Centralized Installation and Administration

PROJECT REPORT

Submitted in partial fulfillment of the requirement for
the award of the degree

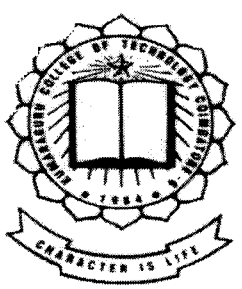
BACHELOR OF ENGINEERING - INFORMATION TECHNOLOGY
Bharathiar University, Coimbatore.

Submitted by:

Ashwin Baboo. R	**(9927S0046)**
Balan. V	**(9927S0047)**
Santhosh. M	**(9927S0072)**

Guided by:

Prof. Dr. S. Thangasamy B.E (Hons)., Ph.D.
(Head of Department, Computer Science and Engineering)



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE - 641006.

MARCH 2003

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE

This is to certify that the project report entitled

Centralized Installation and Administration

Is a bonafide record of work done by

Ashwin Baboo. R	**9927S0046**
Balan. V	**9927S0047**
Santhosh. M	**9927S0072**

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING
(INFORMATION TECHNOLOGY)

S. Thangasamy 19/3/'03
Prof. Dr. S.Thangasamy
Project Guide

S. Thangasamy 19/3/'03
Prof. Dr. S.Thangasamy
Head of the Department

Submitted for the University Examination held on 20.3.2003

Internal Examiner

External Examiner

SiliSys TECHNOLOGIES
POWERING THE INFO WAY
www.silisysindia.com

March 6th 2003

CERTIFICATE

This is to certify that

Mr. R. Ashwin Baboo
Mr. V. Balan
Mr. M. Santhosh

has worked as a Project Trainee and Completed the Project entitled

CENTRALISED INSTALLATION AND ADMINISTRATION

for the period of October 2002 to February 2003.

Authorised Signatory

11, Avarampalayam Road Opp. Sri Ramakrishna Hospital New Sidhapudur Coimbatore - 641 044
Ph : 212779, 210292, 213425 e-mail : info@silisysindia.com

ACKNOWLEDGEMENT

We are greatly indebted to our beloved Principal Dr. K.K. Padmanabhan B.Sc.(Engg.), M. Tech., Ph. D., who has been the backbone of all our deeds.

We would like to thank Prof. Dr.S.Thangasamy Ph. D., Head of the Department of Computer Science and Engineering for his consummate technical guidance, with constant encouragement and suggestions in carrying out this project successfully.

We earnestly express our gratitude to our course coordinator Mr.K.R.Baskaran B.E.,M.S. for lending a helping hand in this project .

We would be failing in our duty if we do not express our gratitude to Miss.D.Malathi and Mr.T.Vadivel Kumar of Silisys Technologies,CBE for their willingness to spare their precious time and help us to work towards our goal .

We also owe much to our friends and parents for their moral support and valuable help rendered to us.

ORGANIZATION PROFILE

COMPANY

Silisys Technologies, primarily an IT solution providing company, is located in Coimbatore, South India. The company provides all types of IT solutions right from client / server computing to web enabled solutions

SERVICES

Though we are basically solution providers, they believe more in serving our client to the best of their requirements. The company have a well-focused team, whose intentions are focused in providing the best possible service that the client requires.

The key behind the success of SiliSys Technologies lies in working along with clients rather than working for them.

E-SOLUTIONS

The Best way to establish business contacts over the net through e-commerce.

- Portal
- Vortal
- Cortal
- C2C, B2C, B2B.

ADVANCED E-SOLUTIONS

We provide advanced e-solutions to meet your business needs. Our services include

- Intranet
- Extranet
- WAP
- Data Warehousing.

CLIENT-SERVER PACKAGES

As a customer based solution maker, we from SiliSys technologies give you the best.

- Readymade template designs.
- Customized client-server packages.
- System side packages.

SYNOPSIS

Networking is the booming field in Information Technology. Local area network is one of the networks, which undergoes the client server computing concepts. This concept helps in sharing information. The systems that are present in the network need some necessary software to be installed in them. To install software to the clients in the network, the administrator has to install them individually. The objective of the project is to eliminate the complexity of installation of software to client systems in local area network. It consists of installation and administration processes.

The installation part helps in installing the software to the clients from the server. This is performed simultaneously. This saves the time and reduces the effort. Handling the whole network from the server makes the process easier for administrator by providing the following features,

- *Message Communication*
- *Locking and releasing the clients*
- *Logoff, Shutdown, rebooting the clients*
- *Details about user name of client access*

CONTENTS

		Page No.
Acknowledgement		**i**
Organization profile		**ii**
Synopsis		**iv**
Chapter 1 Introduction		
1.1	**Current Status of the problem taken up**	**1**
1.2	**Relevance and Importance**	**2**
Chapter 2 Literature Survey		
2.1	**Imaging the Software**	**4**
2.2	**Sharing the Software**	**5**
Chapter 3 Requirements		**6**
Chapter 4 Proposed approach to the system		
4.1	**Installation**	**7**
4.2	**Administration**	**8**
Chapter 5 System Design		
5.1	**Input Design**	**12**
5.2	**Output Design**	**16**
Chapter 6 Implementation Details		
6.1	**Development Environment**	**17**
6.2	**System Implementation**	**20**
6.3	**Server Program**	**29**
6.4	**Client Program**	**53**

