Development of Remote Access Network Communication (RANC)

Randhir Kumar
Dept. of Information Technology
Gyan Ganga Institute of Technology & Management
Bhopal, M.P, India

Akash Anil
Dept. of Information Technology
Gyan Ganga Institute of Technology & Management
Bhopal, M.P, India

ABSTRACT
This paper involves the development of an application through which user can monitor the computers over a network and can perform the administrative tasks such as accessing programs, hardware details, monitoring users, installation of any software on remote System and also can share all the information with compressed and encrypted format.

General Terms
Remote Access, Networking, Communication.

Keywords
Establishing Connection, Data Compression, Encryption and Decryption, RANC Protocol, Adaptive Protocol.

1. INTRODUCTION
The Remote Machine Control System is a Software tool which is an application through which user can monitor the computers over a network and can perform the administrative tasks such as accessing programs, hardware details, monitoring users. Using this software the two systems will be connected to each other and also the systems running on the different environment can communicate to each other. Using this software the user can initiate different processes in remote machine and can gather much information from the remote machine like list of drives, list of files and copying or sending the file from server machine to client machine. We fetch software and hardware information from window registry editor . Under registry folder there are sub folders for software from where we can fetch information about particular products and similarly we will get information about hardware of client. Using this project we can shutdown , logoff remote machine and also able to send different types of message according to requirement. Using this software the user can control the mouse of the remote machine. It can lock or release the mouse as per its requirements. Using this software the user can view the desktop screen of the remote machine through which it is connected. The remote machine desktop will be shown on the user screen itself. Using this software the server can view the activities performed by the clients among each other and can provide the facility to perform the administrative tasks. Data will be compressed for avoiding overhead and congestion in network applying data compression algorithm . By applying encryption and decryption algorithm security will be provided for confidential data transmission. There are many types of software available in the market that can help to import the remote computer to your computer based on Remote Access Network Computing using remote Frame Buffer Protocol. Software like PC anywhere by semantic and GVNC of Linux works exactly how this system should work and both work fine, But this software let the client gain full access to the control of the server which is not required in the proposed system. Previously the software can perform the administrative tasks but the clients have the full access of the remote system which is not required in RANC software, it has just the connection and by using it the client system can access the remote system. With the wide spread utilization of object technology, it has become more and more important to employ the object oriented paradigm in distributed environments as well. This raises several inherent issues, such as references spanning address spaces, the need to bridge heterogeneous architectures[11]. The Java environment, designed by Sun Microsystems, has probably experienced the greatest evolution recently. From the broad spectrum of the Java platform segments, we will focus on Java RMI [12]. In the last decade, distributed systems engineers have often relied on middleware platforms to increase their productivity. Residing between the operating system and distributed applications, middleware systems provide abstractions that hide from application developers several details inherent to distributed programming, such as network communication primitives, data marshalling and unmarshalling, failure handling, heterogeneity, service lookup and synchronization. Java RMI [13] – are the most common middleware platforms. In such systems, developers invoke methods on remote objects using the same syntax of local invocations; therefore, code to handle distributed communication looks similar to code that handles communication in centralized systems[14].

2. OVERVIEW OF THE PROPOSED SYSTEM

2.1 System Objectives
Based on the feasibility study of the requirements provided by the client, the Customer Requirement Analysis Document is prepared to formally seek clarifications from the customers. After the customer's needs (both implied and stated) are understood and a detailed analysis of the risk factors is made the following tasks are handled.

- An outlay of general work schedule is formed.
- An estimate of the time required is made.
- Resources and manpower to be involved in the project are identified
- System objectives are formulated.

2.2 Proposed System
The proposed system is designed in such a way that most of the above problems found in the existing system are eliminated. Some of the features expected from the proposed system are as follows:-

- User friendly interfaces such that any person accessing the computer can easily use this system.

- Faster and quicker access to the server machine.
- Disallowing the extra burden over the net.
- The user can view the desktop of the server and know what is happening.
- The system should be flexible enough to facilitate upgrades.
- The system should be able to generate appropriate error and exception messages.

