KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)
Department of Computer Science and Engineering
Coimbatore – 641006

April 2003 P-988

REMOTE NETWORK MONITORING

PROJECT REPORT

Submitted in partial fulfillment of the
requirements for the award of the degree of
Master of Computer Applications
Of Bharathiar University, Coimbatore

Submitted By

Karthik Kumar V.S.
Reg. No.: 0038M1029

INTERNAL GUIDE

Mr.K.Ramasubramanian M.C.A
Lecturer,
Dept of Computer Science and Engineering
Kumaraguru College of Technology
Coimbatore-641006

CERTIFICATE

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)
Department of Computer Science and Engineering
Coimbatore – 641006

April 2003

Certificate

This is to certify that the project work entitled

"REMOTE NETWORK MONITORING"

Submitted to the

Department of Computer Science and Engineering
KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University)
Coimbatore – 641006

In partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of original work done by **Karthik Kumar V.S.** Reg. No. **0038M1029** during the period of study in the department of computer science and Engineering. Kumaraguru College of Technology, Coimbatore under my supervision and this project work has not formed the basis of award of any Degree / Diploma / Associate ship / Fellowship or similar title any candidate of any university.

[illegible]
Head of Department.

[illegible]
Staff in charge

Submitted to university Exam held on 18-4-03

[illegible] 16/04/2003
Internal Examiner

[illegible] 16/4/03
External Examiner

DECLARATION

DECLARTION

I hereby declare that the project entitled "***REMOTE NETWORK MONITORING***" submitted to **Kumaraguru College of Technology**, Coimbatore afflicted to Bharathiar University as the project work of Master ***of Computer Applications*** degree, is a record of original work done by me under the supervision and guidance of **Mr. Ramasubramanian M.C.A.,** *Lecturer, Kumaraguru College of Technology, Coimbatore.* And this project work has not formed the basis of award of any Degree / Diploma / Associate ship / Fellowship or similar title any candidate of any university

V.S. Karthik Kumar

Place : COIMBATORE

Date : 7-4-03

Signature of Candidate

(Karthik Kumar V.S.)

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

Here I would like to take this opportunity to thank and appreciate all those who have been with me throughout the project period.

I express my profound respect and sincere gratitude to Dr. K.K. Padmanaban, Ph.D, Principal, Kumaraguru College of Technology, Coimbatore, for his kind cooperation in allowing me to take up this project work.

I record my sincere thanks to Dr. S.Thangasamy, Ph.D, Head of the Department, Computer Science and Engineering, Kumaraguru College of Technology.

I am greatly privileged to express my deep gratitude to my guide Mr. K. Ramasubramanian, M.C.A., Lecturer, Dept of Computer Science and Engineering, Kumaraguru College of Technology, for his valuable advice and encouragement.

I also owe my sincere thanks to Mr.Arumugam Muthukumar, M.Sc., M.C.A., M.Phil, Course coordinator, Master of Computer Applications, Kumaraguru College of Technology, Coimbatore for his guidance and immense support throughout my project work.

I express my sincere and heart felt thanks to Mr.S.Ganesh Babu, M.C.A., and Mr.V.Vijilesh, M.C.A., Lectures, Dept of Computer Science and Engineering, Kumaraguru College of Technology, for their valuable advice and encouragement to finish this project.

Karthik Kumar V.S.

SYNOPSIS

SYNOPSIS

"REMOTE NETWORK MONITORING" is a utility software developed for Kumaraguru college of Technology. This Software takes snapshots of the client systems screen that are connected to a Local Area Network(LAN) and displays it on the server. Also it captures the currently running processes in the client system connected to the network.

Provision for viewing the system information of client system from the server is made.

It is also possible to control the remote client from the server and the operations involved in the clients are Log Off, Shut Down or Locked.

A Chat Module is also available for text communication between the client and the server.

The main users of this software are System Administrators who wish to monitor the client activities at any instance of time. The product is developed using Microsoft Foundation Class Library & Win32 APIs under Visual C++ 6.0. The system is based on the TCP/IP family of protocols.

CONTENTS

CONTENTS

INTRODUCTION

1. INTRODUCTION

1.1. Project Overview

"**REMOTE NETWORK MONITORING**" is developed using Microsoft Foundation Class Library & Win32 APIs under Visual C++ 6.0. The system is based on the TCP/IP family of protocols. The functions of this utility software are capturing the screen, system information of the client, remote control of the client, text communication between the server and the clients. The product is divided into two major categories – Server side and Client side.

The Client Program is installed in client systems that are to be monitored and the Viewer Program is installed in the server of the Local Area Network.

The Main features of the product are as follows:

- Capture the client screen

  The first feature of this product is capturing the client screen and to display them in the server. Provision for saving the captured image is also provided. Option for automatically saving the captured screens is provided.

- Capturing the active processes & terminating a client process from the server

  Another feature of this product is capturing the processes that are currently being executed in the client system. These processes include both foreground and background processes. Also provision is made to kill a client process or application from the server itself.

➢ System Information viewer

The system information of the client can be viewed from the server. The hardware information such as OEM-ID, Number of Processors, page size of RAM, the Processor Type, Minimum Application Address, Maximum Application Address and Active Processor Mask can be viewed.

The System Name and the User Name of the system can be viewed. The System Directory, Windows Directory, the Environment Variables set on the client machine can be viewed.

The Mouse Information such as whether the buttons are swapped, the speed and its threshold value can be viewed. Also provision is made to control the speed of the mouse from the server.

The Version of the client's operating system can be viewed. The Software's Installed in the Client Machine can also be viewed.

The Protocol's Installed in the Client Machine can also be viewed.

➢ Remote Control

The Client Machine can be logged off, Shut down or locked in case the client is in a Windows N.T. platform from the server.

➢ Chat

The Chat Program in the Client and the Server enables the users to communicate between them in the form of Text Messages. Any number of users can participate in the chat. It is in the form of Open Chat.

1.2. Organization Profile

About

Kumaraguru College of Technology (KCT), Coimbatore is a private co-educational Engineering college started in 1984 under the auspices of Ramanandha Adigalar Foundation, a charitable educational trust of Sakthi Group. The college has obtained Technical Institution Par Excellence ISO 9001:2000.

Under the able guidance and patronage of Arutselvar Dr.N.Mahalingam, Chairman, Sakthi Group and Prof. K.Arumugam, Correspondent, the college has developed excellent facilities and resources such as spacious classrooms, seminar halls, well-equipped laboratories, workshops, dedicated Internet (576 kbps) connectivity and well-qualified faculty.

Currently the college offers 13 under-graduate (B.E., B.Tech., B.Sc.),10 post-graduate (M.E., M.Sc., M.C.A) and 1 Ph.D. programs under affiliation to the Anna University and with the approval of All India Council for Technical Education (AICTE).

The college has 11 academic departments and 2 research centres each headed by a competent and experienced professor. Five of the departments have also been accredited by National Board of Accreditation. Altogether the college has more than 155 well-qualified teaching staff and about 145 supporting technical staff. The combined student intake during current year is 666 and the total number of students on roll is 2343.

The value of the education and training imparted by the college is highlighted by interest shown by leading companies for on-campus recruitment. In the last academic year more than 200 students have been placed in through our Placement centre.

Mission

Kumaraguru College of Technology is committed to provide quality education and training in Engineering and Technology to prepare the students for life and work, equipping them to contribute to the technological, economic and social development of India. The college pursues excellence in providing training to develop a sense of professional responsibility, social and cultural awareness and set the students on the path to leadership.

Quality Policy

Kumaraguru College of Technology strive to achieve customer satisfaction by providing Quality education and training in Engineering and Technology in a congenial and disciplined environment through

- Involvement at all levels
- Upgradation of facilities and human resources
- Commitment to continual improvement.

SYSTEM STUDY AND ANALYSIS

2. SYSTEM STUDY AND ANALYSIS

2.1. Existing System

There is no such system which has all the functionality but as individual modules. The administrator has to switch over software for each and every functionality needed. It has also has the limitations of

- In screen capture the image captured is not sharp.
- Doesn't provide simple and efficient user interface.
- No sufficient interaction between the user and the software.

2.1.1. System Analysis

System analysis is an activity that encompasses most of the tasks collectively called as Computer Science Engineering. This is the most important step in a software project where we get a general idea about the needs of the end-users by having man-to-man conversation with them, and about the various conditions and restrictions that have to be taken care of while developing the software.

- Identify the user's need
- Evaluate the system concept for feasibility
- Perform economic and technical analysis
- Allocate functions to hardware, software, people, database and other system elements
- Establish cost and schedule constraints
- Create a system definition that forms the foundation for all subsequent engineering work

2.1.1.1. Identification of need

As a first step in the analysis of the system, the end-users of the proposed project were met to get first hand information regarding their needs and wants. Ideas from both the sides were exchanged in order to get a standard and satisfactory system.

2.1.1.2. Feasibility Study

The feasibility study is carried out to test if the proposed project is worth being implemented. Given unlimited resources and infinite time, all projects are feasible. Unfortunately, such results and time are not possible in real life situations. Hence it becomes both necessary and prudent to evaluate the feasibility of the project at the earliest possible time in order to avoid unnecessary wastage of time and effort and professional embarrassment over an ill-conceived system. The following feasibility studies were carried out for the proposed project, namely:

2.1.1.2.1. Technical feasibility

These are the technical feasibility constructs

- The memory capacity of the existing hardware is quite sufficient for the execution of the system
- The speed of the existing hardware and the system is quite sufficient
- Technical enhancement may be needed in this system in future, and it will not force barriers to estimated budget

Thus a through study reveals that this project is technically feasible.

2.1.1.2.2. Economic feasibility

The cost of the system is evaluated here

- There is no extra cost needed for implementing the system, because this organization already has an local network facility and windows environment
- Since it is very easy to use, no training is needed. So training cost can be avoided.
- This system is flexible so that further enhancement is possible according to the future needs of the users.

2.1.1.2.3. Behavioral feasibility

People are inherently resistant to changes, and if the user needs a sufficient amount of training, it would result in the expenditure of the user's time, which is precious enough for them and also for the organization. So, generally the user would reject a proposal if it were going to consume much amount of time an effort from them. So, the outcome of establishing of project should bring user-convenience and satisfaction.

As, the suggested project is much more advantageous and requires less amount of time and effort for the users, they readily welcomed the proposed system, when asked for approval.

2.2. Proposed system

The aim of the proposed system is to have such a system in the college to monitor the student activities in the lab and to manage the lab efficiently by the lab in charge. The proposed system is a Win32 Windows based application, developed using C++, Microsoft Foundation Class and Win32API under Visual C++ 6.0.

It is base on the TCP/IP family of protocols.

Proposed System Objectives:

- Provide simple & efficient user interface.
- Captures the client screen.

- Captures the currently running processes also.
- Access System Information of client from the server itself.
- Control the remote client from the server.
- Chat between clients and server.

After development the system will be put to various testing procedures.

The testing procedures would ensure that the new system is error free from bugs. All the functional part of the system would be tested for defects. This would optimize the system in the usage of memory.

The processing schemes take the concept of time complexity by optimizing the loop structures and memory allocations to dynamic variables. This would ensure optimized usage of resources making the system more fast and reliable. The coding structure and the standard would ensure that there are no memory leaks and that the allocated memory is returned to the free store.

The proposed system is to have tool tip help to the menus, tool bar, item etc. this would ensure an improved way to access the functions of the systems. High lighting facility in the data display screens would ensure quick viewing of data.

Thus the proposed system is to take advantage of all the new technologies and ensures a smooth functioning.

2.3. Requirements on New System

2.3.1. Scope

- Efficiently monitor the clients connected to the network.
- Minimize the system administrator's desk time
- The system provides an easy user-friendly interface to the users of the system.
- User authentication is provided by means of a user name & password.
- View the system information
- Log off or Shut down the remote clients.

2.3.2. General Description

2.3.2.1. Product Perspective

The perspective of the projects is to provide the user with an effective means for monitoring the client system, view system information & take control of the client system.