Chapter 7 Experimental Results

7.1	Unit Testing	58
7.2	Integration Testing	59
7.3	Condition Testing	60
7.4	Experimental Results	60

Conclusion 63

Future Outlook 64

Bibliography 65

Annexure

Appendix – A	Screen Shots	66
Appendix – B	Sample Socket Program	71

1. INTRODUCTION

1.1 CURRENT STATUS OF THE PROBLEM TAKEN UP

The EDP department of corporate bodies generally implement Client/Server architecture. When they need to install high-end software like Visual studio, Flash, etc over a large number of systems, they have to manually reach out to every individual system to accomplish the task of installation. In case of restarting the system after installation the administrator has to again reach out to every individual system. This is obviously a time consuming and a hectic process to the administrator.

1.1.1 INSTALLATION

- The software is installed in every system in a network individually.
- Even in a shared mode, the user has to move to each and every system and run the setup.

1.1.2 ADMINISTRATION

- The facilities such as shutdown, logoff, message passing and restart are available only with latest versions of operating systems like

windows 2000, Novell, etc, wherein a particular system cannot be restricted from the above operations.

- These options are not available within Windows 98 and Windows NT, which are most commonly used.

1.2 RELEVANCE AND IMPORTANCE

To ease out an administrator's job of handling the above situation in a versatile manner the idea of centralizing the installation of software was a thought and thus developed. This view was also extended towards administration, which allows the administrator to capably handle the clients. Thus the design was sketched out into two modules, namely software installation and network administration.

Being an excellent system side development tool VC++ was used in order to deliver improved support for reuse, registry handling, tool interoperability and resource management.

Installing software over multiple clients in a network simultaneously from the server end was captured in the software installation module. It

facilitates the administrator to select/deselect-connected clients and track the installation process.

The administrator is additionally provided with feature to manage clients in a network remotely as proposed in the network administration module.

The administration module provides with the following featured as listed below

Locking	-	locks the selected clients.
Unlocking	-	unlocks the selected clients.
Shut down	-	shuts down the selected clients.
Log off	-	logs off selected clients.
Restart	-	restarts selected clients.
Message passing	-	between clients and the server.
User details	-	system name, login name

2. LITERATURE SURVEY

Currently the installation of software to the clients in a network is done individually. When the network size is large, then individual installation becomes a tedious task. This form of installation requires more time and more effort from the administrator. There are some techniques that are used to install software in the clients. They are

- Imaging the Software
- Sharing the Software

2.1 IMAGING

Imaging means copying the content of one hard disk to another. This form of installation will copy both operating system and the software to the client system. This can be done only for a single system at a time. It takes more time when the clients are large in number.

After installation, the I.P Address and the computer name of the system should be updated. The disadvantage of this method is that a particular software alone cannot be installed.

2.2 SHARING

This method is useful in making any software sharable in the network. Actually the software is not being installed in the clients but the clients can share it via the network. The client can login to a specified name for using that software.

After the sharable software is installed in the server, the directories that are needed by the client is made accessible. For example when the client needs Turbo-C, then the TC directory is made sharable in the server so that it can be shared in the network.

When the Novell Server is installed, the necessary directories are created in the 'sys' partition. When some directories in the server are to be shared, a partition is created and the administrator gives access rights to the users. In this technique, the client registry need not be updated.

A main drawback of imaging software is that it consumes more time, as imaging should be done individually in all clients and also all software cannot be shared. But through 'Centralized Installation and Administration' we can install any software to the clients from the server simultaneously. Hence it consumes less time.

3. REQUIREMENTS

3.1 SOFTWARE REQUIREMENTS

Language	:	VISUAL C++ 6.0
Operating System	:	Windows 9x , NT

3.2 HARDWARE REQUIREMENTS

Processor	:	Pentium II
CPU Speed	:	600 MHZ
RAM	:	64 MB
CD ROM	:	52X
Hard disk	:	4 GB
Keyboard	:	101 keys.

3.3 ADDITIONAL REQUIREMENT

LAN Networking.

4. PROPOSED APPROACH TO THE SYSTEM

This chapter describes in detail about the Installation and Administration. Installation is performed to the client systems from the server. Administration deals with controlling of the client machine from the server. These are explained as follows.

4.1 INSTALLATION

4.1.1 Server Installation

The system in which the server software is present acts as the server. This system can also be a network server. Any software has to be installed in the server before it is installed in the clients.

Before installing the application software, all the information regarding the server system is gathered. The information includes existing files, folders, keys and values of Registry.

