language
3.	Kryo and kryonet	APIs used
4.	NetBeans 7.0/ Eclipse	Interface and Programming
5.	MS Word/ Ms Excel/ MS PowerPoint	Documentation
6.	Sourceforge.net	Configuration and Management
7.	Microsoft Windows 7/XP/Ubuntu 11.04	OS for Development and Testing

1.5 Features

Following are the features of our project:

- User friendly GUI.
- Text chat support.
- Quick keyboard and mouse response.
- Supported by Windows and Linux operating system.

2 Literature Review

One of the trends we have been observing for some time now is the blurring of divisional lines between different types of malware. Classifying a newly discovered 'creature' as a virus, a worm, a Trojan or a security exploit becomes more difficult and anti-virus researchers spend a significant amount of their time discussing the proper classification of new viruses and Trojans. Depending on the point of view, very often, the same program may be perceived as a Remote Desktop Tool allowing a potentially malicious user to remotely control the system. A Remote Desktop Tool is remote control software that when installed on a computer it allows a remote computer to take control of it. With remote control software you can work on a remote computer exactly as if you were right there at its keyboard.

Basically, Remote Desktop is based on client and server module where server acts as master and client acts as slave. There are many ways to establish the connection between client and server in Java platform. Among them one of the easy and efficient way is provided by the API called Kryonet.

2.1 Kryonet

It is an open source API. It is efficient for TCP and UDP client/server network communication using NIO. It uses Kryo serialization library for the serialization of objects. This API runs on both Desktop and Android. It is Ideal for any Client/Server application. It is very efficient, so is especially good for games. Kryonet can also be useful for inter-process communication.

2.1.1Running a server

This code starts a server on TCP port 54555 and UDP port 54777:

```
Server server = new Server();
server.start();
server.bind(54555, 54777);
```

The start method starts a thread to handle incoming connections, reading/writing to the socket, and notifying listeners. The following code adds a listener to handle receiving objects:

```
server.addListener(new Listener() {
   public void received (Connection connection, Object object) {
      if (object instanceof SomeRequest) {
         SomeRequest request = (SomeRequest)object;
         System.out.println(request.text);

         SomeResponse response = new SomeResponse();
         response.text = "Thanks!";
         connection.sendTCP(response);
      }
   }
});
```

The Listener class has other notification methods that can be overridden. Typically a listener has a series of instanceof checks to decide what to do with the object received. In this example, it prints out a string and sends a response over TCP. Kryo automatically serializes the objects to and from bytes.

2.1.2 Connecting a client

The following code connects to a server running on TCP port 54555 and UDP port 54777:

```
Client client = new Client();
client.start();
client.connect(5000, "192.168.0.4", 54555, 54777);

SomeRequest request = new SomeRequest();
request.text = "Here is the request!";
client.sendTCP(request);
```

The start method starts a thread to handle the outgoing connection, reading/writing to the socket, and notifying listeners. The thread must be started before connect is called, else the outgoing connection will fail.

In the above example, the connect method blocks for a maximum of 5000 milliseconds. If it times out or connecting otherwise fails, an exception is thrown (handling not shown). After the connection is made, the example sends a "SomeRequest" object to the server over TCP.

The following code adds a listener to print out the response:

```
client.addListener(new Listener() {
   public void received (Connection connection, Object object) {
      if (object instanceof SomeResponse) {
         SomeResponse response = (SomeResponse)object;
         System.out.println(response.text);
      }
   }
});
```

2.1.3Registering classes

In order for the above examples to work, the classes that are going to be sent over the network must be registered with the following code:

```
Kryo kryo = server.getKryo();
kryo.register(SomeRequest.class);
kryo.register(SomeResponse.class);

Kryo kryo = client.getKryo();
kryo.register(SomeRequest.class);
kryo.register(SomeResponse.class);
```

This must be done on both the client and server, before any network communication occurs. It is very important that the exact same classes are registered on both the client and server, and that they are registered in the exact same order.

2.1.4Remote Method Invocation

Kryonet has an easy to use mechanism for invoking methods on remote objects (RMI). This is done by creating an ObjectSpace and registering objects with an ID:

```
ObjectSpace objectSpace = new ObjectSpace();
objectSpace.register(42, someObject);
// ...
objectSpace.addConnection(connection);
```

Multiple ObjectSpaces can be created for both the client or server side. Once registered, objects can be used on the other side of the registered connections:

```
SomeObject someObject = ObjectSpace.getRemoteObject(connection, 42,
SomeObject.class);
SomeResult result = someObject.doSomething();
```

The `getRemoteObject` method returns a proxy object that represents the specified class. When a method on the class is called, a message is sent over the connection and on the remote side the method is invoked on the registered object.