- Provision for adding new user to the system and allowing the existing user to modify the user-name and password .
- Provision for selecting the inputs from a list of active clients.
- Provision for capturing screens & processes separately.
- Provision for log off, shut down, lock the client from the server.
- Provision to chat between clients and server.
- Provision to kill a client process or application from the server.
- Provision to view client system information

2.3.2.2. Product Functions

The following points would briefly explain the functions of the proposed system.

- Capture full screen/active window of client system.
- Capture foreground / background processes of client system
- Kill a client process or application from the server.
- View system information.
- Log off, shut down and lock the client from the server.
- Chat between the clients and server.

2.3.2.3. General Constraints

The general constraints regarding this software are:

- A time gap occurs between sending the request and display the output.
- Whenever a new system is added to the network, its details should be entered.
- The server program installed in the client systems should be loaded at time of system startup so that they remain as a background process & accept the requests.
- Except the administrator no other person should access this program excluding chat.

2.3.2.4. Specific Requirements

2.3.3.1. Functional Requirements

List of inputs:

The users of the system should be provided with a point & click interface, i.e., the users of the software mainly users mouse to provide inputs.

Initially the user should authenticate themselves by means of a user name and password. Once the authentication process is over he/she is allowed to perform the required operations provided by the software.

The only input for the server program (installed in client systems) is that the user should start the server so that it can accept incoming requests.

The primary input is the IP address of the client system to be captured.

2.3.3.2. Performance Requirements

Security

Security should be implemented by means of a user name/password validation process. Only authorized users should be allowed to access the software.

Availability

Availability is the probability that a program is operating according to requirements at a given point of time. The availability is an indirect measure of the maintainability of software.

The successful function of the software depends on the validity of the inputs given to it. If the data entered is not appropriate or data is missing the system should indicate possibility of an error.

Capacity

Capacity measures number of systems a software can access. The software should manage up to 5 user sessions simultaneously.

Response Time

Response time is the time with in which a system identifies the instruction of the user and responds to it.

The time required to capture the screen/process from the client system and to display it in the server should be approximately 5 seconds.

2.3.3. Design Constraints

2.3.3.1. Hardware Limitations

- A Pentium processor of 166Mhz speed(to make sure that application does not take too long to run)
- A random access memory of 64 MB.
- A mouse and keyboard
- LAN(Ethernet Interface)

2.3.3.2. User Interface and Screen Formats

It is required to maintain certain GUI standards during the development of this system. The user can either use a Mouse or a Keyboard to operate the product.

The user should be provided with menus, dialog boxes and controls such as static text, edit box, group box, buttons, scrollbars, radio buttons, check box etc. The system should make best use of the resources provided by Visual C++ IDE.

2.3.3.3. Operation required by the user

The users of the system should access the software after providing a user-name & password. The only operation required by the user is to provide inputs to the system. The user should provide valid inputs to the system.

2.4. User Characteristics

The system has been designed as a very easy to understand point-click interface system. The whole system is partitioned into two-Client side and Server side depending on the mode of operation of the system.

The main users of the system are system administrators. Since the product is user friendly the user need to possess only minimum data very knowledge. They should also have knowledge on basic networking concepts.

PROGRAMMING ENVIRONMENT

3. PROGRAMMING ENVIRONMENT

3.1. Hardware Configuration

Server System

Processor	: Pentium R III
Processor Speed	: 500 MHz
RAM	: 128 MB, SDRAM
Cache Memory	: 512 KB
Hard Disk	: 20 GB
Display	: 17” Color Monitor (1024 * 768)
Mouse	: Microsoft 2 button mouse
Keyboard	: Samsung 105 Key

Client System

Processor	: Pentium R III
Processor Speed	: 350 MHz
RAM	: 64 MB, SDRAM
Cache Memory	: 512 KB
Hard Disk	: 8 GB
Display	: 17” Color Monitor (800 * 600)
Mouse	: Microsoft 2 button mouse
Keyboard	: Samsung 105 Key

Networking

Network Adapter	: Intel 21040 based Ethernet controller (16 Bit)

3.2. Description of Software and Tools used-Resources for the choice

Operating System : Windows NT 4.0
Programming Language: C++ and MFC
Programming Environment: Visual C++

3.2.1. About Windows NT 4.0:

Windows NT is a 32-bit, preemptive multitasking operating system that belongs to Microsoft Windows family of operating system products. Microsoft Windows NT is the most powerful and reliable Operating System. It is a robust & secure 32-bit network operating system with the familiar Windows 95 user interface. Some of its other features include multi user, multitasking and multithreading capabilities.

Windows NT is a powerful tool for centralized network administration and security. It provides highly reliable fault tolerance features. It offers remote access server capabilities. Windows NT has the capability to run more than one process at the same time. Not all networks are prone to attack, and Windows NT does not impose performance penalties by applying cryptographic techniques to all network traffic. Instead, its philosophy is to support specific applications that must cryptographically protect data in transit across a network. However, it does use some common sense and basic cryptographic techniques in its standard, underlying protocols. The advances security features of Windows NT Workstation NT Workstation 4.0 can be used in a variety of network environments.

- Windows NT server is enabled to work as a network operating system. Unlike previous versions of Windows. Windows NT really is a complete, true operating system in itself, not relying in DOS for lower level function. When a computer with Windows NT starts up, it starts immediately in Windows NT.

- Windows NT is a 32-bit Operating System with a graphical interface. It is not a revision of any of the other Windows operating systems such as Windows for Workgroups 3.x, but rather an entirely new operating system.

The feature provided includes

Portability: Unlike most operating system, Windows NT can run on variety of platforms. This flexibility can be a great advantage when implementing a computer strategy for an organization.

Multi-tasking Operations: from the perspective of the end user, multi-tasking means that different type of application can be running in the background.

File Systems: Windows NT supports a variety of file systems, including FAT, NFTS and VFAT.

File Allocation Table (FAT): The file system used in DOS.

New Technology File System (NTFS): The file system introduced by Windows 95

Security: Windows NT's security features such as a mandatory logon procedure, memory protection, and auditing and limited network access have been developed.

Support for many clients: A wide variety of clients can serve as workstation on a Windows NT network such as Windows 3.x, Windows for Workgroups, MS-DOS, Windows 95 and Windows NT Workstation.

Windows NT has been designed to operate well on the same network as Novell NetWare and Unix file, print and application servers. Windows NT includes tools to provide seamless connectivity to Net/ware servers.

Storage Space: Windows NT supports a virtually limitless amount of memory and hard disk space.

RAM: Windows NT supports 4 gigabytes

Hard Disk Space: Windows NT supports 16 hex bytes.

3.2.2. Microsoft Foundation Class Library

The MFC Library is a large set of C++ classes that encapsulates most of the Windows API Currently about 150 classes provide access in C++ form to windows, dialog boxes, GDI objects and other standard Windows elements. The library also includes application framework classes that provide the basic requirements of a Windows application framework classes that provide the basic requirements of a Windows application. Using these classes you can write very little code and yet create complete, fully functioning Windows programs.

MFC is fast becoming the standard set of C++ classes used for writing Windows applications. This has been fueled by Microsoft's decision to license MFC to other compiler vendors. MFC currently comes with the following compilers:

- Microsoft Visual C++
- Borland C++
- Watcom C++
- Symantec C++

MFC is also striving to become a real multiplatform development environment. Microsoft distributes highly compatible versions of MFC for Intel, Macintosh, Alpha and MIPS platforms.

3.2.3. MFC App Wizards

Custom AppWizards can obviously be used to automate repeated tasks, but they're also great teaching tools. While sample code is nice, a custom AppWizard can be much more useful, because users can see how the pieces they specify get put into the resulting project. However, there are cases for which a sample makes more sense than an AppWizard. An AppWizard isn't always useful for a piece of technology that will be tangential to the developer's main goal. A widget, for example, might not be ideal for an AppWizard, because users will want to use it in the context of a grander scope—other than some early Java adapters a few years ago, I've never seen anyone get too excited about an application that's functionally centered around a widget. However, if you're writing a graphics framework, an AppWizard might be very useful, because users will want to build their applications within the context of your framework.

A project need not have the scope of MFC to justify an AppWizard. Smaller APIs can still benefit from having small AppWizards, especially if there are common applications that would make use of the target API. APIs that depend on process are good candidates as well. For example, a grid vendor might find use for an AppWizard that allows the user to specify an ODBC source and a table name, among other parameters. This might produce an application that allows users to view and modify a table using the grid. A sample could be used for this, but the AppWizard has four major advantages in this case:

- ➢ The user can create and compile a useful application immediately after installing a product.
- ➢ The user can search for the parameters entered into the wizard and see how customization should be implemented.
- ➢ The user has a meaningful application.
- ➢ The AppWizard can be enhanced to provide more things for the customer. Samples typically stay static over time, because as they get more complex, it gets harder for the customer to change them to fit new needs.

3.2.4. About Visual C++ IDE:

An IDE, or Integrated Development Environment, is a program that hosts the compiles, debugger, and application-building tools. The central part of the Visual C++ package is DeveloperStudio, the Integrated Development Environment (IDE)Developer Studio is used to integrate the development tools and the Visual C++ compiler. You can create a Windows program, scan through an impressive amount of online help, and debug a program without leaving Developer Studio.

Developer Studio Tools

Once upon a time, Windows programmers used simple text editors and tools that were hosted on MS-DOS to create their Windows programs. Developing a program under those conditions was tedious and error-prone. Times have definitely changed. Developer Studio includes a number of tools that you might once have paid extra to purchase.

- An integrated editor offers drag-and-drop and syntax highlighting as two of its major features.
- A resource editor is used to create Windows resources, such as bitmaps, icons, dialog boxes, and menus.
- An integrated debugger enables you to run programs and check for errors.

3.2.5. MFC Socket Classes

MFC now supports several classes related to WinSock and Internet programming, and the number is growing all the time. Some are more related to specific application protocols than to WinSock itself.

WinSock support from MFC comes in two flavors:

- CAsyncSocket: this class encapsulates asynchronous mode WinSock programming and provides callback functions for event notifications. This is a virtual base class and you must derive a new socket class to use it.
- CSocket: This class is derived from CAsyncSocket to provide a higher level avstraction for working with WinSock. It manages blocking and background processing of Windows messages, to provide the application with a synchronous interface to the underlying asynchronous CAsyncSocket class.

MFC applications that use these classes are linked with WS2_32.LIB.

SYSTEM DESIGN AND DEVELOPMENT

4. SYSTEM DESIGN AND DEVELOPMENT

System design is a solution, a "how to" approach to the creation of a new system. This important phase is composed of several steps. It provides the understanding and procedural aspects details necessary for implementing the system recommended in the feasibility study.

Emphasis is on translating the performance requirements into design specifications. Design goes through logical and physical stages of development.

System design is the development of a blueprint of a computer system solution to a problem that has the same components and interrelationships among the components as the original problem. System design has the following properties

- Schedules design activities
- Works with the user to determine the various data inputs to the system
- Plans how data will flow through the system
- Designs required outputs
- Program specification

4.1. Input design

Input Design is the most important part of the overall system design, which requires very careful attention. Often the collection of input data is the most expensive part of the system. Many errors may occur during this phase of the design. So to make the system study, the inputs given by the user is strictly validated before making a manipulation with it. Thus, by validation it is possible to

- ➢ Produce an effective method of input.
- ➢ Achieve the highest possible level of accuracy
- ➢ Ensure that input is acceptable to and understood by the user staff.

Input design is concentrated on estimating what the inputs are and how often they are to be arranged on the input screen, how frequently the data are to be collected etc.

The Remote Network Monitoring provides many user-friendly features that help the user to interact with the system easily. The input screens are designed in such a manner that avoids confusion and guide the user in correct track. A thorough study has been made on the type and how the input form is to be designed. Some inputs from user may cause severe errors and is strictly validated. This software provides a point & click Single Document Interface to its users. The layout of the input screen is also taken into account. A very good look and feel is provided through the organized arrangement of controls such as menus, edit fields, dialog boxes, buttons etc.

Types of inputs

The nature of the input data is determined from the beginning of the input design. The main input types can be categorized as,

- ➢ External
- ➢ Internal
- ➢ Operational
- ➢ Computerized
- ➢ Interactive

Input screen for the Remote Network Monitoring are very simple and user-friendly. Users are allowed to access the software only after a user authentication process. If irrelevant data is entered message screens are displayed.