These details are gathered and stored in the secondary storage (e.g. files). So they would not be lost during the rebooting of the system. This helps in storing data permanently. The space that is required to install the respective software is calculated. After gathering all the information, the necessary software is installed. At this instance the tracking operation

commences. Tracking of all these changes results in obtaining all the new files, new keys and values of the Registry.

4.1.2 Client Installation

After the software is installed in the server system, the group of clients to whom the software has to be installed is selected by means of IP Addresses. All the clients who are present in the range of IP Addresses are searched and displayed. From this list, the necessary clients can be selected and proceeded with installation. The spaces in the client systems are verified so that the software can be installed successfully. The new files, folders, new keys and values of Registry are sent to client systems from the server. Hence, the clients receive all the attributes of the software and the software is installed successfully.

4.2 ADMINISTRATION

This module is mainly applied to control the client machines from the server machine. The set of client machines can be selected and controlled. This module provides the following facilities:

Getting the Currently Connected Client Machine Name

At the time of initializing the Administration part, the IP settings are displayed. It consists of the starting field and ending field of the IP Address. The IP settings can be changed by the Administrator. A signal is sent to every system using Socket Programming for searching the client machines. In the client machines the system name is got using GetHostName() function. The server receives the system name from the client and displayed. In case, if the system is switched off, the signal gets timed out.

Sending Information To Clients

This helps in sending different messages by the Administrator to different clients. The set of clients that are obtained within the range of IP Addresses are listed out. The necessary clients can be selected and required information or message can be passed to them. For eg; A message can be sent to the clients at the time of installation to make them aware about the installation process.

Reboot the Client Machine

Some software needs to be rebooted in order to respond after installation due to registry update. The Administrator can handle the rebooting of client systems if necessary. For example, to restrict the access of any user, the client can be rebooted.

Log Off The Client Machine

The clients can be identified by means of system names or user login name that are available in the list that is present in the server. This helps the Administrator in identifying the list of clients who have logged in. The Administrator can log off a particular client or group of clients.

Shut Down The Client Machine

The Administrator can see the list of Clients who have logged in. The Administrator can shutdown a particular client or group of clients. This also helps in shutting down the systems that are not shutdown by the users.

Locking The Client Machine

Locking a client machine helps in blocking the user from performing any type of operations. The Key Board, Mouse and other types of input devices are blocked for the list of clients selected. This helps to avoid disturbances during the installation of software from the server to the clients.

Un-Locking The Client Machine

Un-Locking is the process of releasing the lock. The locked clients can be released by selecting the clients from the list and Un-Locking them. The selection of releasing the lock is optional.

User details

User details displays the user name and the time at which the user has logged in. These details are shown for the selected Systems. The user details helps in distinguishing different clients that are connected to the network.

5. SYSTEM DESIGN

5.1 INPUT DESIGN

To enter the various data required for the system, 5 input screens are designed for installation and administration. The various controls such as list box, edit box, command buttons and a status bar are used for designing the screen by which the user can easily follow. The input to the system was designed such that the required information can be collected and manipulated accordingly by the server and the clients as per requirements.

The goal of the input design is to make the data entry easier, logical and free from errors. The application is being developed in a user-friendly manner. The forms are designed in such a way that during processing the cursor is placed in the position where the data must be entered. The user is also provided with an option of selecting an appropriate input from the list values.

5.1.1 SOFTWARE INSTALLATION

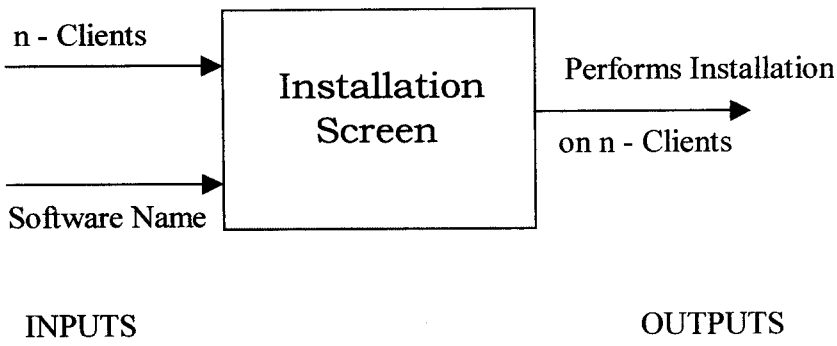


Figure 5.1 - Installation

The screen for the software installation modules requests the user to provide with the information as shown in the figure 5.1 to achieve the target of installation on multiple clients simultaneously.

