TRIBHUVAN UNIVERSITY

Institute of Engineering

Pulchowk Campus

Department of Electronics and Computer Engineering

MINOR PROJECT REPORT ON "SCEARU" EG677CT

Submitted By:

Basanta Raj Onta	27106	basanta_onta@hotmail.com	9841472370
Binit Amatya	27109	binitamatya@hotmail.com	9741001666
Samita Lalchan	27137	esa_lale@hotmail.com	9841369496

Submitted To:

Department of Electronics and Computer Engineering
Pulchowk Campus
2nd March, 2008

1. Acknowledgement

We would like to thank our teachers, Mr. Jayaram Timsina and Mr. Bikash Shrestha for their guidance and friends for their support, kind help and cooperation.

We have to give special thanks to our teachers for their fruitful and valuable suggestions. Without their help and guidance this project would have been very hard to complete. They provided us with the wealth of information for making our project possible. And our friends were very helpful in their suggestions that helped a lot in development of our project and gave us a lot of good ideas which were very fruitful in our project development.

Any suggestion and improvements regarding the project are highly appreciated.

Basanta Raj Onta	(27106)
Binit Amatya	(27109)
Samita Lalchan	(27137)

2. Abstract

In the field of networking, there is always the need of the software that enables the user to remotely access other computer, and work on it. This type of working environment increases the productivity of the organization and minimizes the cost. A user can ask for help remotely and get it instantly.

This project is aimed to be such software. The project is titled “SCEARU”, from which the word “Share” originated. The title matched our desire as we want to share the display and input devices in the network. In simple words, SCEARU is a project that allows a computer to be remotely accessed by another computer. SCEARU can also be used to just share the desktop of a remote computer without giving any control to the client.

This project has been designed for the completion of the 3rd year Minor Project for Bachelor of Engineering in Computer from Institute of Engineering, Pulchowk Campus.

3. Table of Contents

4. Project Background

In multi-computer system, the remote user may need to connect to other system and make a few changes to it. Generally speaking, the way is to move to all the computer system individually and controlling it. This procedure is time-consuming and might take really long time if the computers are in different physical areas.

To tackle this problem, simple software that allows user to remotely view a computer and control it, can be implemented with great success. There are many complex hardware and software solutions such as VNC, Synergy and KVM, which tackle the same problem. Our project also tries to tackle this problem in a simple way.

5. Introduction

In our program, a user can log on to a remote computer from his own computer and have access to all application, files, disks, and several other resources of the remote computer as though he were in front of it.

Programs can be left running in the remote computer and the user can see the remote desktop on the screen and also move the mouse and browse the remote computer.

Separate programs have been developed for handling the server and client computers. The server program basically listens for the client connection at a specific port when it is instructed to do so. When it receives a connection from the client, it checks the IP address, port address and password of the client and allows access to the client only if password is correct. Once the client is connected, the client can view the server’s desktop and give input commands to the server, if the server has that option enabled. The connection can be closed by either the client or the server as required. If the client disconnects the server waits for another connection.

6. Objectives

The major objectives of our software were (as specified in the proposal):

The Software Package will have two major parts

i) Controlled System
ii) User System

A system can be turned into a Controlled System or a User System by configuring the software.

i) Controlled System

- Will receive Keyboard and Mouse Events as recorded by the user system

- Will act according to the events it receives

- Will forward the current desktop screen periodically to the User system.

- Will be in locked mode as long as a User System is connected to it, i.e. only one User System can connect to a Controlled System at a time.

ii) User System

- Will forward its Keyboard and Mouse Events to the Controlled system

- Will periodically refresh the remote desktop window to show the current desktop screen to the Controlled System.

The software will encrypt sensitive information like passwords and will compress the data packets to achieve greater efficiency.

7. Development Environment

Visual C#, is one of the languages that can be used to create applications that will run in the .NET CLR. It is an evolution of the C and C++ languages and has been created by Microsoft specifically to work with the .NET platform. As it is a recent development, the C# language has been designed with hindsight, taking into account many of the best features from other languages while clearing up their problems. Developing applications using C# is simpler than using C++, as the language syntax is simpler. However, C# is a powerful language and there is little we might want to do in C++ that we can't do in C#.