4.2. Output Design

Output design generally refers to the results generated by the system. For many end-users, output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application.

The objective of a system finds its shape in terms of the output. The analysis of the objective of a system leads to determination of outputs. The most common type of outputs is reports, screen displays, printed forms, graphical drawings etc.

The outputs also vary in terms of their contents, frequency, timing and format. The users of the output, its purpose and sequence of details to be printed are all considered. If the outputs are inadequate in any way, the system itself is inadequate.

The basic requirements of output are that it should be accurate, timely and appropriate. When designing output, the system analyst must accomplish things like, determining what information to be present, whether to display or print the information, select output medium and to decide how to distribute the output to intended recipients.

The types of outputs are:

External Outputs

These are the type of outputs, whose destination will be outside the organization and which require special attention as they project the image of the organization.

Internal Outputs

These are the type of outputs, whose destination is within the organization. It is to be carefully designed, as they are the user's main interface with the system.

Interactive outputs

These are the type of outputs, which the user uses in communicating directly with the computer.

Remote Network Monitoring provides its users with simple & unambiguous output screens. The output screens are provided with scrollbars so that the users can have a full view of the captured image, captured process, system information.

4.3. Module Design

A module is defined as a collection of program statements with four basic attributes: input and output, function, and internal data.

Modular systems consist of well-defined, manageable units with well-defined interfaces among the units. Desirable properties of a modular system include:

- ➢ Each processing abstraction is a will-defined subsystem that is potentially useful in other applications.
- ➢ Each function in each abstraction has a single, well-defined purpose.
- ➢ Each function manipulates no more than one major data structures.
- ➢ Functions share global data selectively. It is easy to identify all routines that share a major data structure.
- ➢ Functions that manipulate instances of abstract data types are encapsulated with the data structure being manipulated.

Modularity enhances design clarity, which in turn eases implementation, debugging, testing, documentation, and maintenance of the software product.

The Remote Network Monitoring project is divided into two parts based on their functionality. They are the Server side and the Client side.

➢ **Menu**

This module provides the user interface. It contains two options only, one for starting the server and another for stopping the server. Once the server is started, the application is hidden & it waits for incoming requests.

➢ **Communication**

This module establishes a path between the client system and the server system using TCP/IP protocol. Once the connection is established data can be transmitted back and forth. It accepts data from the screen capturing and process-capturing module so that they can be sent to the client side. The connection is disconnected after successful data transmission.

➢ **Screen capturing**

This module captures the client screen. It is this bitmap that is sent to the communication module. The capturing process is done with the help of Win32 APIs.

➢ **Process Capturing**

This module captures the foreground and background process in the system and stores in a file. It is this file that is transmitted to the client. The capturing process is done with the help of Win32 APIs.

The various modules in the client side of the project are

➢ **System Information**

The system information of the client can be viewed from the server. The hardware information such as OEM-ID, Number of Processors, page size of RAM, the Processor Type, Minimum Application Address, Maximum Application Address and Active Processor Mask can be viewed.

The System Name and the User Name of the system can be viewed. The System Directory, Windows Directory, the Environment Variables set on the client machine can be viewed.

The Mouse Information such as whether the buttons are swapped, the speed and its threshold value can be viewed. Also provision is made to control the speed of the mouse from the server.

The Version of the client's operating system can be viewed. The Software's Installed in the Client Machine can also be viewed.

The Protocol's Installed in the Client Machine can also be viewed.

➢ **Control**

The Client Machine can be logged off, Shut down or locked in case the client is in a Windows N.T. platform from the server.

➢ **Chat**

The Chat Program in the Client and the Server enables the users to communicate between them if form of Text Messages. Any number of users can participate in the chat. It is in the form of Open Chat.

➢ **Display**

This module is responsible for displaying the captured screens and processes. They are displayed in separate windows.

4.4. Process Design

A computer procedure/process is a series of operations designed to manipulate data to produce output from a computer system; the procedure may be a single program or a series of programs.

The detailed design of computer procedure follows acceptance by management of an outline design proposal. The aim now is to design procedure/processes as lower levels of detail which will define the detailed steps to be taken to produce the specified computer output (or intermediate stages) from the initial input of data. When complete, these procedure definitions together with data specifications are organized into specification for programmers from which the required programs can be written.

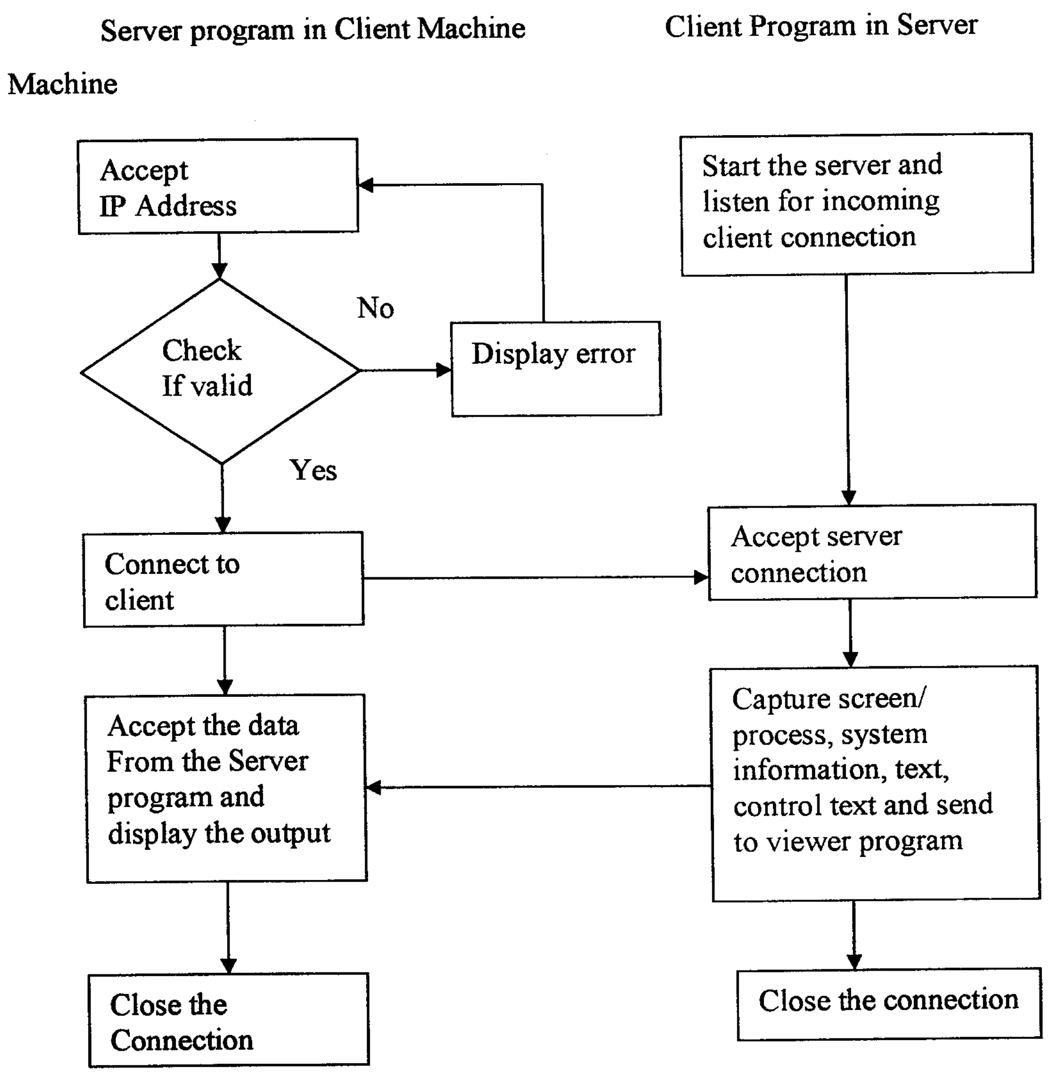
Design tools:

Various tools are used by system analysts to specify computer procedures, eg, narrative, flowcharts and decision tables. The following are some of the most commonly used design tools:

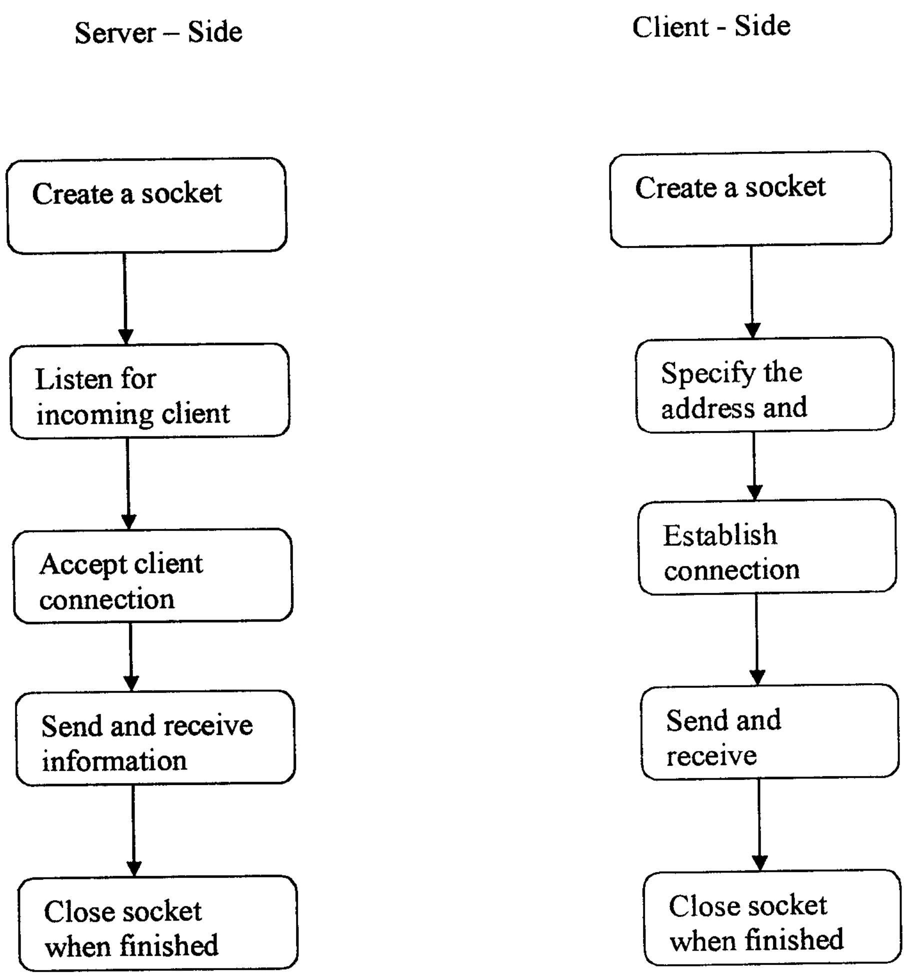
- Flowcharts
- Computer Run Chart
- Computer Procedure Flowchart/Process Diagrams
- Network Chart
- Interactive System Flowchart
- Decision Tables