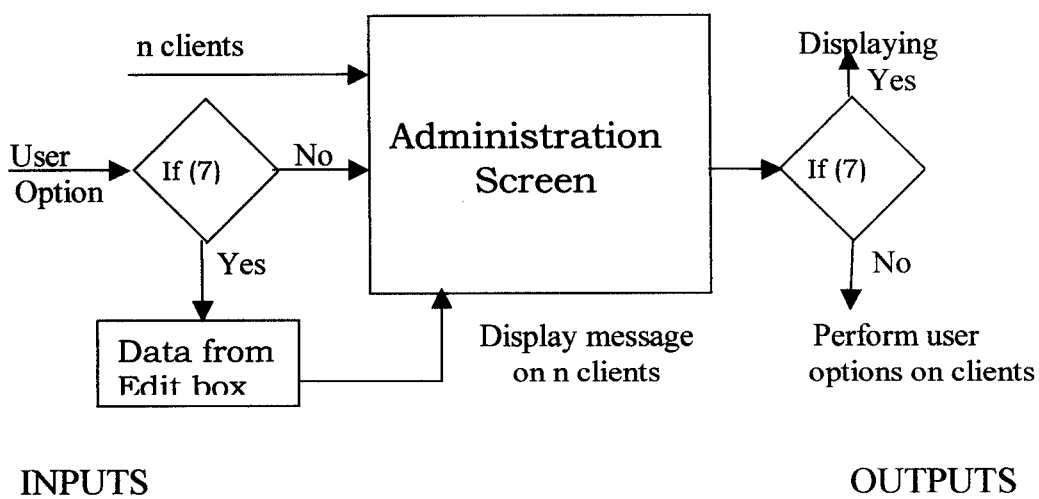
A list box shows the list of system names of connected clients. The items of the list box are dynamically added or removed as and when a client gets connected or disconnected

When user clicks the add button the selected system names will be displayed in the list box as a list of selected clients. In case the user decides to revoke the clients that are selected, he can perform it by clicking the remove button after selecting the particular client from the list box, which contains a list of selected clients.

A text box is provided for the user to enter the software name when the software has to be installed using a CD in the server machine. When the user want to install a software that has been previously installed through this software, he clicks the 'Reinstallation' button, then the list of software that has been installed will be displayed in a list box. On double clicking he can select the software. The process of installation is commenced on the click of Install button.

To indicate the user about the progress of installation process, a status bar is provided. The newly added details of the server's registry as a result of the installation process are reflected in the clients' registry finally.

5.1.2 NETWORK ADMINISTRATION



User options – (1) user details (2) locking (3) Unlocking (4) logoff (5) restart (6) shutdown (7) message passing

Figure 5.2 Administration

The administrative module deals with listing out the clients connected to the server and the various operations that can be performed on them. At this point the input from the user would be the selection of n clients from the list and the required operations as shown in figure 5.2. To provide the user with a standard way to make selections carry out commands and perform input and output tasks, appropriate controls have been incorporated.

A list box shows the list of system names of connected clients. The items of the list box are dynamically added or removed as and when a client gets connected or disconnected respectively, list box is suitably designed to support multiple selections so that the user can choose more than one client at a time.

Various administration operations are provided in the form of buttons for the user click one operations at a time. The options include *user details, locking, unlocking, restart, shutdown, logoff and message passing*. Once the button is clicked the operation is carried out on the selected clients. A text box is provided where in the user can type the message to be sent to the clients chosen.

5.2 OUTPUT DESIGN:

The user is constantly provided with messages during every event that is invoked. Outputs generated by the system during the process to the end users are also promptly displayed.

At the server side, with respect to the administration module, the end user is provided by the confirmation message about the completion of the operation selected by him.

With respect to the installation module, to indicate the user about the progress of installation process, a status bar is provided. On the termination of the file transfer process an indication is given to the user that the file transfer is completed. On the completion of the installation process, the server displays a message box stating the completion of installation in all selected client machines.

6 IMPLEMENTATION DETAILS

6.1 DEVELOPMENT ENVIRONMENT

MICROSOFT VISUAL C++6.0

VISUAL C++6.0 includes the Microsoft developer studio Integrated Development Environment (IDE). This environment is the centerpiece of most any interaction to create visual c++ projects, including source file creation, resource editing, compiling, linking, debugging and many other useful features that will make the development tasks much simpler. Visual C++ is the first and foremost C++ complier, made up of many components, which paves the way for efficient programming.

PROJECT WORKSPACE:

Working with developer studio is working with project workspaces. These workspace represent a particular set of projects, which can represent anything from a single application, to a function library, or to an entire suite of applications. Each workspace may include any number

of different projects that we want to group into a single workspace so that we can work closely with each separate project at the same time.

The project workspace (.dsw) file is responsible for maintaining all of the information that defines our workspace and the projects that we have included in it

REGISTRY:

The Registry is a system-defined database that applications and system components use to store and retrieve configuration data.

The Registry is a hierarchically organized store of information. Each entry in the tree –like information structure is called a *key*. A key may contain any number of sub keys, it can also contain data entries called *values*. In this form, the registry stores information about the system, its configuration, hardware devices, and software applications. A registry key is identified by its name. Key names consist of printable ASCII characters except the backslash (\), space and wildcard (* or ?) characters. The use of key names that begin with a period (.) is reserved. Key names are not case sensitive.