2.1.5TCP and UDP

Kryonet always uses a TCP port. This allows the framework to easily perform reliable communication and have a stateful connection. Kryonet can optionally use a UDP port in addition to the TCP port. Both ports can be used simultaneously, it’s not recommended to send huge amount of data on both at the same time because the two protocols can affect each other.

TCP is reliable, meaning objects sent are sure to arrive at their destination eventually. UDP is faster but unreliable, meaning an object sent may never be delivered. Because it is faster, UDP is typically used when many updates are being sent and it doesn't matter if an update is missed. Kryonet does not currently implement any extra features for UDP, such as reliability or flow control. It is left to the developer to make proper use of the UDP connection.

2.1.6Threading

Kryonet imposes no restrictions on how threading is handled. The Server and Client classes have an update method that accepts connections and reads or writes any pending data for the current connections. The update method should be called periodically to process network events.

2.1.7LAN server discovery

Kryonet can broadcast a UDP message on the LAN to discover any servers running:

```
InetAddress address = client.discoverHost(54777, 5000);
System.out.println(address);
```

This will print the address of the first server found running on UDP port 54777. The call will block for up to 5000 milliseconds, waiting for a response.

2.1.8Logging

Kryonet makes use of the low overhead, lightweight MinLog logging library. The logging level can be set in this way:

```
Log.set(LEVEL_TRACE);
```

2.2 Kryo

It is an open source and easy to use API. It is fast, efficient for Java serialization. This API runs on both Desktop and Android and is Object graph serialization framework for Java. It is anytime objects need to be persisted, whether to a file, database, or over the network.

2.3 Java Robot Class

In this project, we need to move the mouse pointer, simulate key stroke and capture the screen, which is done by Robot class. It uses serialization to send screenshots from client to server and also send server events like mouse move, key press, key release instead of sending images and events in raw data format over the network.
By using the Java Robot class we can generate the actual input events that are different from posting events to the Abstract Window Toolkit (AWT) event queue. Some of the methods Java Robot classes are:

- `mouseMove` - Moves the mouse pointer to a set of specified absolute screen coordinates given in pixels
- `mousePress` - Presses one of the buttons on the mouse
- `mouseRelease` - Releases one of the buttons on the mouse
- `keyPress` - Presses a specified key on the keyboard
- `keyRelease` - Releases specified key on the keyboard
- `createScreenCapture` - Takes a screenshot

3 Feasibility Study

The main objective of this study is to determine (Remote Desktop) RD is feasible or not. Mainly there are three types of feasibility study to which the developed system is subjected as described below. The key considerations are involved in the feasibility:

1. Technical feasibility
2. Economic feasibility
3. Operational feasibility

The developed system must be evaluated from a technical viewpoint first, and their impact on the organization must be accessed. If compatible, behavioral system can be devised. Then they must be tested for the feasibility.

3.1 Technical Feasibility

Remote Desktop Application satisfies technical feasibility because this Service can be implemented as a stand-alone application.

- Microsoft Windows Operating System compatible.
- Linux Operating System Compatible.

3.2 Economic feasibility

Our project entitled Remote Desktop (RD) is economically feasible because it is developed using very less amount of economic resources. It is Free and Open Source Software (FOSS).

3.3 Operational feasibility

Operational feasibility should be accounted after the software is developed so that it can cope up with the defined objectives.

- The application is user friendly with its GUI and handy to use.
- The application is be affordable because the requirement is just normal computers with working LAN port.
- Since this application is developed in Java it runs both on Windows and Linux system.