4.4.1. Process Diagram

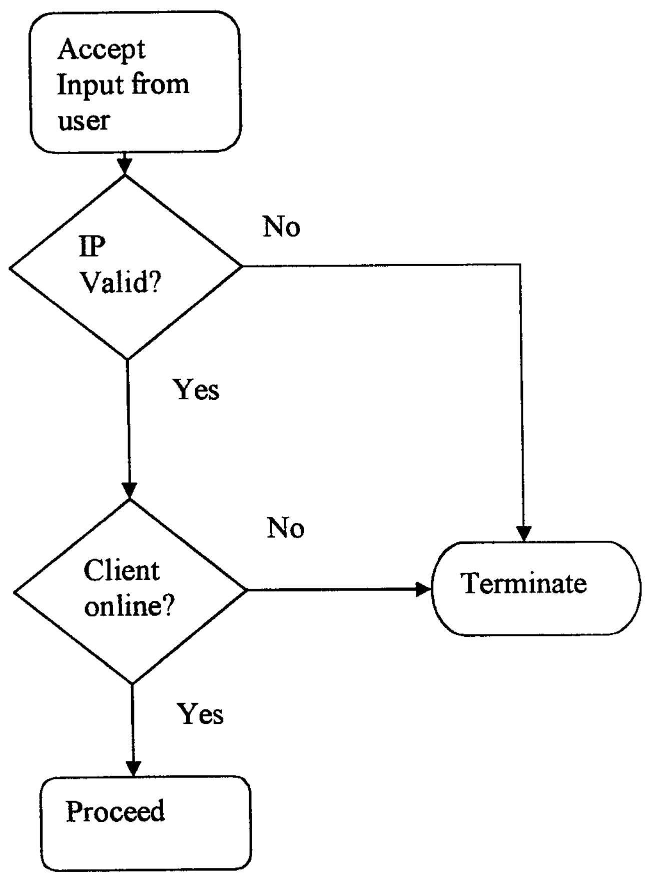
Overall Process Diagram


Server program in Client Machine
Client Program in Server
Machine
Accept IP Address
Check If valid
No
Display error
Yes
Connect to client
Accept the data From the Server program and display the output
Close the Connection
Start the server and listen for incoming client connection
Accept server connection
Capture screen/ process, system information, text, control text and send to viewer program
Close the connection

Client – Server Communication Diagram

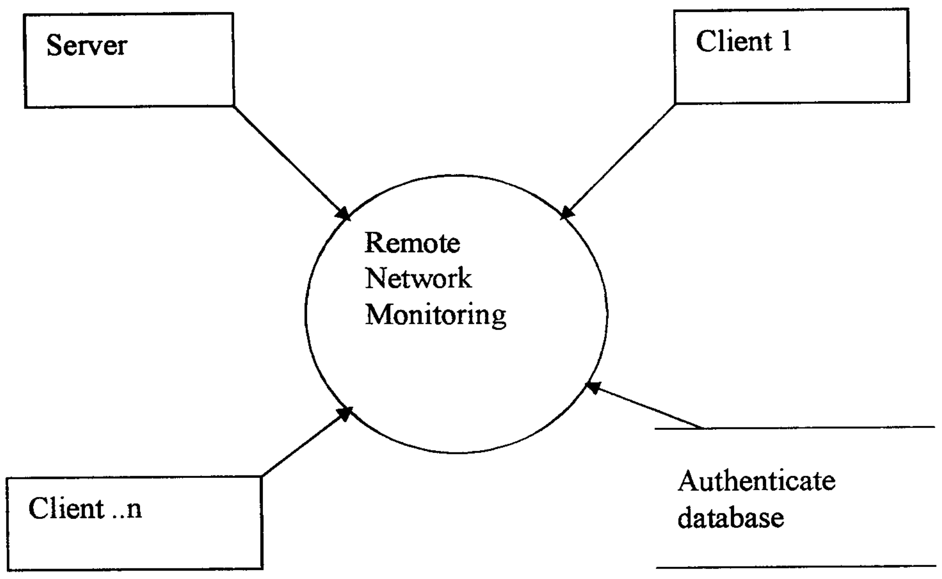


User Input Validation Diagram

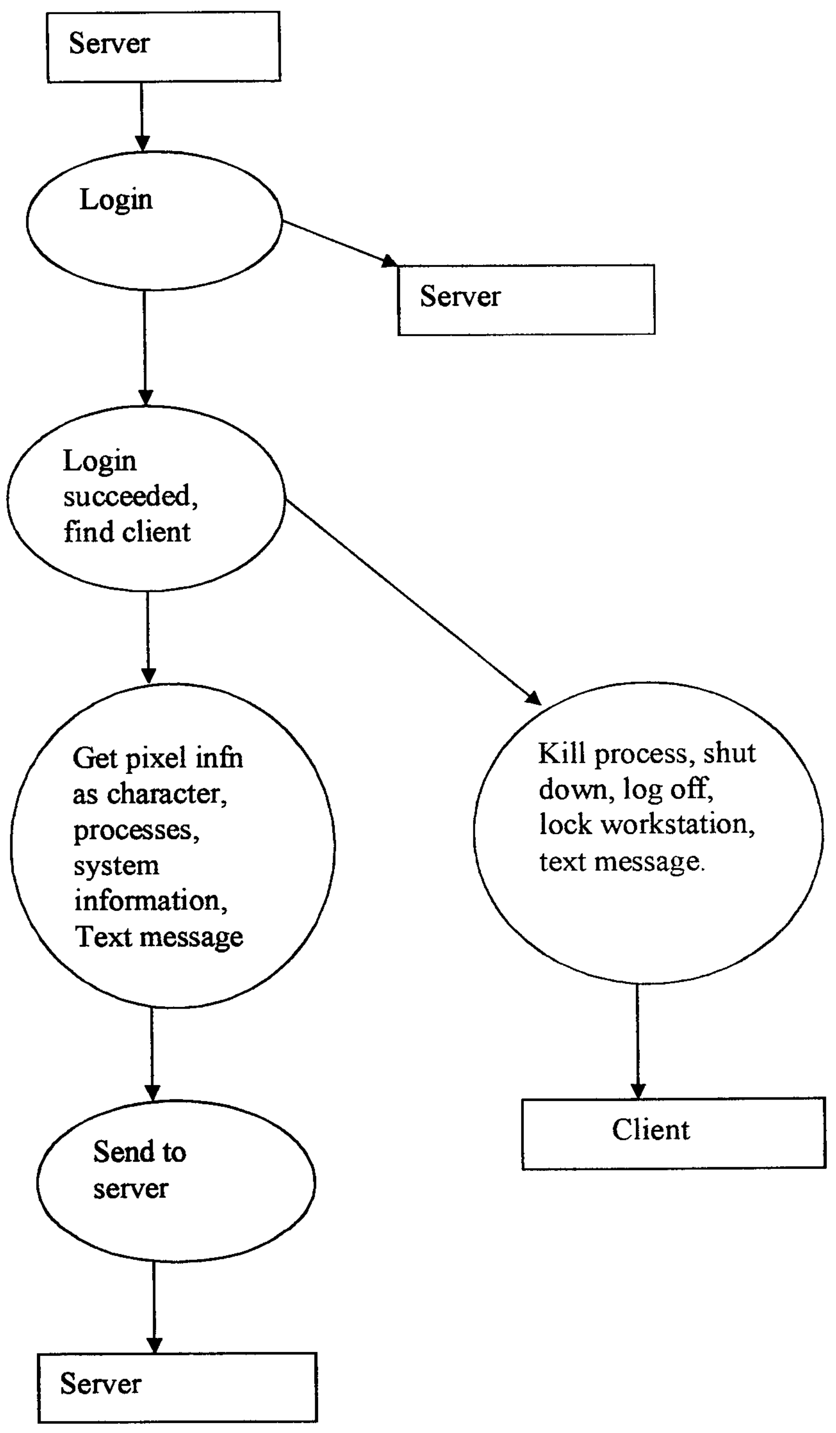


Data Flow Diagram

Level 0 Diagram



Level 1 Data Flow Diagram


Server
Login
Server
Login succeeded, find client
Get pixel infn as character, processes, system information, Text message
Kill process, shut down, log off, lock workstation, text message.
Send to server
Client
Server

SYSTEM TESTING & IMPLEMENTATION

5. SYSTEM TESTING AND IMPLEMENTATION

5.1. System Testing

System testing is the critical element of software quality assurance and represents the ultimate review of specification, design and coding. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved testing includes verification of the basic logic of each program and verification that the entire system works properly. Test case design focuses on a set of techniques for the creation of test cases that meet overall testing objectives. The logical design and the physical design should be thoroughly and continually examined on paper to ensure that they will work when implemented.

When the programmers have tested each program with the test data designed by them, and have verified that these programs link together in the way specified in the computer run chart to produce the output specified in the program suite specification, the complete system and its environment must be tested to the satisfaction of the system analyst and the user.

Objectives of Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has high probability of finding an as yet undiscovered error. A successful test is one that uncovers an as yet undiscovered error.

Testing demonstrates that software functions work according to specifications. In addition, data collected from testing provides a food indication of software reliability and some indication of software quality as a whole. Testing results in the deduction in the number of errors. Critical modules are tested as early as possible.

Software testing for REMOTE NETWORK MONITORING has been done during the pre-implementation stage using various software testing strategies and they are discussed below.

Testing Methods

Unit Testing

Unit testing was performed in order to check whether the developed programs function as specified. Unit testing is mainly used to test the smallest part (i.e.) module of the software design. Using the detailed design descriptions as the guide, important control paths are tested to uncover errors. Unit testing comprises the set of tests performed by the programmer prior to integration of the unit into a larger system.

A program unit is usually small enough that the programmer who developed it can test it in greater detail, and certainly in greater detail, than will be possible when the unit is integrated into an evolving software product.

There are four categories of tests that a programmer will typically perform on a program unit.

Functional tests

It involves exercising that the mode with nominal input values for which the expected results are known, as well as boundary values and special values.

Performance tests

These determine the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

Stress tests

These are those tests designed to intentionally break the unit. A great deal can be learned about the strengths and limitations of a program by examining the manner in which a program unit breaks.

Structure tests

These are concerned with exercising the internal logic of a program and traversing particular execution paths. Some refer collectively to function, performance and stress testing as black box testing while structure testing is referred to as white box or glass box testing. The major activities in structural testing are deciding which paths to exercise, deriving test data to exercise those path, determining the tests coverage criterion to be used, executing the test cases and measuring the test coverage achieved when the test coverage's achieved when the test cases are applied.

All modules of the newly developed system, as soon as they are completed, have been tested and errors found were corrected. Functional test has been carried out on all the modules to determine whether they produce expected results. Structure test was conducted in order to make sure that the internal logic of the modules is free of errors.

Sub - System Testing

This phase involves testing collection of modules, which have been integrated into sub-system. Sub-system may be independently designed and implemented. The most common problems that arise in large software system are sub-system interface mismatches. The sub-system process should therefore concentrate on the detection of interface errors by rigorously exercising these interfaces.

The modules available in the server-side alone are considered as a subsystem. Similarly in the case of client-side also. Each and every module in the respective subsystem was integrated and their interfaces were tested.

The server-side and the client-side are treated as two separate sub-systems. These subsystems and their interface were also tested in this stage.

System testing

The sub-system is integrated to make up the entire system. The testing process is concerned with finding errors, which result from unanticipated interactions between sub-system and system components. It is also concerned with validating that the system meets its functional requirements.

The two subsystems, namely, server-side and client-side, were integrated to form the whole system. The communication between these two sub-systems forms the major interface between them. This communication interface was tested for errors. The entire system was tested in this stage.

Interface testing

Interface testing takes place when modules or sub-systems are integrated to create larger systems. Each module or sub-system has a defined interface, which is called by other program components. The objective of this testing is to detect faults which may have been introduced into the system because of interface errors or invalid assumptions about interfaces. There are different types of interface between program components and consequently different types of interface error can occur.

Parameter interfaces: these are interfaces where data or sometimes function references are passed from one component to another.

Shared memory interfaces: these are interfaces where block of memory is shared between sub-systems and retrieved from there by other sub-system.

Procedural interfaces: these are interfaces where one sub-system requests a service from another sub-system by passing a message to it. A return message includes the results of executing the service.

In this stage, the module-level interfaces and the system-level interfaces were tested thoroughly for errors.

Validation testing

Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can be reasonable expected by the users. After validation test has been conducted, one of two possible conditions exists:

- A derivation from specification is uncovered and a deficiency list is created.
- The function or performance characteristics confirm to specification and are accepted.

Output testing

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specific format. The outputs generated or displayed by the system under consideration are tested by asking the users about the format required by them.

The outputs produced by the newly developed system were shown to the users and suggestions regarding the screen formats were invited. Those suggestions that were found vital were taken into account.

User acceptance testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of development and making changes wherever required.

As this system was developed under constant discussion with the users of the system, the changes requested by them were made during the system development stage itself.

During the above phases of testing, Remote Network Monitoring revealed some of the errors were detected and corrected then and there. Testing cannot show the absence of the defect. Hence during the implementation of the commercial plan system undetected error, if any, may be detected and it forms the part of maintenance.