2.3 Methodology

2.3.1 Java Programming

Due to exciting features like Simple, Distributed, Robust, Interpreted and Compiled, Secure, Architectural neutral, Portable, High performance, Multi-threaded and Dynamic nature of JAVA, it will be easy to implement the desire specification. And it is best suited for this project.

2.3.2 Remote Method Invocation

Remote Method Invocation (RMI) enables the software to implement distributed Java - to- Java applications in which the methods of remote Java objects can be invoked from other java virtual machines possibly on different hosts. A Java program can make a call on a remote object once it obtains a reference to the remote object in the bootstrap naming service provided by RMI or by receiving the reference as an argument or a return value and due to this accessing of the remote objects it is being used up in developing software. Client can call a remote object in a server and that server can also be a client of other remote objects[6][1]. RMI uses Object Serialization to marshal and unmarshal parameters.

Invoking a remote method on a server object

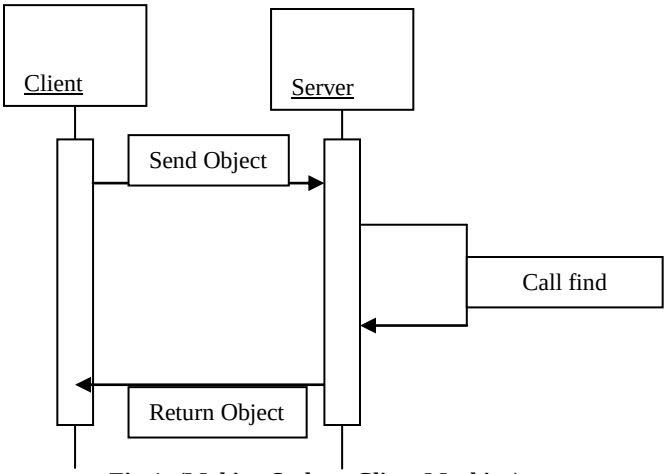


Fig-1: (Making Stub on Client Machine)

Stub is a Java object that resides on the client machine as shown in Fig (1). Its function is to present the same interfaces as the remote server. Remote method calls initiated by the clients are actually directed to the stub.

The stub packages the parameters used in the remote into a block of bytes. This packaging uses a device-independent encoding for each parameter. The process of encoding the parameters is called **parameter marshalling**.

The purpose of parameter marshalling is to convert the parameters into a format suitable for transport from one virtual machine to another. **Information block generated by stub consists of**

- An identifier of the remote object to be used
- A description of the method to be called

The marshalled parameters

On the server side, a receiver object performs the following actions for every remote method call

- Unmarshal the parameters
- Locates the object to be called
- Calls the desired method
- Captures and marshals the return value or exception of the call
- Sends a package consisting of the marshalled return data back on the client.

2.4 Uses of Proposed System

2.4.1 Workstation to Workstation (Windows)

In a PC support environment this would be the most useful feature. With the server software loaded into each machine on the network and the viewer on restricted machines (or carried around on a floppy disk by support personnel) it would be easy for remote diagnostics to be carried out on PCs that were located remotely (perhaps many miles away), either via the company LAN or a dial-up connection to the remote PC. The issue of monitoring/spying also crops up here if it is suspected that a member of staff is not utilizing his/her time correctly then the viewer, if connected in View only mode, can easily "snoop" on the user. A technician or even someone senior who might not have the relevant technical skills could do this in View Only mode no real technical skills are required to start up, view and subsequently shut down the software.

2.4.2 Workstation to Server

It would enable a viewer on any type of workstation (or Windows CE machine) to connect to a server running one of the supported operating systems, so allowing remote control (or monitoring) of that server. The most common uses here would be for server control/monitoring from the support person's desk while at work or when dialing into the server from home (the viewer software being run on a laptop, desktop or Windows CE machine).

2.4.3 Workstation to Workstation (cross-platform)

The server and viewer software is also available for the Macintosh machines it is now possible to remotely run for example a non-Windows-compliant application on a Windows machine so enabling a vast array of Macintosh software to effectively be "run" on Windows PCs (in spite a little slow and with some obvious restrictions on functionality).