4 System Architecture & Design

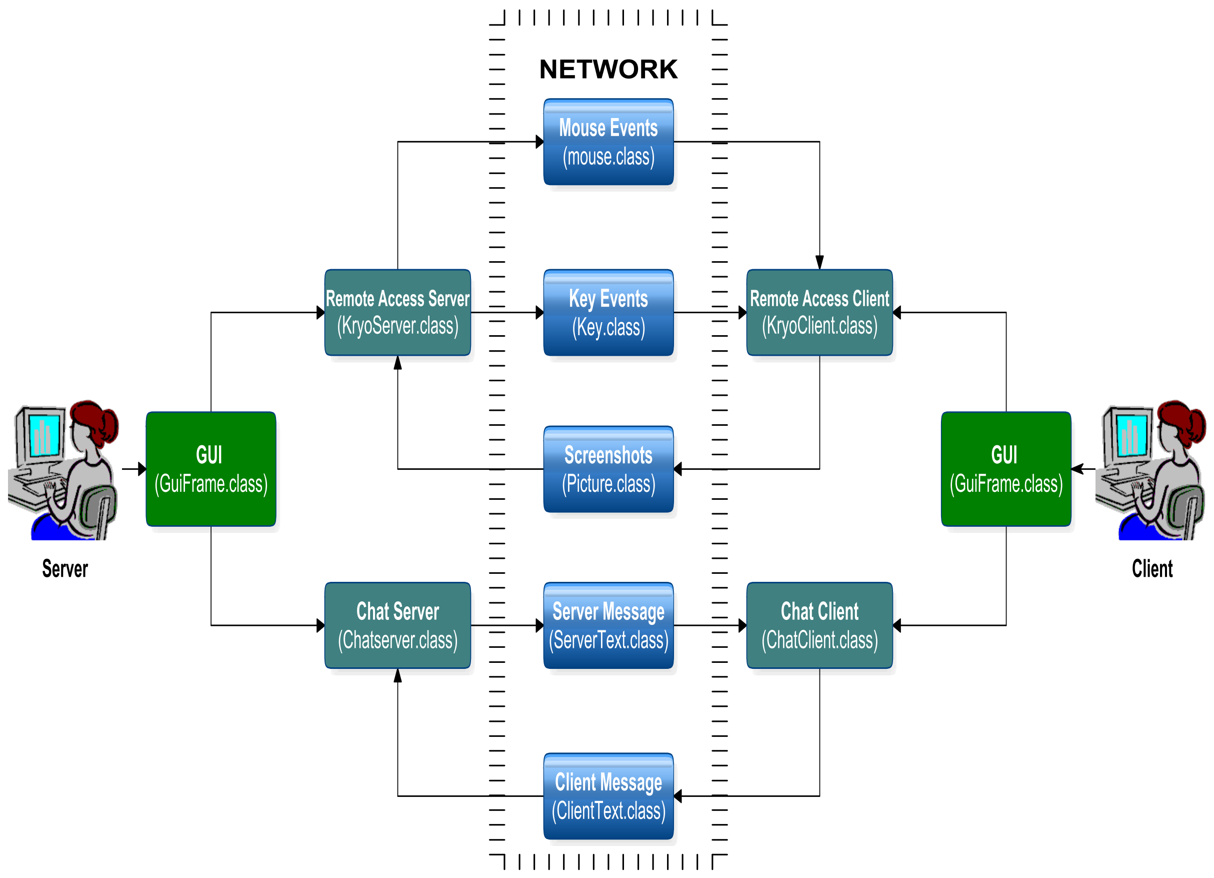

NETWORK
Mouse Events (mouse.class)
Remote Access Server (KryoServer.class)
Key Events (Key.class)
Remote Access Client (KryoClient.class)
GUI (GuiFrame.class)
Screenshots (Picture.class)
GUI (GuiFrame.class)
Server
Client
Chat Server (Chatserver.class)
Server Message (ServerText.class)
Chat Client (ChatClient.class)
Client Message (ClientText.class)

Figure 8: Block diagram of Remote Desktop

The above figure shows the architecture design of our project. The server module contains three prime classes namely: GuiFrame, KryoServer and ChatServer. The client module also contains two prime classes namely: KryoClient and ChatClient. Apart from two prime classes Mouse, Key, Picture, ServerText and ClientText classes are common in both server and client module.

Following are the classes used in this application:

- **GuiFrame.class**: It creates the basic GUI on both server & client side and provides an option to tune the computer either as server or client. It also the necessary inputs for the connection.

- **KryoServer.class**: It starts the server, binds the port and waits for the connection. It loads the screenshot received from the client module. It also captures the keyboard and mouse events and sends over the network to the client module.
- **ChatServer.class**: It starts the server, binds the port and waits for the connection. It generates the basic frame which displays the message from the client and sends the text message to the client.
- **KryoClient.class**: It starts the client and connects the client to the specified computer with the provided IP address. It captures and sends the screenshot and implements the mouse & keyboard events received from the server.
- **ChatClient.class**: It starts the client and connects the client to the specified computer with provided IP address. It generates the basic frame which displays the message from the server and sends the text message to the server.
- **Mouse.class**: It contains the attributes of mouse event. This class is registered on both server and client side in order to send the instance of this class over the network.
- **Key.class**: It contains the attributes of keyboard event. This class is registered on both server and client side in order to send the instance of this class over the network.
- **Picture.class**: It contains attributes of screenshot captured from the client in the form of byte array. This class is registered on both server and client side in order to send the instance of this class over the network.
- **ServerText.class**: It contains the attributes of text message sent by server. This class is registered on both server and client side in order to send the instance of this class over the network.
- **ClientText.class:** It contains the attributes of text message sent by client. This class is registered on both server and client side in order to send the instance of this class over the network.