A value in the registry is identified by its name. Value names consist of the same characters as key names. The values itself can be a string, binary data or a 32-bit unsigned value.

PREDIFINED REGISTRY KEYS:

The registry contains several predefined keys.

The HKEY_LOCAL_MACHINE key contains entries that describe the computer and its configuration. This includes information about the processor system board, memory and installed hardware and software.

The HKEY_CLASSES_ROOT key is the root key for information relating to document types and OLE types. This type is a subordinates key to HKEY_LOCAL_MACHINE. Information that is stored here is used by shell applications such as the program manager, file manager, or the explorer and by OLE applications.

The HKEY_USERS key serves as the root key for the default user preferences setting as well as individual user preferences.

The HKEY_CLASSES_USER key is the root key for information relating to the preferences of the current (logged in) user.

Under Windows 95, there are two additional predefined keys. The HKEY_CURRNET_CONFIG key contains information about the current system configuration settings. This key is equivalent to a sub key (such as 0001) of the key HKEY_LOCAL_MACHINE\Config.

The HKEY_DYN_DATA key provides access to dynamic status information, such as information about plug and play devices.

6.2 SYSTEM IMPLEMENTATION

This describes about the various installation and administration procedures. The Socket programming briefs the Client/Server communication, which is used in all the modules. The installation procedure describes the steps that are followed for installing software in the client systems. Administration procedure describes about the various functions that are provided to the administrator to control the clients.

6.2.1 INTRODUCTION TO SOCKET PROGRAM

CLIENT SERVER COMMUNICATION

Socket programming is a basic communication interface between the client and the server machine. The main protocols used in Visual C++ are

- Transfer Control protocol/Internet protocol (TCP/IP)
- User Data gram Protocol (UDP)

1) UDP

The main difference between the TCP/IP protocol and the UDP protocol is that UDP protocol is a connection less protocol i.e. whenever any data transfer takes place, the acknowledgement will not be sent back to the source.

2) TCP/IP

TCP\IP is a four-layer connection-oriented protocol. Whenever any data transfer takes place, the acknowledgement will be sent back to the source, which is necessary to identify the problem occurring in the network (loss of data, collision of data). TCP/IP protocol can be implemented by the following procedure.

TCP/IP SOCKET PROGRAMMING

SERVER	*CLIENT*
	INITIALIZE FOR COMMUNICATION
* **WSAStartup**	* **WSAStartup**
* **Socket**	* **Socket**
	* **Bind**
	* **Listen (waits for server)**
	COMMUNICATION BEGINS
* **Connect**	* **Accept**
(CONNECTED)	**(CONNECTED)**
* **Send (request)**	* **Receive**
* **Receive**	* **Send (Respond)**
	COMMUNICATION ENDS
* **CloseSocket**	* **CloseSocket**
* **WSACleanUp**	* **WSACleanUp**

Socket program is useful in making a connection between the server and client. The various API's that are used in socket program are explained in detail.

WSA Startup

This API is used in both Client and Server machine. Before any communication starts, this API should be called, to initiates the WINDOW'S DLL file called WINSOCK.DLL [3]. By specifying the port number and IP address a communication is possible between server and client. Any low level or bit level data transfer will not be a problem here.

All the data transfer, synchronization, traffic control, acknowledgement are taken care in this DLL file. At the time of calling this API, the version of the windows socket should be mentioned which is used to synchronize the communication. Otherwise the data format may vary at the time of communication.

Socket

The Socket API is used to create the socket in a machine. Socket is an area through which any communication takes place. Data packet is transferred from one machine to another, along with the destination IP address and the port number. Whenever any data packet comes from one

machine to another, IP address and the port number of the packet is compared with the existing socket's details[3].

If the details match, then the data will be transferred to the client. This concept is used in Socket communication. For communicating between the client and the server, sockets are created in both machines. At the time of Socket creation the following details should be mentioned. Those are

- Address family.
- Type of data transfer.
- Protocol used.

Initialization of Socket variables:

- Address family should be 'AF_INET'
- Type of data transfer should be 'SOCK_STREAM' (It provides reliable stream of data transfer).
- Protocol is TCP/IP.

Bind

This API is used to register a particular Socket in the machine. As the client program waits for server's request, the client program should

bind the Socket. The Server need not bind the socket .The server program will be directly connected with the created socket [3].

As soon as the client binds the socket, the socket will be ready to send or receive the data. At the time of binding two main parameters should be specified.

- The created Socket present in the Client machine is mentioned here.
- IP Address of the local machine
- Port number at which the Socket waits. (This software has the constant port number in both server and the client. i.e. 5100)
- Address family for data communication.

Listen

This is applicable to the client machine. As the client program always waits for server's request, it should be in the listening state [3]. At the time of calling the LISTEN API the following parameters should be mentioned.

- The socket of the client machine.
- Number of connections possible.

Connect

Whenever the server wants to send some requests to the client program, it should be connected with the client. As the client program is always in the LISTEN state, the server simply calls and connects with the client directly [3]. For connecting, the server initiates the CONNECT API. The following details should be mentioned for this API.