3. IMPLEMENTATION OF THE SYSTEM

3.1 Platform Independent

This software is going to be implemented in Java language to make it platform independent, portable, cost effective and it is too simple to implement. It will enhance my skills in programming language as well as in networking discipline. The language java consists of such technologies which will be helpful to me in implementing the RANC software.

3.2 Peer-to-Peer Communication

The Java Swing is been used up to make the GUI more effective. In implementing the software RANC it will help in the peer-to-peer communication.

It will also help RANC software to communicate among the computers running on different platform. It is a part of Java Foundation Classes (JFC) that implemented a new set of GUI Components. A number of packages that are presented in swing make it easy to design.

Swing is a set of classes that provides more powerful and flexible components than are possible with the AWT[2][3]. In addition, to that the familiar components such as buttons, check box and labels swings supplies several exciting additions including tabbed panes, scroll panes, trees and tables. Even familiar components such as buttons have more capabilities in swing. For example a button may have both an image and text string associated with it. Also the image can be changed as the state of button changes. Unlike AWT components swing components are not implemented by platform specific code instead they are return entirely in JAVA and therefore are platform-independent[4]. The term lightweight is used to describe such elements. The number of classes and interfaces in the swing packages is substantial.

3.3 RANC Protocol

The RANC protocol is a simple protocol for remote access to graphical user interfaces. It is based on the concept of a remote frame buffer or RFB[6]. In the past we have tended to refer to the RANC protocol as the RFB protocol, so you may have seen this term in other publications. The protocol simply allows a server to update the frame buffer displayed on a viewer. Because it works at the frame buffer level it is potentially applicable to all operating systems, windowing systems and applications. It includes X/Unix, Windows 3.1/95/NT and Macintosh, but might also include PDAs, and indeed any device with some form of communications link. The protocol will operate over any reliable transport such as TCP/IP[6].

It is truly a "thin-client" protocol that has been designed to make very few requirements of the viewer. In this way, clients can run on the widest range of hardware and the task of implementing a client is made as simple as possible.

3.4 Rectangular updates

The display side of the protocol is based around a single graphics primitive: "put a rectangle of pixel data at a given x, y position". It might seem an inefficient way of drawing arbitrary user interface components. But because we have a variety of different encoding schemes for the pixel data, we can select the appropriate scheme for each rectangle we send and make the most of network bandwidth, client drawing speed and server processing speed.

The lowest common denominator is the so-called raw encoding, where the rectangle is simply pixel data sent in left-to-right scan line order. All clients and servers must support this encoding. However, the encodings actually used on any given RANC connection can be negotiated according to the abilities of the server, the client and the connection between the two systems[1].

The copy rectangle encoding for example, it is very simple and efficient and can be used when the client already has the same pixel data elsewhere in its frame buffer. The server simply sends an x, y coordinate giving the position from which the client can copy the rectangle of pixel data. It means that operations such as dragging or scrolling a window, which involve substantial, changes to the screen may only require a few bytes. Most clients will support this encoding, since it is generally simple to implement and saves bandwidth.

A typical workstation desktop has large areas of solid colors and of text. Some of our most effective encodings take advantage of this by efficiently describing rectangles consisting of one majority (background) colors and 'sub-rectangles' of different colors.

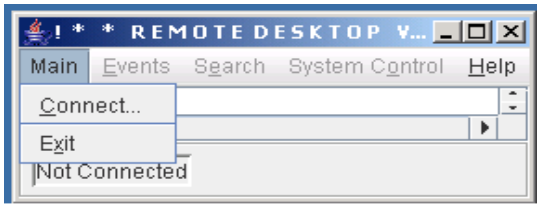
3.5 Adaptive update protocol

A sequence of these rectangles makes a *frame buffer update* (or simply *update*). An update represents a change from one valid frame buffer state to another, so in some ways is similar to a frame of video, but it is usually only a small area of the frame buffer that will be affected by a given update. Each rectangle may be encoded using a different scheme. The server can therefore choose the best encoding for the particular screen content being transmitted and the network bandwidth available.