5.2. System Implementation

A crucial phase in the systems life cycle is the successful implementation of the new system design. Implementation is the stage of project when the theoretical design is turned into a working system. Implementation involves creating computer-compatible files, training the operating staff, and installing hardware, terminals and telecommunications network (where necessary) before the system is up and running. A critical factor in conversion is not disrupting the functioning of the organization.

In system implementation, user training is crucial for minimizing resistance to change and giving the new system a chance to prove its worth. The training aids includes user manuals, help screens, data dictionary, job aids etc.

There are three types of implementation:

- Implementation of a computer system to replace a manual system.
- Implementation of a new computer system to replace an existing one.
- Implementation of a modified application to replace an existing one, using the same computer.

The newly developed REMOTE NETWORK MONITORING has to be implemented successfully. Training aids for staffs have been provided in the form of help screens. Since organizations system undergoes continual change, the application will undoubtedly have to be maintained. Modifications and changes will be made to the user requirement in order to keep pace with the changing environment.

Software development is incomplete without any documentation. Documentation for the newly developed system is provided to satisfy the following needs.

- Protect the system when personnel are promoted, transferred or leave.
- Represents long-term money savings because it reduced the cost of training.
- Eases system maintenance by centralizing materials describing the system
- Provides a permanence reference on the system.

Documentation of the Remote Network Monitoring has been done at various levels as described below:

Management documentation

Management documentation has the least details but includes

1. An overview of the system.
2. System goals and objective.

3. Implementation Details.

Program documentation

The body of every program contains comments explaining the purpose of each module. The comments are included in procedural sections, functions also where complex logic is involved. They may assist maintenance programmers because they reduce the need to read the program line by line when making enhancements.

Refinements based on feedback

The feedback from the operators of the new system will be taken into account and analyzed with great attention and remedial measures will be taken as necessary. Solution will be found for valid & feasible suggestions. The reasons for discarding certain invalid or not-feasible suggestions, if any, will be presented to the user up to their satisfaction.

CONCLUSION

CONCLUSION

Remote Network Monitoring is successfully designed and developed at Kumaraguru College of Technology, Coimbatore. This project is developed using Microsoft Foundation Class Library & Win32 APIs under Visual C++ 6.0. This project was done keeping in mind the fact it should follow all the steps of software engineering process and covers the complete Software Development Life Cycle. The users of the software are provided with easy and friendly interface. The user interface provided by this project will be widely accepted by the users in general the software was tested thoroughly to ensure that it works effectively and efficiently. A complete documentation that is provided makes the changes and enhancements that are to be done very easy and provides the vitality of documentation. The preference of Visual C++ to any other GUI is also justified since any further enhancement of this project would mean that more of platform independent extensions that are fore seen with this project.

The application is tested with the user requirements and verified for validity. The software requirements have been met. Needed documents were generated and adequate documentation has been provided for maintenance and future enhancements.

SCOPE FOR FUTURE DEVELOPMENT

SCOPE FOR FUTURE DEVELOPMENT

Though this application meets the requirement specification, there is always some room for progress in the future. This project has left the scope of further enhancements wide open as it was developed in an environment that supports Intranet widely.

The software can be extended so that it allows the user to remotely control a PC over an office network or even over a dial-up connection.

Also compression techniques such as Huffman encoding, run length encoding, etc can be incorporated so that captured screen can be compressed and the data transmission through the network can be achieved efficiently.

BIBLIOGRAPHY

BIBLIOGRAPHY

Books

- Roger S. Pressman,
  "Software Engineering – A Practitioners Approach", McGraw Hill International Editions, Fourth Edition, 1990

- Elias M. Awad,
  "System Analysis and Design", Galgotia Publications,Second Edition, 1995

- Lee,
  "Introducing System Analysis and Design", Galgotia Book Source, 1980

- Kruglinski J. David, Shepherd George, Wingo Scott, "Programming Visual C++", Microsoft Press, Fifth Edition, 1998

- Herbert Schildt, "Network Programming for Windows", Microsoft Press, 2000

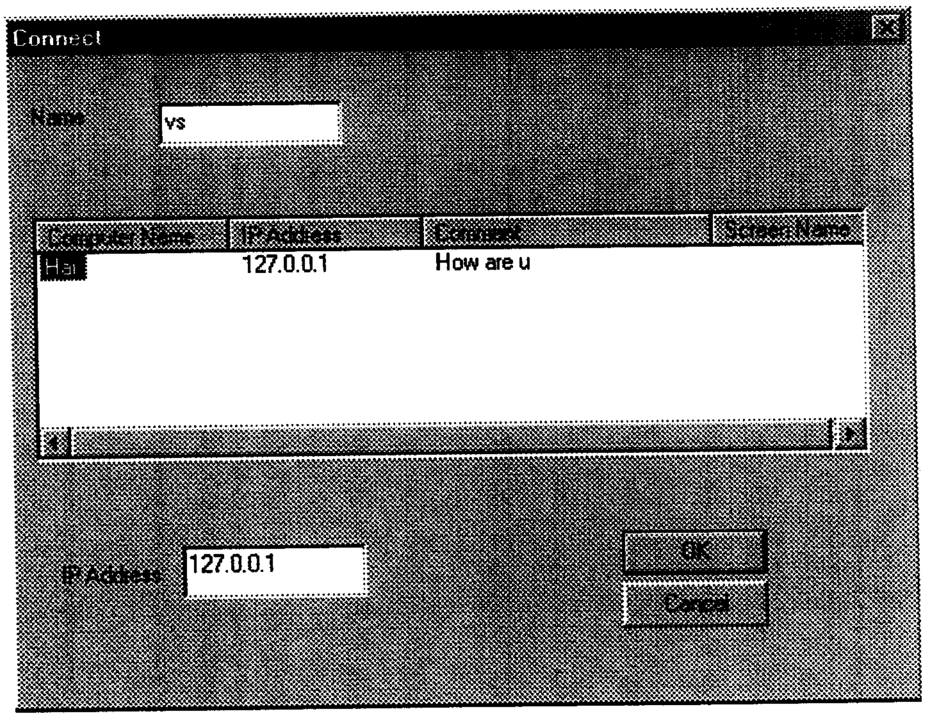
- Antony Jones and Jim Ojlund, "Network Programming for Windows", Microsoft Press, 2000
- Lewis Napper, "WinSock 2.0" IDG Books Worldwide, Inc, 2000

Web Sites

- www.developersdex.com
- www.askme.com
- www.w3schools.com
- www.microsoft.com/msdn
- www.apexsc.com
- www.rougewave.com

APPENDIX-I SAMPLE SCREEN

Screen #1: IP Listing Screen

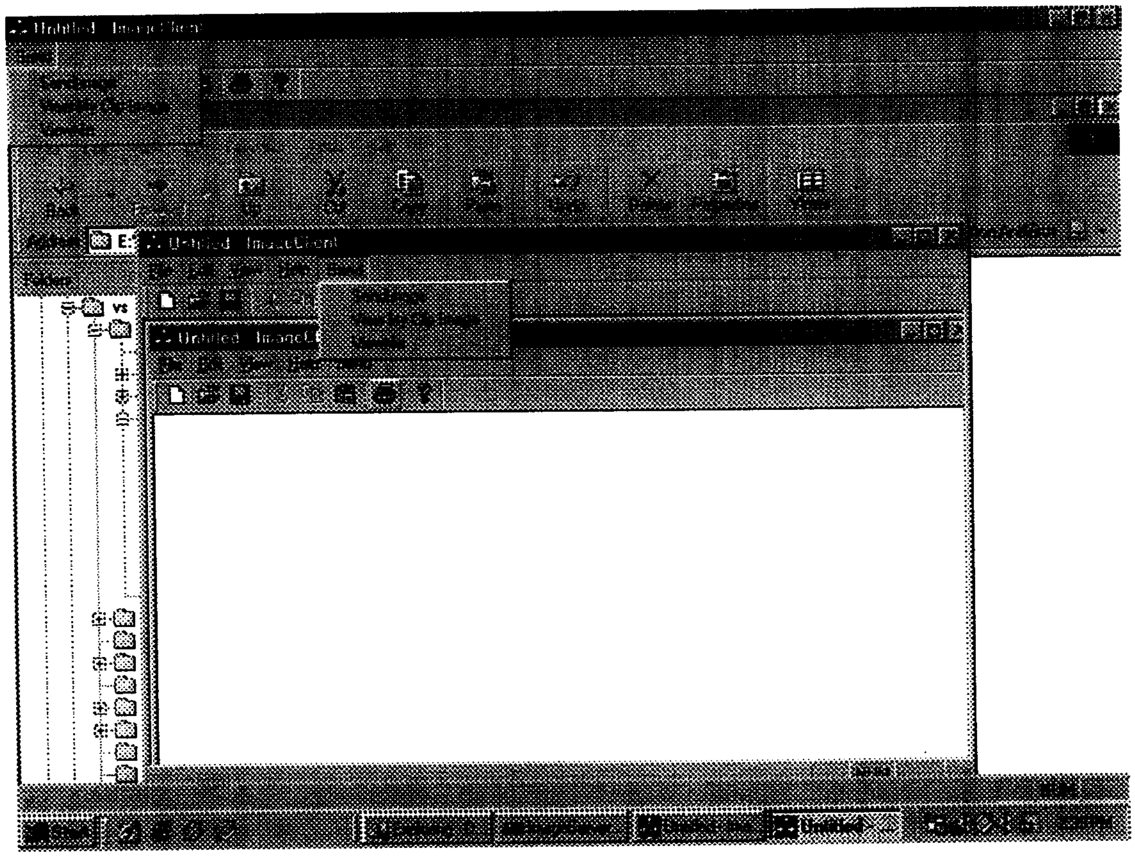

Connect

Name: vs

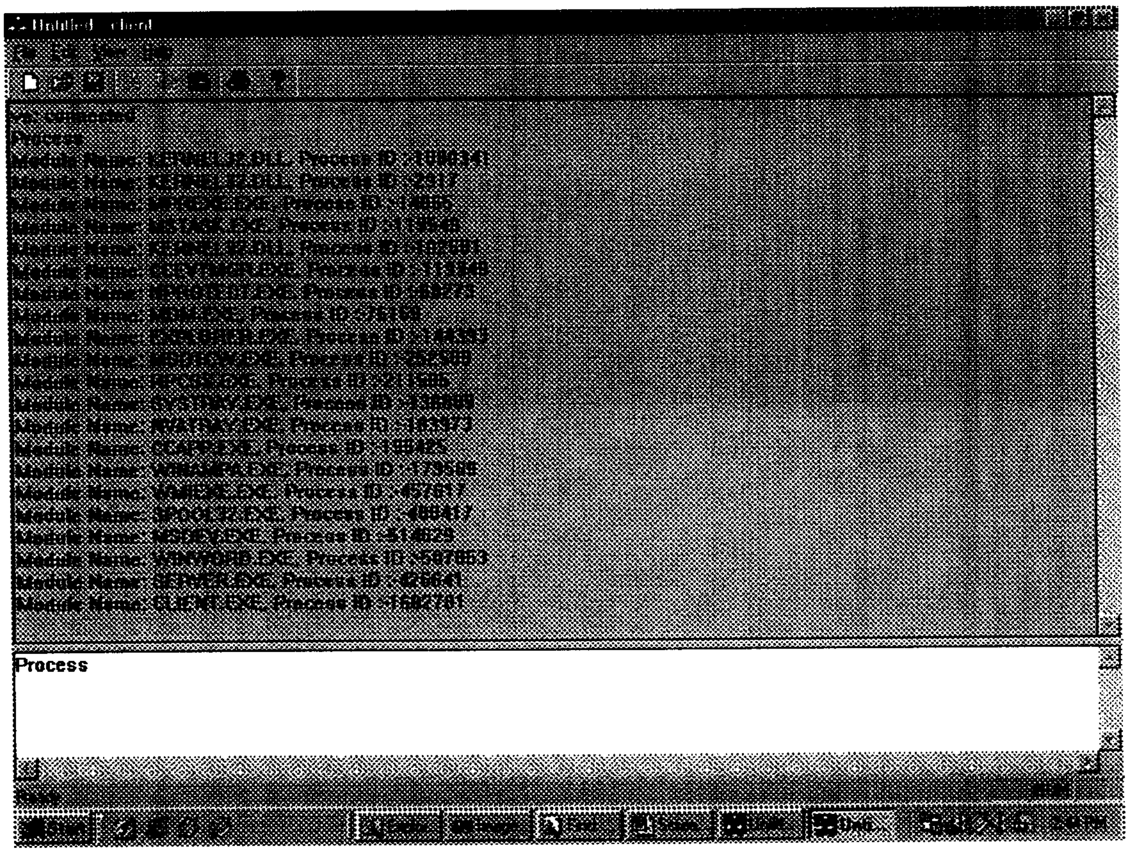
Computer Name	IP Address	Comment	Screen Name
Hai	127.0.0.1	How are u	