- The Socket of the server machine.
- The destination (client) IP address.
- The destination (client) PORT address.

Accept

Whenever any CONNECT request comes from the server, ACCEPT API will be initiated in the client machine. So automatically the client program comes out of the LISTEN state and calls this API. The API should specify the local socket. The return value of the API will be the SERVER SOCKET. The client socket is useful for data transfer in the client side. After ACCEPT API is called, communication is established between the client and the server, until the socket is closed [3].

Send

After the connection is made in between the client and the server, any information can be sent in both directions. For sending the data from the client machine, should be provided the following details

- The SERVER SOCKET (received from ACCEPT API)
- The information in the form of 'string'

For sending the data from the server machine, should be provided the following details

- The SERVER SOCKET created in Server machine.
- The information in the form of 'string'

Receive

After the connection is made in between the Client and the Server, any information can be received in both directions. The control will lies with this API until any information is sent to the Client machine. For receiving the data from the Server machine, should be provided the following details

- The SERVER SOCKET (received from ACCEPT API)
- The information in the form of 'string'

For receiving the data from the Client machine, should be provided the following details

- The SERVER SOCKET created in Server machine.
- The information in the form of 'string'

Close Socket

This API is used to close the socket in the current machine, so that communication does not take place [4].

WSACleanUp

Any type of internal (packet format) communication is done by WINDOWS operating System by allocating the resources. After all the process are completed, the resources should be released. WSACleanUp API does this. It is needed in both Client and Server machine [4].

The above activities are explained in Figure 6.1. Which shows the actual communication between the Server and the Client.

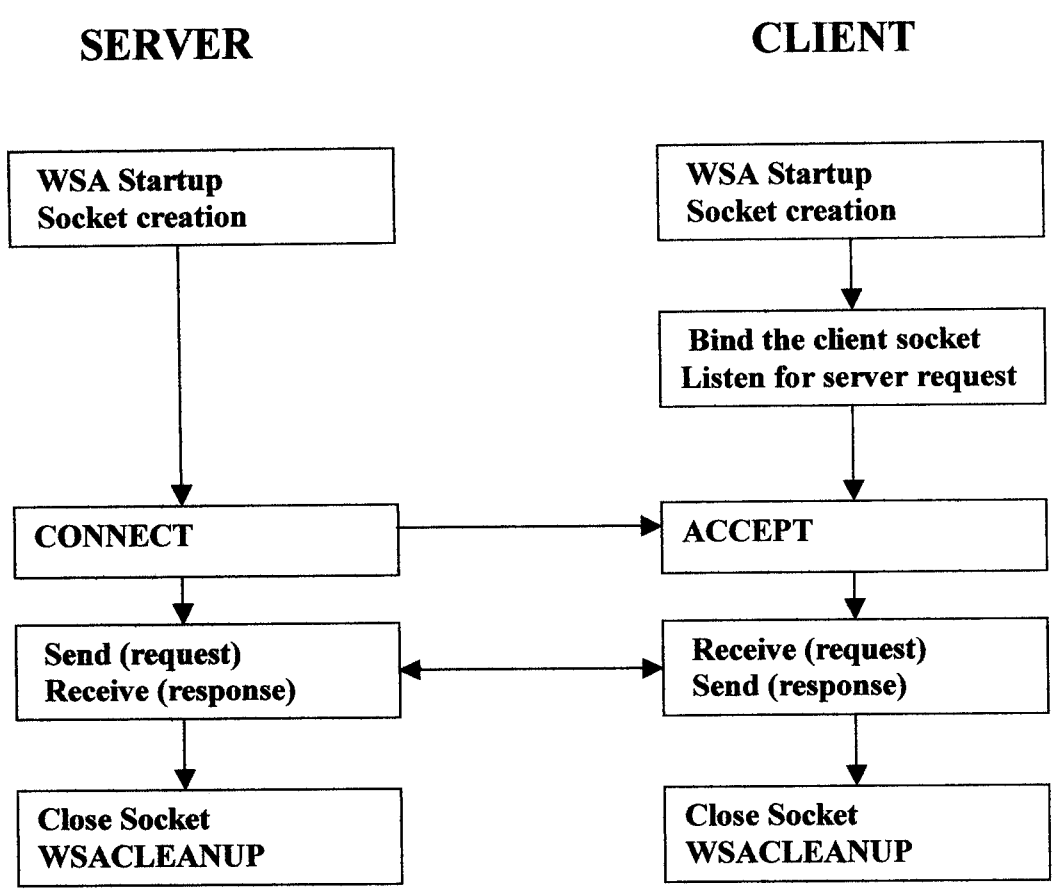


Figure 6.1 Client Server Communication Flowchart

6.3 SERVER PROGRAM

The Server Program can be in the Server Machine or in any Client Machine. Any One of the Client Machine in which the Server program runs acts as a Server for this Software and other Machines act a Clients. Server Program contains three main operations

(i) Server Installation

(ii) Client Installation

(iii) Administration

6.3.1 SERVER INSTALLATION

Any Software installation will change the composition of the machine. Software installations are done by extracting CAB files. The CAB file is in the compressed format. At the time of installation, the CAB file will be extracted and the necessary files will be installed.