For proper programming environment for Visual C#, we used Microsoft Visual Studio 2005 IDE.

8. Implementation

The implementation of the program is as follows:

The server is started with proper options, according to the needs of the user and the user sets the password for the connection. The client connects to the server and sends the password. If the password is authenticated, the server sends its screen resolution to the client and client returns its resolution to the server. The size of the picturebox of the form is set to the smaller screen resolution. If the client resolution is smaller than the server resolution, resolution of the server is set to match the client resolution. If the disable wallpaper option is true, then the server wallpaper is disabled.

The server then sends the screenshot of its desktop to the client which is shown on the picturebox of the client form. Each successive screenshot taken at the server is then XORed with the previous screenshot and compressed. If the screenshot is similar to the previous screenshot, the XORed value will contain mostly zeros and will be very compressed. However if the screenshot is very different from the previous screenshot, the compressed value is sometimes larger than the XORed value, so the size of the two values are checked and the smaller sized byte array is sent to the client with appropriate control word. If the byte array being sent is of compressed bytes, then the control word is 0xFF else, it is 0x80.

The client checks the control word and if the byte array being received is compressed, decompresses it. The decompressed value is then XORed. This XORed value or the XORed value received is XORed with the screenshot being shown in the picturebox to get the new picture, which is then shown.

Each keyboard or mouse event in the client is handled and sent to the server. A byte is sent as the control word along with the event details. The bits of the control word have following meaning:

For Keyboard, the Control word is as follows:

0	Alt	Ctrl	Shift	0	0	0	0

For Mouse, the Control word is as follows:

1	Left	Middle	Right	0	0	0	0

The keycode is sent to the server after the control word for keyboard events and the mouse click location is sent in case of mouse events. At the server, the keyboard event is simulated using keybd_event() and mouse event is simulated using mouse_event() of the USER32.dll API provided by windows.

When the client or the server closes the connection, 0x06 is sent to the other system to signal the closing of connection.

After the connection is closed by the client or the server, the previous system state of the server is restored by re-enabling the wallpaper and resetting the resolution. The server then waits for another client to connect