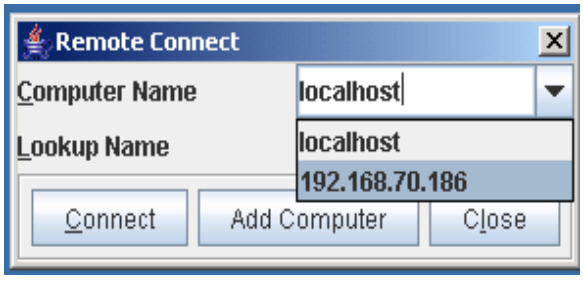
The update protocol is demand-driven by the client. That is, an update is only sent by the server in response to an explicit request from the client. It gives the protocol an adaptive quality. The slower the client and the network are the lower rate of updates becomes. Each update incorporates all the changes to the 'screen' since the last client request. With a slow client and/or network, transient states of the frame buffer are ignored, resulting in reduced network traffic and less drawing for the client. It also improves the apparent response speed.

4. RESULTS

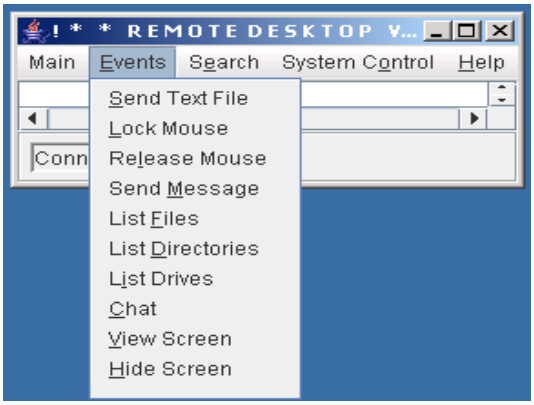
Initial Interface between Client and Server



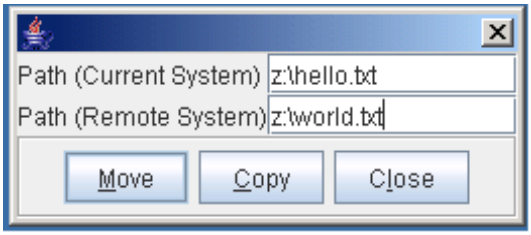
Getting Connected to the Remote System.



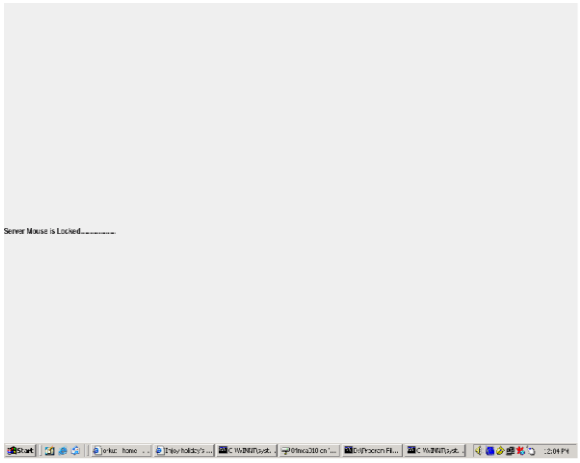
Modules working among the Client and Server systems



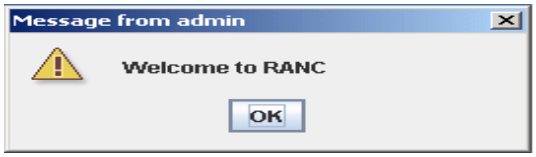
Sending Text Files to remote system.