4.1 System Testing

Since this project involves client and server module, we tested this application on three different computers. Two of them were running Windows 7 OS and one had Windows XP.

4.1.1 Challenges Faced

There was no trouble to establish the connection between the client and server. Earlier while sending the screenshot from the client side some errors were generated. The error log given as:

```
00:01 DEBUG: [kryonet] Unable to send TCP with connection: Connection 1
com.esotericsoftware.kryo.SerializationException:   Unable   to   serialize
object of type: Server.Picture
```

The log was extracted from output console of Netbeans IDE. Later we resolved the error by increasing the client and server buffer size.
The other major problem we faced was to load the image received from client on the server side. Though we managed to load the image somehow we had problem to listen mouse and keyboard events simultaneously on the Panel that loaded the image. Later, we traced the error in the code and found the error was caused by multiple mouse and key listener added to the panel. The error then was rectified by running the code just once though the image is loaded to the panel continuously.
Similarly, we came to face another problem regarding the size of JFrame and image size. The imaged loaded in the frame was a little trimmed on all sides. This was caused by the `setSize()` method of the JFrame class. This method sets the specified size of JFrame including the title bar and borders on three sides. Later, we eliminated this unusual error by setting the size of contentPane rather the setting the size of JFrame.

4.1.2 Limitations

Though we managed to resolve most of the problems we faced, still there are some challenges we could not get through. Those challenges led to create some of the limitations of the application. Some of the limitations of the applications are listed below:

- The keyboard event is limited to single key operation. Combo keys such as Alt + Tab, Win + R, Ctrl + Alt + Del etc. are not implemented.
- The cursor type of the server side does not change with the change in cursor type of client side (e.g. On client side the cursor changes to hand form while pointing to hyperlinks while in server side the cursor remains as it is i.e. pointed form).
- The screenshot image sent from the client takes time to load in server side.

- This application doesn’t work in the Internet. It is limited to LAN.
- This application can only make TCP connections.
- There are no security measures implemented against possible hacking and loss of data.
- There is no facility for multiple clients to connect on single server simultaneously.
- Client cannot block specific portion of desktop or program for remote access.

4.1.3Future Enhancements

We have plans to modify this application for more practical and commercial use. Hence, we have listed some of the points as future enhancements as below:

- File Transfer.
- Modifying System services.
- Voice and Video Chat.
- Desktop control through Android.
- Installing the software in server computer on another machine.
- Using encryption methods for the secure connection and transfers.
- Password security.
- Using the application via the Internet.

5 Conclusion

Hence, the networking software, Remote Desktop Viewing & Controlling Application is developed to meet the minimum objective of this project. We gained the general concept of Java programming language. By the end of the project we are able to present an application that will help the user to view and control the remote computer. This application also helps to interact with the user of remote computer.
This application is Free and Open Source Software (FOSS). The application and the related documents can be downloaded from sourceforge.

6 Bibliography

1. O'Reilly Head First Java 2nd Edition (2010), Kathy Sierra & Bert Bates
2. Wikipedia, (2001-2011)
    - http://en.wikipedia.org/wiki/Remote_Administration_Tool
    - http://en.wikipedia.org/wiki/Java_%28programming_language%29
    - http://en.wikipedia.org/wiki/TeamViewer
3. Kryonet, (2011)
    - http://code.google.com/p/kryonet/
    - http://kryonet.googlecode.com/svn/api/index.html
    - http://code.google.com/p/kryo/
4. TeamViewer,(2011), http://www.teamviewer.com/en/index.aspx
5. Stackoverflow , (2011), http://stackoverflow.com/questions/8027849/clear-components-of-jframe-and-add-new-componets-on-the-same-jframe
6. Tycoontalk, (2011), http://tycoontalk.freelancer.com/coding-forum/63227-image-resizing-in-java.html
7. Codeproject,(2011), http://www.codeproject.com/KB/IP/RemoteAdminJava.aspx
8. Slideshare,(2011), http://www.slideshare.net/thadeshvar/project-report-format
9. McGraw Hill Java The Complete Reference J2SE 5th Edition (2004), Herbert Schildt