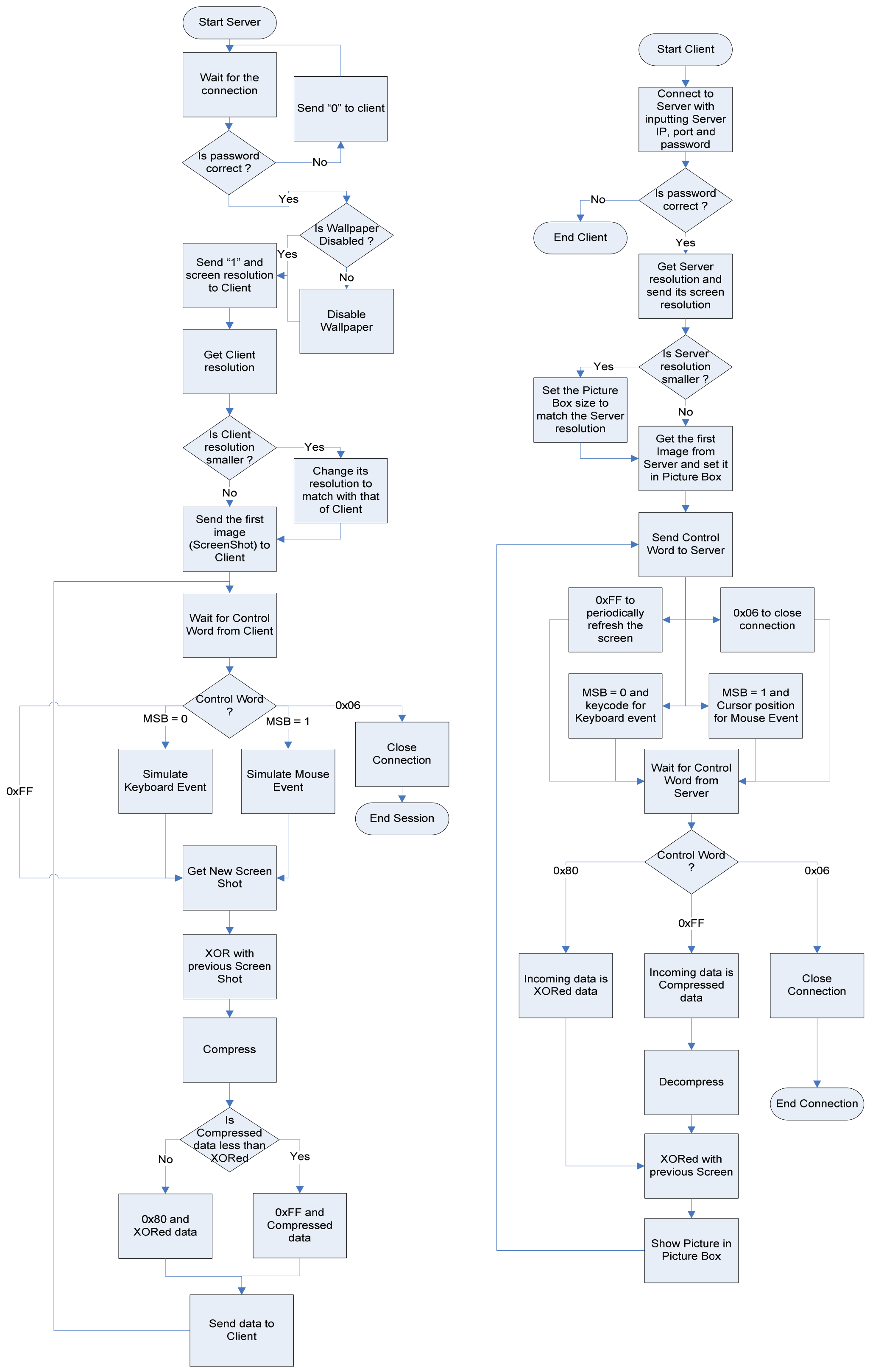
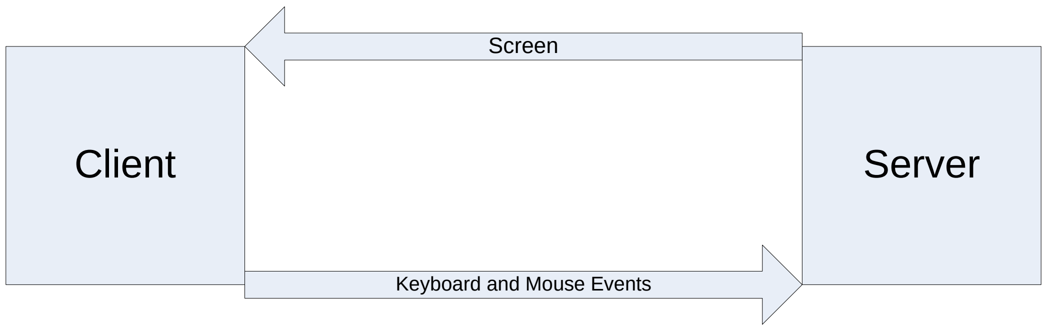

Start Server
Wait for the connection
Send "0" to client
Is password correct ?
No
Yes
Is Wallpaper Disabled ?
Send "1" and screen resolution to Client
Yes
No
Disable Wallpaper
Get Client resolution
Is Client resolution smaller ?
Yes
Change its resolution to match with that of Client
No
Send the first image (ScreenShot) to Client
Wait for Control Word from Client
Control Word ?
MSB = 0
MSB = 1
0x06
Close Connection
End Session
0xFF
Simulate Keyboard Event
Simulate Mouse Event
Get New Screen Shot
XOR with previous Screen Shot
Compress
Is Compressed data less than XORed
No
Yes
0x80 and XORed data
0xFF and Compressed data
Send data to Client
Start Client
Connect to Server with inputting Server IP, port and password
Is password correct ?
No
End Client
Yes
Get Server resolution and send its screen resolution
Is Server resolution smaller ?
Yes
Set the Picture Box size to match the Server resolution
No
Get the first Image from Server and set it in Picture Box
Send Control Word to Server
0xFF to periodically refresh the screen
0x06 to close connection
MSB = 0 and keycode for Keyboard event
MSB = 1 and Cursor position for Mouse Event
Wait for Control Word from Server
Control Word ?
0x80
0x06
0xFF
Incoming data is XORed data
Incoming data is Compressed data
Close Connection
Decompress
End Connection
XORed with previous Screen
Show Picture in Picture Box