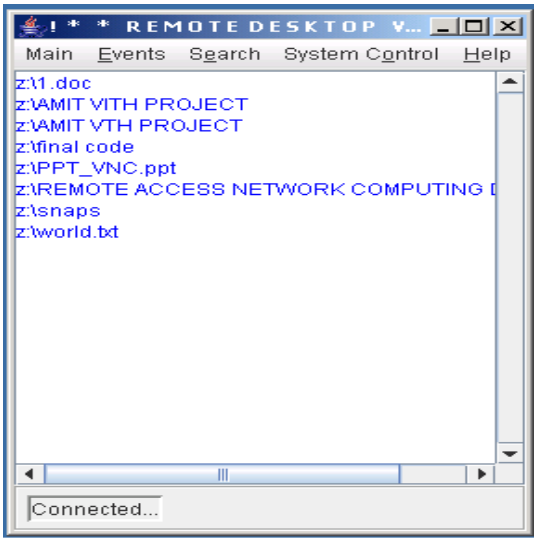
Control on Peripheral Device i.e. mouse



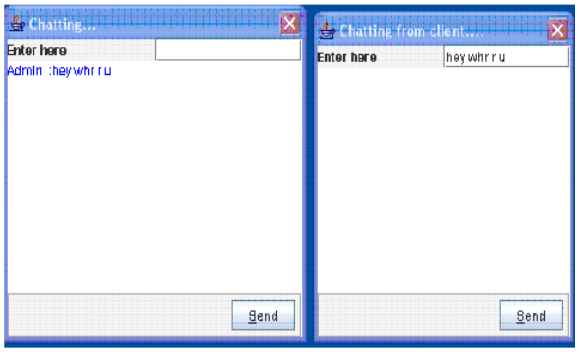
Sending messages to remote/server system.



Description of the Remote system i.e. Files



Chatting service provided by the software.



5. CONCLUSION

A system can not be a perfect system in fact no system can be a perfect system. There is lot of room for improvement in this system which includes: -

- Providing the facility for the client to send the keyboard and pointer events.
- Providing the facility for a user id and password to make system more stable.
- Making the system more versatile by recording the server to client messages or in other words the client can store the changes in the server's desktop.
- Will be able to make the Server system shutdown and restart facility by the Client system.
-

6. ACKNOWLEDGEMENT

The authors wish to thank the management of Gyan Ganga Institute of Tech & Mgmt, Bhopal for their constant encouragement for completion of this work. The authors want to acknowledge Mr.D.Suresh Babu for his co-operation for completing this paper, also to Mr.S.M.RajBharath for his encouragement during this work

6. REFERENCES

[1] Orielly "Java Network Programming" 3rd edition.

[2] Deitel "Advanced Java2 platform How to Program".

[3] Bruce Eckel "Thinking in Java".

[4] "Core Java-1 and Core Java –2", published by Sun Microsystems.

[5] Herbert Scheldt, "The Complete Reference JAVA 2", Tata McGraw-Hill Edition.

[6] Douglass E. Comer "Internetworking with TCP/IP Principles, Protocols, and Architecture" 5th edition.

[7] Roger S Pressman, "Software Engineering – A Practitioner's Approach" McGraw-Hill International Edition, 2002.

[8] Sun Microsystems website, http:// www.java.sun.com

[9] Research Document of Virtual Networking website, http://www.uk.research.att.com

[10] Virtual Networking website, http:// www.vnc.com

[11] Plasil, F., Stal, M.: 1998, An architectural view of distributed objects and components in CORBA, Java RMI and COM/DCOM. Software - Concepts and Tools

[12] Sun Microsystems: Java Remote Method Invocation Specification. October1997,http://java.sun.com/products/JDK/1.1/docs/guide/rmi

[13] Wollrath, A., Riggs, R., and Waldo, J. (1996). A distributed object model for the Java system. In *2nd Conference on Object-Oriented Technologies & Systems*

[14] Fernando M Q Pereira, Marco T O Valente, Wagner S Pires, Roberto S Bigonha, and Mariza A S Bigonha. Tactics for Remote Method Invocation

AUTHORS PROFILE

Randhir Kumar was born in 1984 in Bihar, India. He completed his graduation in Computer Application from MCRP University, Bhopal & post graduation in Computer Application from Vellore Institute of Technology, Vellore, India**.** He was Engineer at Honeywell Technology Solution Lab India till 2009. Currently he is Sr.Lecturer in Department of Information Technology, Gyan Ganga Institute of Technology & management, Bhopal (M.P), India. His major research areas are data mining, web mining and Image Processing, Networking.

Akash Anil was born in 1985 in Bihar, India. He completed his B.Tech from BPUT University, Orrisa, India. He is currently with Gyan Ganga Institute of Technology & management, Bhopal (M.P), India. His major research areas are Image Processing, digital image processing and network security.