IP Address: 127.0.0.1

Screen #2: Capture Screen



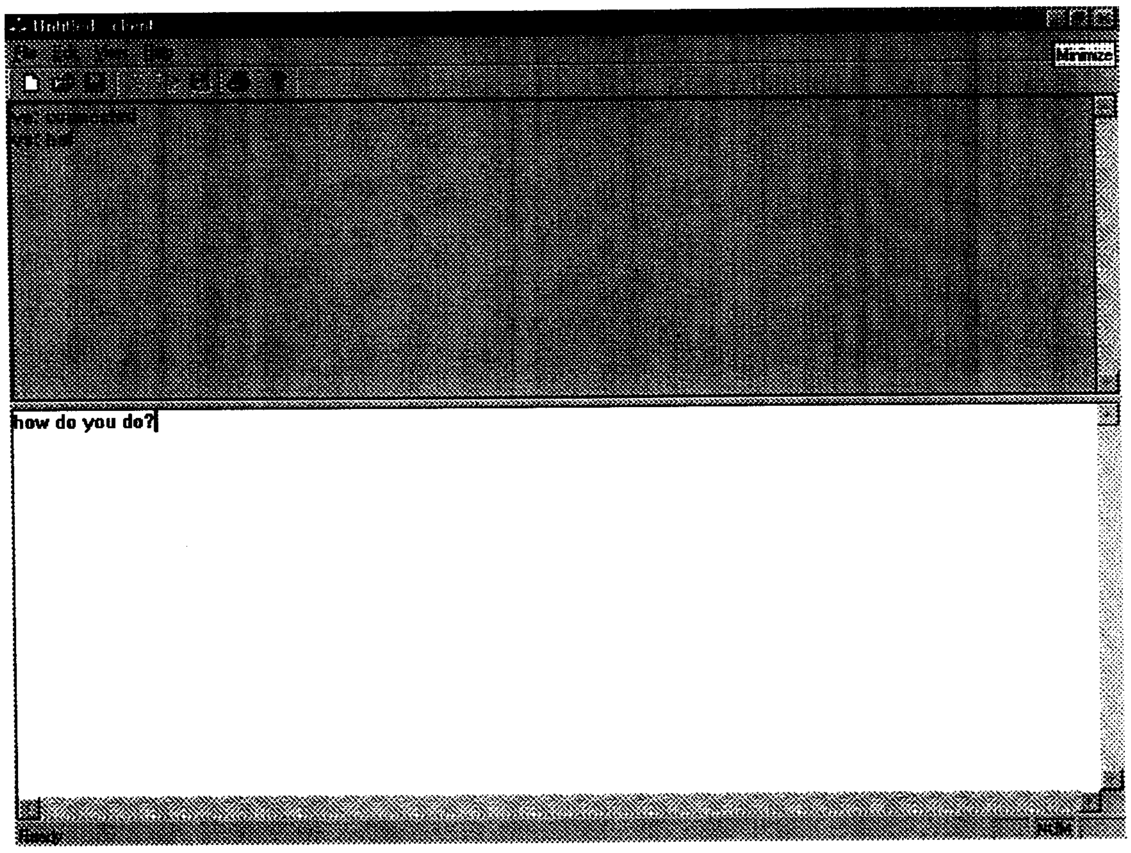
Screen #3: Process Capture



```
Connected
Process
Module Name: KERNEL32.DLL, Process ID :1090341
Module Name: KERNEL32.DLL, Process ID :2317
Module Name: MPREXE.EXE, Process ID :14695
Module Name: MSTASK.EXE, Process ID :119549
Module Name: KERNEL32.DLL, Process ID :102591
Module Name: CCEVTMGR.EXE, Process ID :113849
Module Name: NPROTECT.EXE, Process ID :158273
Module Name: MDM.EXE, Process ID :76159
Module Name: EXPLORER.EXE, Process ID :144393
Module Name: MSGMON.EXE, Process ID :252509
Module Name: RPCSS.EXE, Process ID :211505
Module Name: SYSTRAY.EXE, Process ID :138889
Module Name: AVATRAY.EXE, Process ID :163973
Module Name: CCAPP.EXE, Process ID :198425
Module Name: WINAMP.EXE, Process ID :179589
Module Name: VMIEXE.EXE, Process ID :457617
Module Name: SPOOL32.EXE, Process ID :488417
Module Name: MSDEV.EXE, Process ID :514629
Module Name: WINWORD.EXE, Process ID :587853
Module Name: SERVER.EXE, Process ID :426641
Module Name: CLIENT.EXE, Process ID :1582701
```

```
Process
```

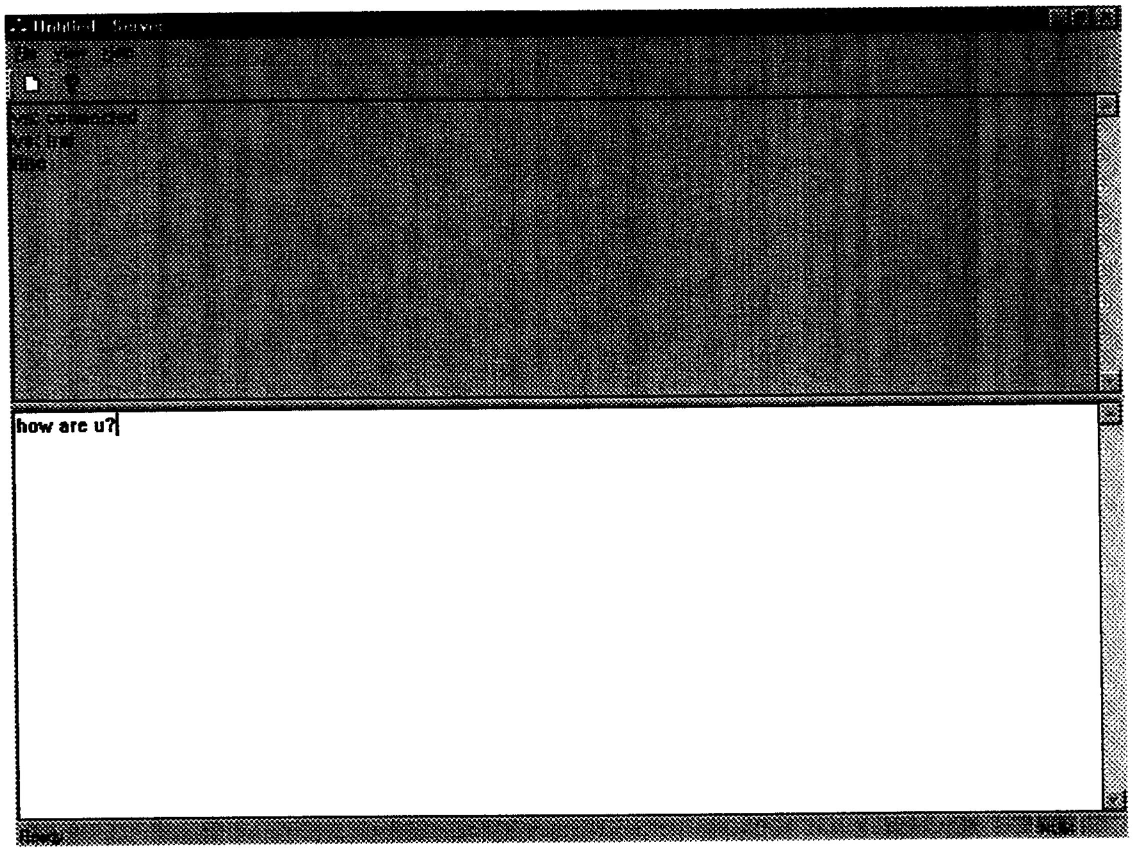
Screen #4: Client Chat Screen



[illegible]

how do you do?

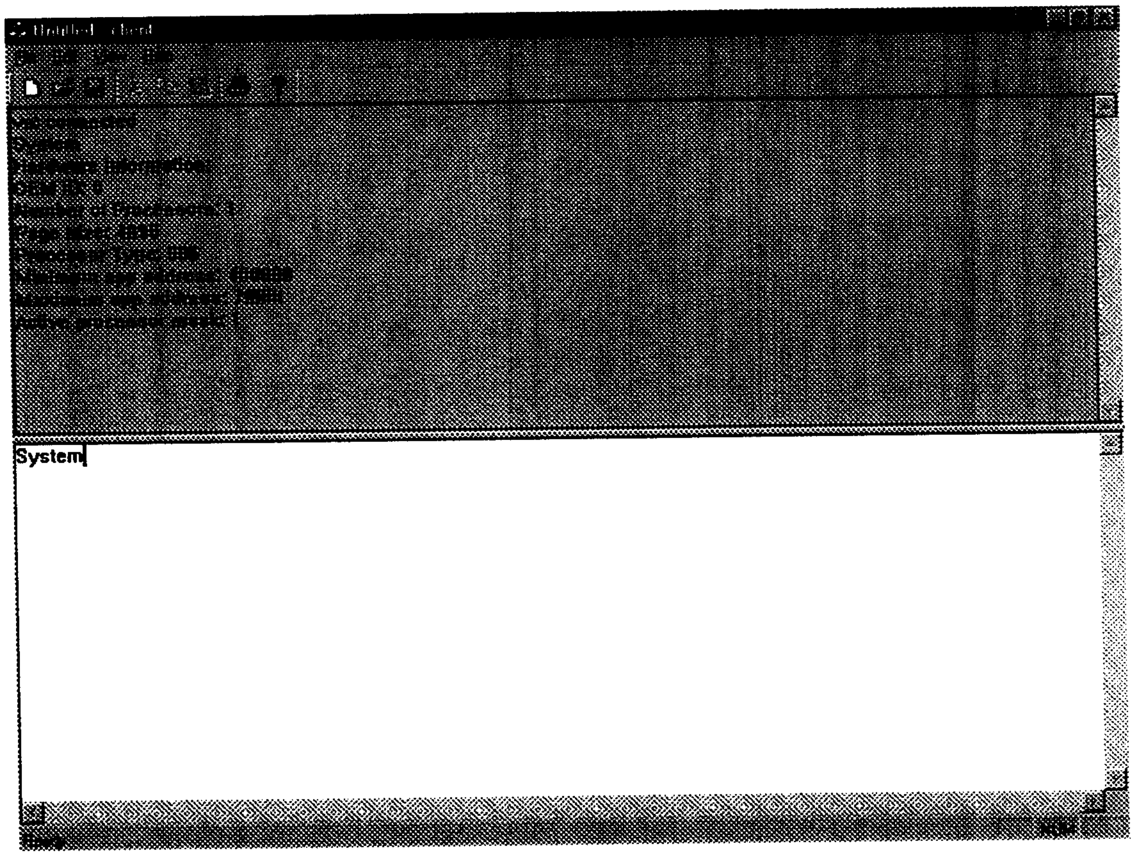
Screen #5: Server Chat Screen



[illegible] connected
[illegible]
[illegible]

how are u?