Figure 1 : Implementation – System Flow Diagram

9. Program Features

The basic feature of the system is to connect to remote desktop and handle the keyboard and mouse events. The client application connects to the server application giving full access to the server (remote) computer. The server desktop can be seen in client monitor. The keyboard and mouse events from client computer are sent to server computer and are accurately handled.



If the clients' screen resolution is smaller than that of the servers', the servers' computer resolution is altered to match with that of clients'. This feature is helpful in providing the proper view at client computer, and handling input events. After the session is ended, the resolution is reset to previous value.

The user can enable or disable the wallpaper for the session during connection in server. By default, the wallpaper is disabled to minimize the size of the screenshot. After the session is ended, the wallpaper is reset to previous value.

Both events are handled using USER32.DLL API provided by Windows. To disable the wallpaper and resetting to previous value, we used the registry information. The information about the desktop is stored in HKEY_CURRENT_USER\Control Panel\Desktop.

The user can enable or disable the viewer input. The feature provides the user (or authorized person) to lock the server so that keyboard and mouse event generated by the client is not handled in server during the session. If the feature is disabled, the client can only view the screenshot sent by the server, but cannot work on it. By default the option is enabled.

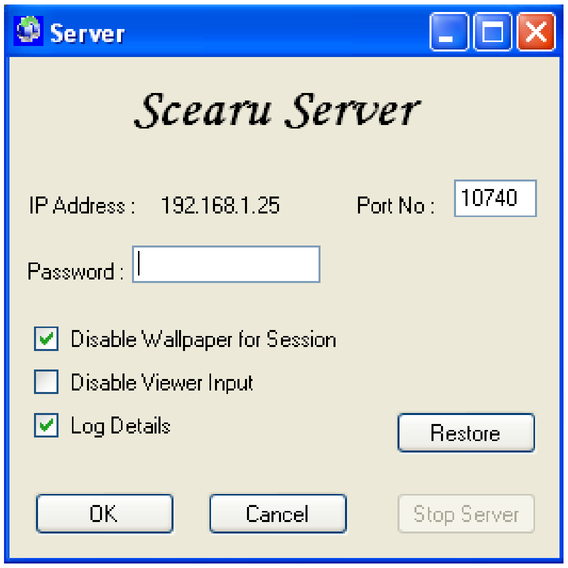
The user can also change the port number. The default value is 10740.The user also can set any password so that the authorized client can only connect. The default password is 123456. The password is encrypted with MD5 hash algorithm, provided in namespace System.Security.Cryptography .The feature provides some degree of security and authentication.

Log information about the each activities performed is kept in server. We used the class EventLog provided in namespace System.Diagnostics. This provides interaction with windows event logs.

We used GDI32.DLL and USER32.DLL API to get the screenshot and Deflate compression algorithm provided in namespace System.IO.Compression for compression.

We have also implemented threading. Each events are performed in different threads. Server (or Client) is run in one thread and the input events are handled in another thread. It makes the operation a bit faster.

Some screenshots of our software are given below:

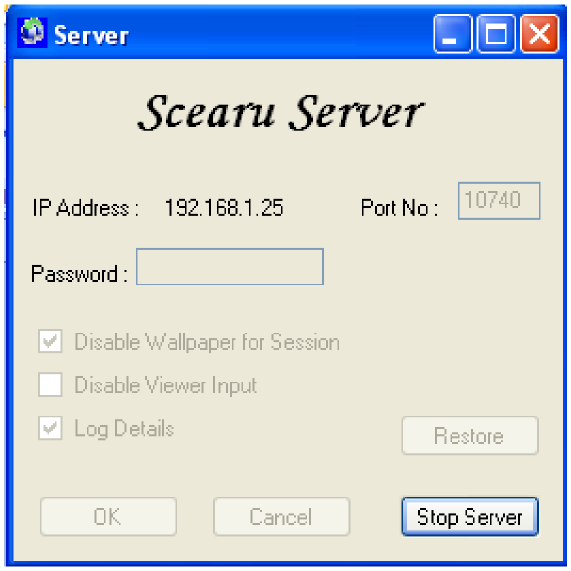


IP Address : 192.168.1.25 Port No : 10740

Password :