The following modifications can be done at the installation time:

- New files can be created and data can be written.
- New files can be copied from the Software to the local machine.
- New directories can be created.
- Registry can be updated.
- Existing files can be modified. (Example, AUTOEXEC.BAT)

The main idea behind installation is tracking the Server machine before and after installation and identifying new changes. These changes are made in all the Clients. Server installation part is used to track the local machine before the user installs the Software.

Steps in Server Installation

- Taking Registry backup before Installation.
- Taking the backup of the Autoexec.bat file.

- Storing the paths of all the files in the Server machine.
- Storing the paths of all the directories in the Server machine .
- Storing the current free space in the hard disk before installation.
- Storing the detail of the Software installation. (Dumping or normal Installation)

Registry is a main part in the Windows environment. The registry is used to store the details about the Software. Some of the details are

- Software location (short cut).
- Start up (Runs automatically).
- Passwords.
- Time of expiration.
- Software details.

Whenever any Software is installed, the above details (keys and values) can be stored in the Registry. The same changes should be made in the Client machine also. The new changes should be found by comparing the old Registries keys and values. The old Registry's backup is taken by 'Exporting' the Registry to a Text file [1].

Autoexec file is a Batch file, which helps to execute the DOS commands at the booting time. Some Software uses this file to set the Software path. The installation may change the Autoexec file. For finding the modifications, the old Autoexec file's backup is stored in a temporary text file.

The main change in the Software installation is due to the addition of new files and directories. The paths of all the directories and files existing should be stored be before installation. After installation of the software the same details should be collected. These two details are compared to find the new files and directories [2].

The paths of the files and directories are found by traversing recursively from the Root directory. After installation, some Software may reboot the System to get the System updated. At that time, these details may be lost as they are stored in the array variables (main memory). For overcoming this problem, the details are stored in the Secondary storage. Which can be retrieved after installation.

The Depth First Search is used to identify the existing file names and Directories [5]. Figure 6.2 shows the above Searching Technique.

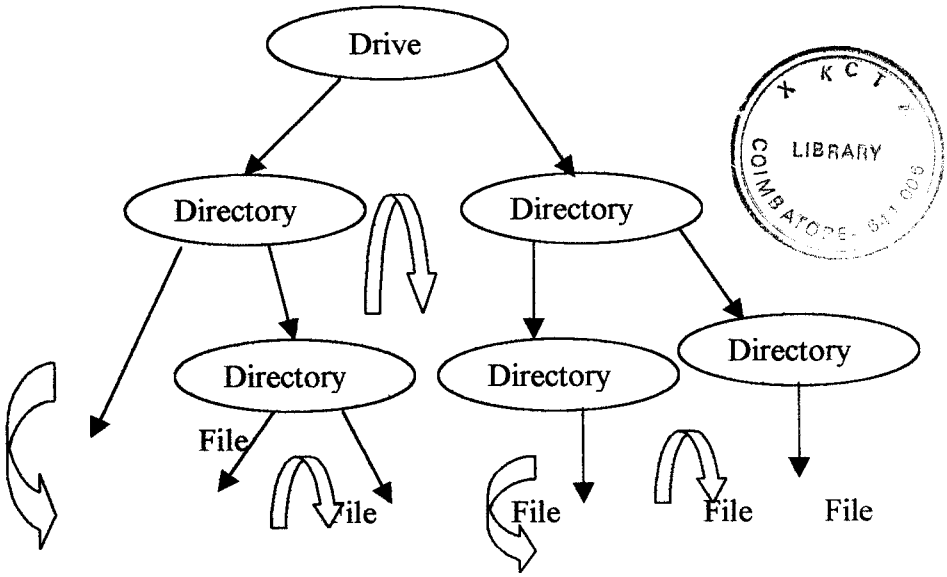


Figure 6.2 Finding files and directories using depth first search

PSEUDO CODE for searching the files and directories

FIND FIRST FILE (Give Root directory)

Recursive function (DIRECTORY PATH)

Get the NEXT FILE

WHILE (NEXT FILE is not NULL)

IF NEXT File's attribute is a directory

STORE DIRECTORY PATH

CALL Recursive function(DIRECTORY PATH)

ELSE IF NEXT File's attribute is a file

STORE FILE PATH

END of WHILE

END of Function.

The installation may affect the existing files by appending some details to them. Every file has its own attributes like File name, path name, Creation time, Last access time, space, Modified time, etc. If an existing file is modified at the installation time, the Modified Time can identify it, as the Current time is stored before installation.

The idea is to check the Current Time with every file's Modified time. If the Modified Time is greater than the Current time, that file is assumed to be a modified file. The Current Time may be lost when System gets rebooted after the Software Installation. So this time is stored in the Secondary memory (File).