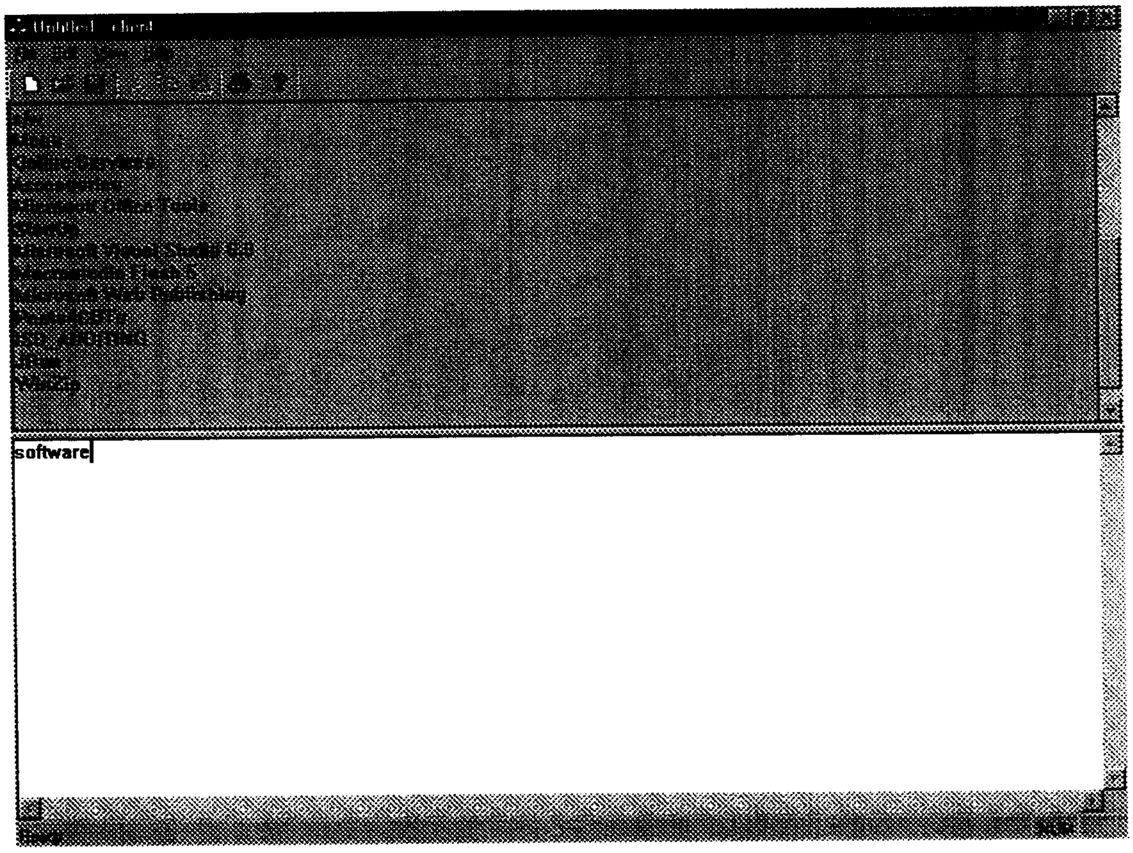
Screen #6: Hardware Information



```
Get connected
System
Hardware information:
OEM ID: 0
Number of Processors: 1
Page Size: 4096
Processor Type: 586
Minimum app address: 10000
Maximum app address: 7ffeffff
Active processor mask: 1
```

```
System
```

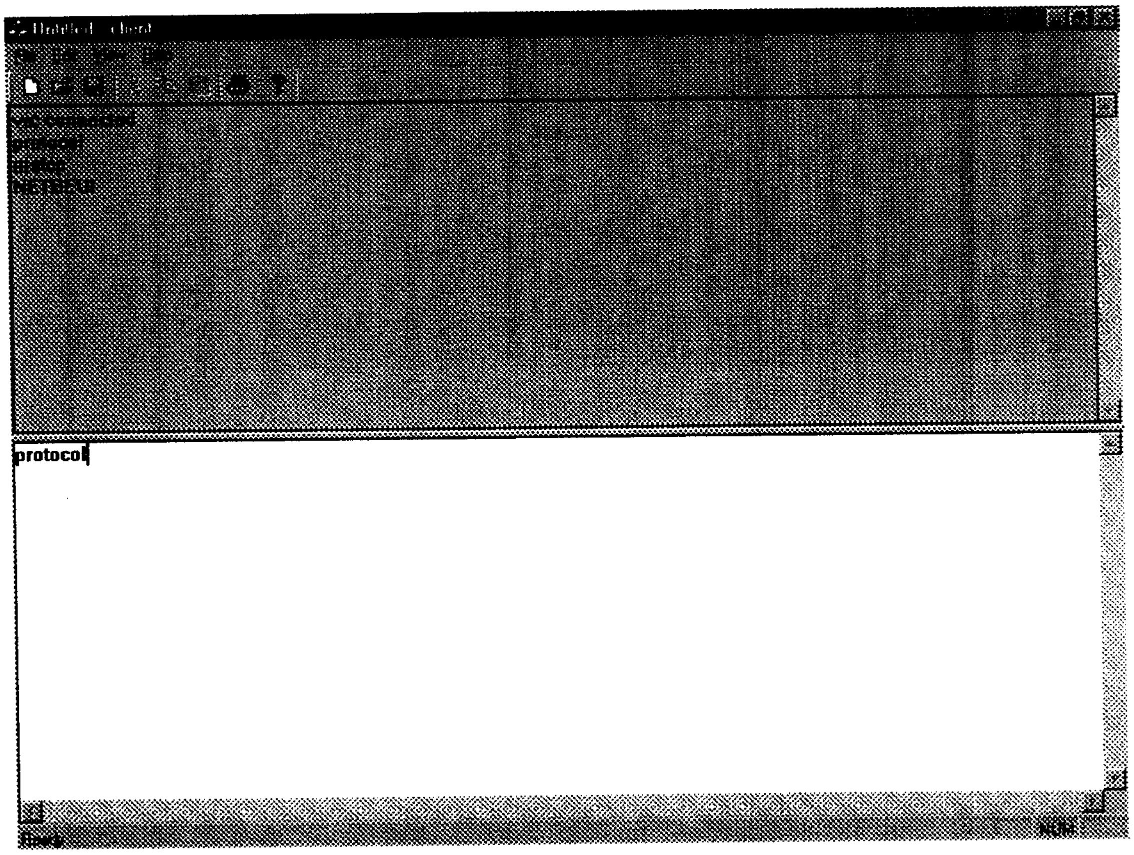
Screen #7: Software Information


Untitled - client

[illegible]
Media
Online Services
Accessories
Microsoft Office Tools
StartUp
Microsoft Visual Studio 6.0
Macromedia Flash 5
Microsoft Web Publishing
[illegible]
ESD_AUDITING
[illegible]
WinZip

software

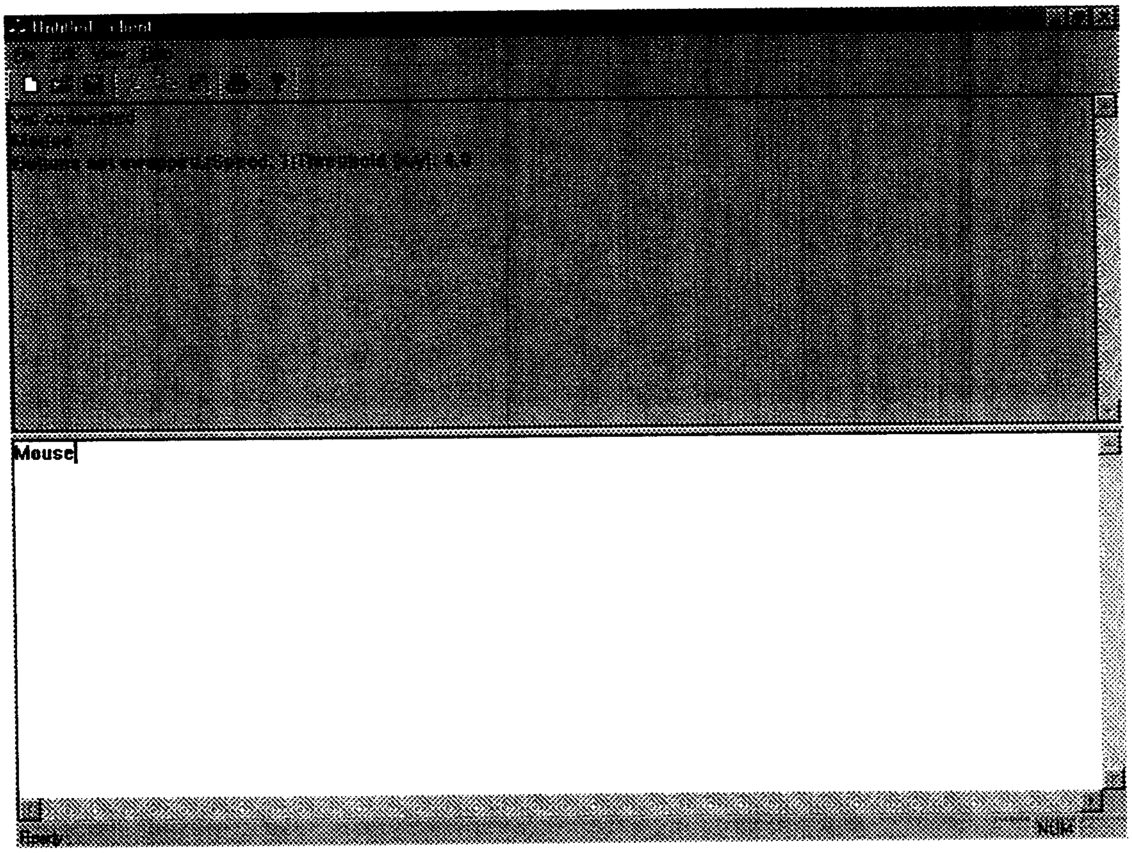
Screen #8: Protocol Information



```
vc connected
protocol
[illegible]
NETBEUI
```

```
protocol
```

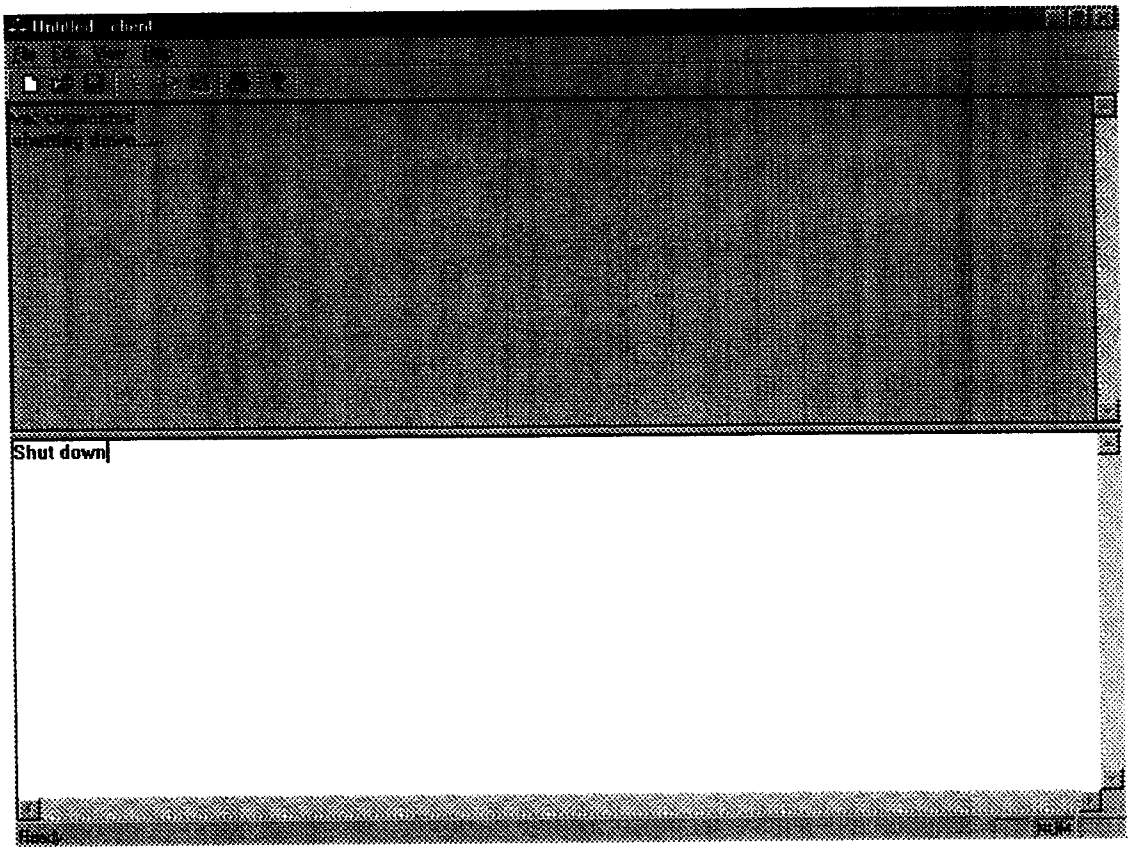
Screen #9: Mouse Information



```
[illegible]
Mouse
Buttons not swapped,Speed: 1,Threshold (x,y): 4,8
```

```
Mouse
```