☑ Disable Wallpaper for Session
☐ Disable Viewer Input
☑ Log Details

Figure 2 : Before starting server



IP Address : 192.168.1.25 Port No : 10740

Password :

☑ Disable Wallpaper for Session
☐ Disable Viewer Input
☑ Log Details

Figure 3 : After starting server

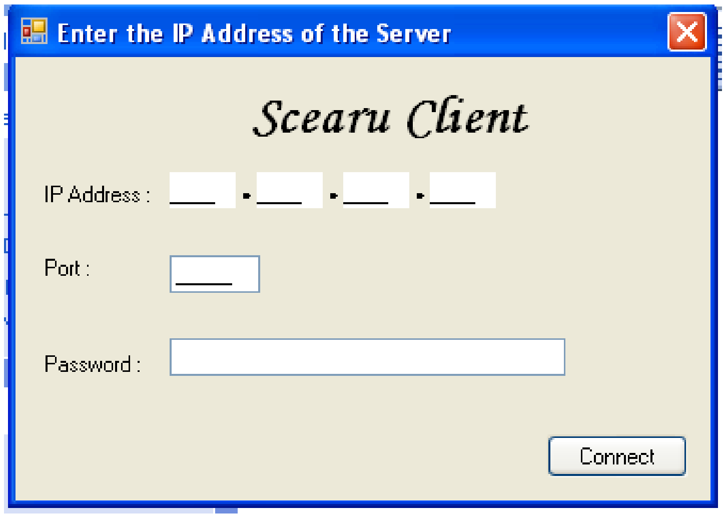


Figure 4 : Client before connection

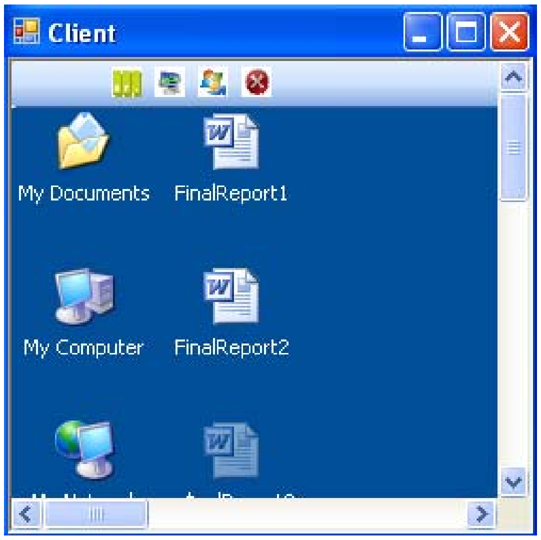


Figure 5 : Client after connection - Normal Screen

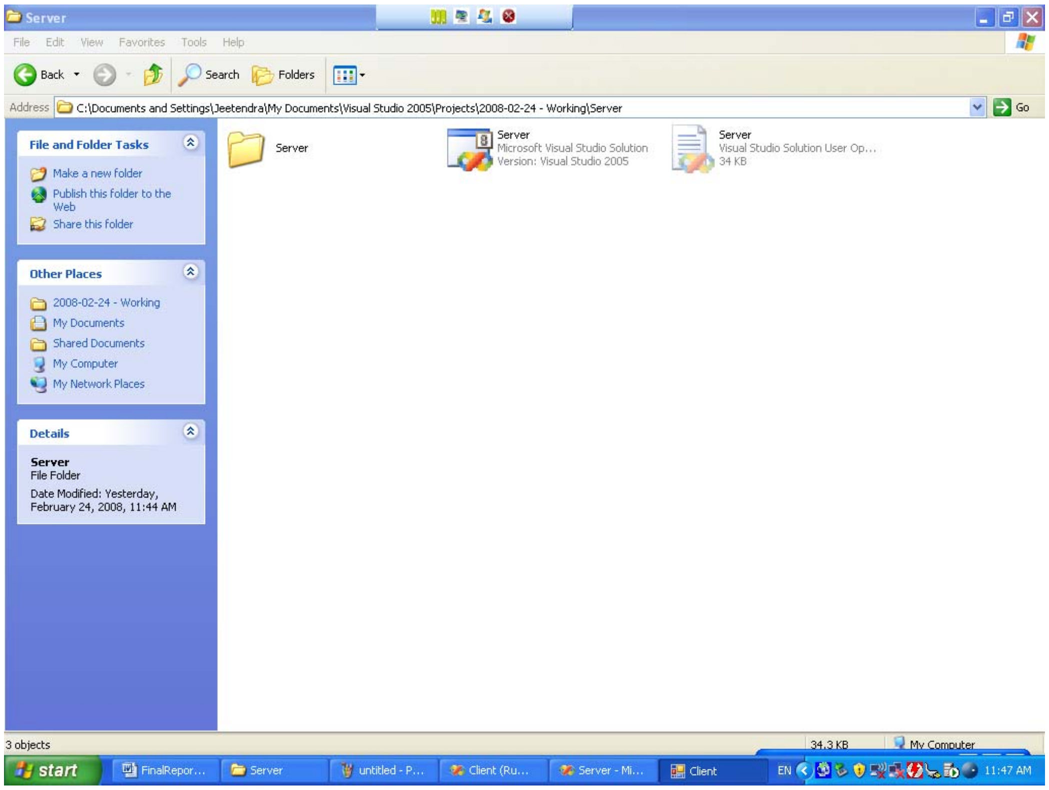


Figure 6 : Server Desktop view from client when window maximized

10. Problems Faced

When we started this project, we had almost no prior experience on C# and Visual Studio.NET, the primary development tools we used in our project. We also did not know anything about Windows APIs. So, we faced a lot of problems. However as we got more experience with working with these development tools, we found it easier to work with and solved most of the problems.

Another problem we faced was the large size of the screenshot. If the screenshot was taken in Bitmap (BMP) format, the size is enormous, the Portable Network Graphics (PNG) format size and Joint Pixel Exchangeable Graphics (JPEG) format size were almost same and the Graphical Interchangeable Format (GIF) format image was the smallest. However, the gif format image was also the least in quality. So we decided to go with the PNG format and use XOR and compression techniques to compress the image where possible.

Apart from that, we also faced problems due to the multithreading nature of the program. To counter race conditions and to synchronize writing of data to the network, we used lock variables.

11. Limitation

Because of time and knowledge, and some uncharacteristic cases, there are some limitations in this version of the system. We have tried our best to handle these limitations.