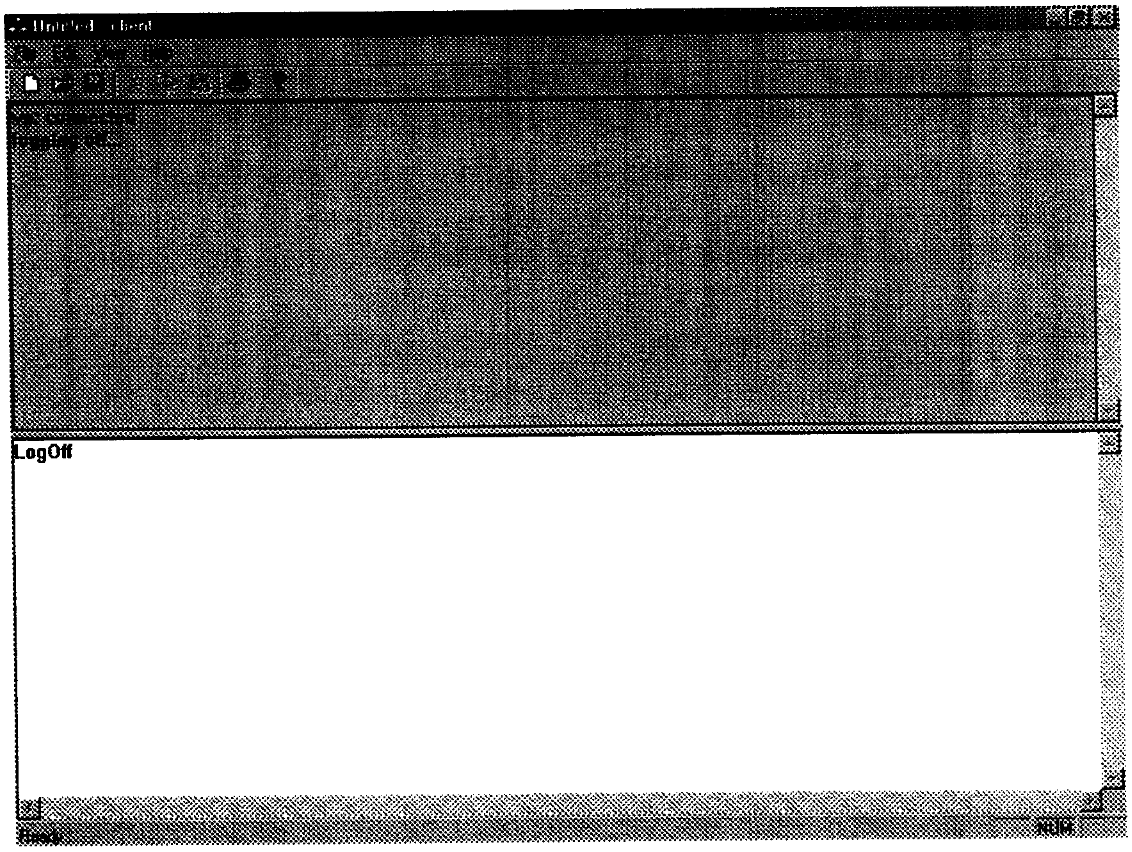
Screen #10: System Shut Down



```
[illegible]
shutting down....
```

```
Shut down
```

Screen #11: System Log Off



```
[illegible]
logging off...
```

```
LogOff
```

APPENDIX-II SAMPLE CODE

```
// clientDoc.cpp : implementation of the CClientDoc class
//

#include "stdafx.h"
#include "client.h"

#include "clientDoc.h"
#include "ConnectDlg.h"
#include "Msg.h"
#include "clientView.h"
#include "SendView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CClientDoc

IMPLEMENT_DYNCREATE(CClientDoc, CDocument)

BEGIN_MESSAGE_MAP(CClientDoc, CDocument)
	//{{AFX_MSG_MAP(CClientDoc)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CClientDoc construction/destruction

CClientDoc::CClientDoc()
{
	m_bAutoChat = FALSE;
	m_pSocket = NULL;
	m_pFile = NULL;
	m_pArchiveIn = NULL;
	m_pArchiveOut = NULL;
}

CClientDoc::~CClientDoc()
{
}
```

```
BOOL CClientDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;

                CConnectDlg Dialog;

                Dialog.m_strHandle=m_strHandle;
                Dialog.m_strServer=_T("");
                Dialog.m_nChannel=0;

                while(TRUE)
                {
                        if (IDOK != Dialog.DoModal())
                                return FALSE;

                        if (ConnectSocket(Dialog.m_strHandle, Dialog.m_strServer,
 Dialog.m_nChannel))
                                return TRUE;

        //              if (AfxMessageBox(IDS_CHANGEADDRESS,MB_YESNO) ==
 IDNO)
        //                      return FALSE;
                }


}



/////////////////////////////////////////////////////////////////////////////
// CClientDoc serialization

void CClientDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                for(POSITION pos=GetFirstViewPosition();pos!=NULL;)
                {
                        CView* pView = GetNextView(pos);
                        CClientView* pChatView =
DYNAMIC_DOWNCAST(CClientView, pView);

                        if (pChatView != NULL)
                                pChatView->SerializeRaw(ar);
                }
        }
        else
        {
                // TODO: add loading code here
        }
}

/////////////////////////////////////////////////////////////////////////////
// CClientDoc diagnostics

#ifdef _DEBUG
void CClientDoc::AssertValid() const
{
        CDocument::AssertValid();
}

void CClientDoc::Dump(CDumpContext& dc) const
{
        CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CClientDoc commands

BOOL CClientDoc::ConnectSocket(LPCTSTR lpszHandle, LPCTSTR lpszAddress,
UINT nPort)
{
                m_strHandle = lpszHandle;

        m_pSocket = new CChatSock(this);

        if (!m_pSocket->Create())
        {
                delete m_pSocket;
                m_pSocket = NULL;
                AfxMessageBox("IDS_CREATEFAILED");
                return FALSE;
        }

        while (!m_pSocket->Connect(lpszAddress, nPort + 700))
        {
                if (AfxMessageBox("IDS_RETRYCONNECT",MB_YESNO) == IDNO)
                {
                        delete m_pSocket;
                        m_pSocket = NULL;
                        return FALSE;
                }
        }

        m_pFile = new CSocketFile(m_pSocket);
        m_pArchiveIn = new CArchive(m_pFile,CArchive::load);
        m_pArchiveOut = new CArchive(m_pFile,CArchive::store);

        CString strTemp;
        if (strTemp.LoadString(IDS_CONNECT))
                SendMsg(strTemp);

        return TRUE;
}

void CClientDoc::SendMsg(CString &strText)
{


                if (m_pArchiveOut != NULL)
                {
                CMsg msg;



                if(!strcmp(strText,"Log Off"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"Shut Down"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"System"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"Mouse"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"Increase Mouse"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"User & Variables"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"version"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"protocol"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"s/w"))
                        msg.m_strText = strText;
                else if(!strcmp(strText,"Lock"))
                        msg.m_strText = strText;
                else
```

```
		msg.m_strText = m_strHandle + _T(": ") + strText;

		DisplayMsg(msg.m_strText);
//		TRY
//		{
			msg.Serialize(*m_pArchiveOut);
			m_pArchiveOut->Flush();
/*		}
		CATCH(CFileException, e)
		{
			m_bAutoChat = FALSE;
			m_pArchiveOut->Abort();
			delete m_pArchiveOut;
			m_pArchiveOut = NULL;

			CString strTemp;
			if (strTemp.LoadString(IDS_SERVERRESET))
				DisplayMsg(strTemp);
		}
		END_CATCH*/
	}
}

void CClientDoc::ProcessPendingRead()
{
	do
	{
		ReceiveMsg();
		if (m_pSocket == NULL)
			return;
	}
	while(!m_pArchiveIn->IsBufferEmpty());
}

void CClientDoc::ReceiveMsg()
{
	CMsg msg;

	TRY
	{
		msg.Serialize(*m_pArchiveIn);

		while(!msg.m_msgList.IsEmpty())
		{
			CString temp = msg.m_msgList.RemoveHead();
			DisplayMsg(temp);
```

```
				}

		}
		CATCH(CFileException, e)
		{
			m_bAutoChat = FALSE;
			msg.m_bClose = TRUE;
			m_pArchiveOut->Abort();

		/*	CString strTemp;
			if (strTemp.LoadString(IDS_SERVERRESET))
				DisplayMsg(strTemp);
			if (strTemp.LoadString(IDS_CONNECTIONCLOSED))
				DisplayMsg(strTemp);*/
		}
		END_CATCH

		if (msg.m_bClose)
		{
			delete m_pArchiveIn;
			m_pArchiveIn = NULL;
			delete m_pArchiveOut;
			m_pArchiveOut = NULL;
			delete m_pFile;
			m_pFile = NULL;
			delete m_pSocket;
			m_pSocket = NULL;
		}
}

void CClientDoc::DisplayMsg(LPCTSTR lpszText)
{
	for(POSITION pos=GetFirstViewPosition();pos!=NULL;)
	{
		CView* pView = GetNextView(pos);
		CClientView* pChatView = DYNAMIC_DOWNCAST(CClientView,
pView);

		if (pChatView != NULL)
			pChatView->Message(lpszText);
	}
}

void CClientDoc::Displaystrl()
{
	view->shutdown();
```

```
}
```

Server Dialog.cpp

```
// ServerDoc.cpp : implementation of the CServerDoc class
//

#include "stdafx.h"
#include "Server.h"

#include "ServerDoc.h"
#include "SrvDialog.h"
#include "Msg.h"
#include "ServerView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CServerDoc

IMPLEMENT_DYNCREATE(CServerDoc, CDocument)

BEGIN_MESSAGE_MAP(CServerDoc, CDocument)
	//{{AFX_MSG_MAP(CServerDoc)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CServerDoc construction/destruction

CServerDoc::CServerDoc()
{
	m_pSocket = NULL;
}

CServerDoc::~CServerDoc()
{
	delete m_pSocket;
}
```

```
BOOL CServerDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;

        CSrvDialog Dialog;

        if (Dialog.DoModal() == IDOK)
        {
                m_pSocket = new CListenSocket(this);
                if (m_pSocket->Create(Dialog.m_nPort+700))
                {
                        if (m_pSocket->Listen())
                                return TRUE;
                }
        }
        return FALSE;


}



/////////////////////////////////////////////////////////////////////////////
// CServerDoc serialization

void CServerDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                // TODO: add storing code here
        }
        else
        {
                // TODO: add loading code here
        }
}

/////////////////////////////////////////////////////////////////////////////
// CServerDoc diagnostics

#ifdef _DEBUG
void CServerDoc::AssertValid() const
{
        CDocument::AssertValid();
}

void CServerDoc::Dump(CDumpContext& dc) const
{
        CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CServerDoc commands

void CServerDoc::ProcessPendingAccept()
{
        CClientSocket* pSocket = new CClientSocket(this);

        if (m_pSocket->Accept(*pSocket))
        {
                pSocket->Init();
                m_connectionList.AddTail(pSocket);
        }
        else
                delete pSocket;
}

void CServerDoc::ProcessPendingRead(CClientSocket *pSocket)
{
        do
        {
                CMsg* pMsg = ReadMsg(pSocket);

                if (pMsg->m_bClose)
                {
        //                      CloseSocket(pSocket);
                        break;
                }
        }
        while (!pSocket->m_pArchiveIn->IsBufferEmpty());

//      UpdateClients();
}
```