The main limitation is the slowness. The connection takes a lot of bandwidth. The screenshot from the server is needed to be sent to client and the input events are needed to be sent to server. This process takes a lot of bandwidth and makes the connection very slow. We have tried to resolve the problem using XORing and compression while sending screenshot. But the problem we faced in that is, if the variation in screenshot is maximum the data packet size increases. This is because we used the simplest compression available.

Also, the application takes a lot of CPU time. The CPU Usage is 100%, which make it even slow.

12. Future Enhancements

Some future enhancements (to-do list) are listed below:

- File transfer
- Use of Mirror Driver to handle video, keyboard and mouse.
- Use of Hook to handle keyboard and mouse events. The problem in using this feature is that if improperly handled, the client computer is never unhooked, and it captures all input events not meant for remote computer. If other application is to be used in client computer, the application tries to send the events to server which is not meant for it.
- Multiple Client handling
- Capture and handle mouse wheel events

13. Conclusions

After the completion of this project, we have been able to produce software which helps the sharing of the desktop, mouse and keyboard between server and client. A client can view the server’s desktop and access it without the physical presence of the user in front of the server computer.

14. References

C# How To Program – H. M. Deitel, P. J. Deitel, J. Listfield, T. R. Nieto, C. Yaeger, M. Zlatkina

Visual C# 2005: How to Program, Second Edition - H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc.

C# The Complete Reference

Network Programming in C#

http://www.codeproject.com

MDSN forum

Other related forums on the internet