Before installing any Software in the Clients, the memory space required for the Software should be known. The space in the Client machine should be checked to know whether it has enough space to install the whole Software. This is done by finding the difference between the 'free space before Installation' and 'free space after Installation'. The free space is measured in the Server machine.

The Software installation can be done in two ways.

- INSTALL
- DUMP

Install

Installation of Software is done in the WINDOWS platform by performing some changes in the local machine. These changes make the Software to run or work properly. They are

- New files can be created and data can be written into them.
- New files can be copied from Installation Software to the local machine.
- New directories can be created.
- Registry can be updated.
- Existing files can be modified. (Example, AUTOEXEC.BAT)

Some files and directories can be stored only in a directory (WINDOWS, Program files) in the WINDOWS platform. The paths of the newly created files and directories will exist in all the Client machines(c:\windows).

Dump

Dumping is placing of some directories or files in the local machine. For example, when Jdk (JAVA DEVELOPMENT KIT) Software is in the Compact Disk, it can be copied to one of the local directory. Dumping can be used if any single file has to be updated or transferred to all Clients.

Installation places the files and directories in some particular directories, which exist in the WINDOWS platform. But Dumping can be done in any Directory that may not exist in the WINDOWS platform by default.

If the Software is dumped, the files and folders can be transferred to the Client as usual. Before transferring any file or directory, the program checks whether the particular path exists in the Client machine. If the path does not exist in the Client machine, it creates a new path and then transfers the file.

6.3.2 CLIENT INSTALLATION

After the Software is installed in the Server machine, the same Software should be installed in the selected Clients. The Server machine's current details are stored in the SERVER INSTALLATION part. Similar type of information should be collected during Client installation. Then the both information are compared so that the modifications on the Server machine at the installation time can be found. These modifications are done on the Client machines. They are

- New files can be added in the Server machine.
- New directories can be added in the Server machine.
- New keys and the values can be added to existing Registry.
- Existing files can be updated.

The main aim in this part is to find modification in the Server machine and transfer to the Client machine.

THE STEPS IN THIS MODULE:

- Checking space in the Client machine before installation.
- Finding the modifications in the Registry.
- Modifying the Client registry keys and values that are found recently.

- Finding the existing files and directories in the Server machine after the installation.
- Retrieving the details of files and folders before Installation.
- Finding new files and folders that are created at the Installation time.
- Transferring the files and folders to the Client machine.
- Transferring the files that are modified at Installation time.

Searching for Clients

The first step in the Client Installation is to identify the connected Clients. The signals are sent and received between the Server and the Client. If data transfer takes place, then the Client machine names will be displayed to the user. Otherwise the Client is assumed that it is connected with the Server. The Server gets the IP address of the Client machines from the user for installation.

If a user wants to know the connected Clients, the Server machine should send a request to all the Client machines one by one. If the Client machine is not connected in the local network, the Client will not receive the request. After few seconds, the socket program returns

"WSASTARTUP - ERROR" which means that the Client machine has not initialized the WSASTARTUP. It also means that the Client is not getting power supply [4]. So the user can know that the Client is not connected in the network.

If the Client program is not installed in the Client machine but the Server requests the Client for connection; the Server receives an error named 'WSATIMEOUT' which means that there is no such Client program within the socket.

If the Client program is running in the Client machine, the Client responds to the Server, by sending the Client name to the Server machine.

It happens at the time of searching the Clients. So the Server program will accept variety of responses. The Server displays the Client names in the first list box.

The user is allowed to choose the Clients. The selected Clients will be displayed in the second LIST box. The Installation is done for only those Clients that are in the second list box.

Checking free space before Client Installation

The next activity is to check the free space in the Server machine after Server installation. The free space of the Server machine (before Installation) which is stored in the Secondary storage.

(Filename=SDK_SPACE.TXT)

(Path=C:\WINDOWS\CIAA\SDK_SPACE.TXT)

The existing free space is retrieved after Installation. These two spaces are compared so that the memory occupied by the installed Software is found. The details of free space are sent to all the connected Clients. The Client receives that detail and compares it with the local free space in the Client machine. If the free space is not enough in the Client Machine, the Client program sends the message to the Server. The Server reports to the user that the particular machine does not have free space to install the Software. This Client name is removed automatically from the second LIST box.

Registry update in Client Machine

The registry of the Server machine is exported to a temporary file. The file has the full registry details. The keys, sub keys and their key values will be available in that file. This file should be transferred to the Selected Clients. The Client machine will be asked to update the existing registry. The updated information is taken from the transferred file and new keys and key values are imported to the Client Registry

Identifying the new Files and Directories

The main part of this module is to find the new files and directories that are added at the Installation time. The 'SOFTWARE NAME' is given by the user. As this project does all the works in the C:\windows\CIAA directory, it creates a new directory with the Software name. With in this directory, the Server program creates two files.

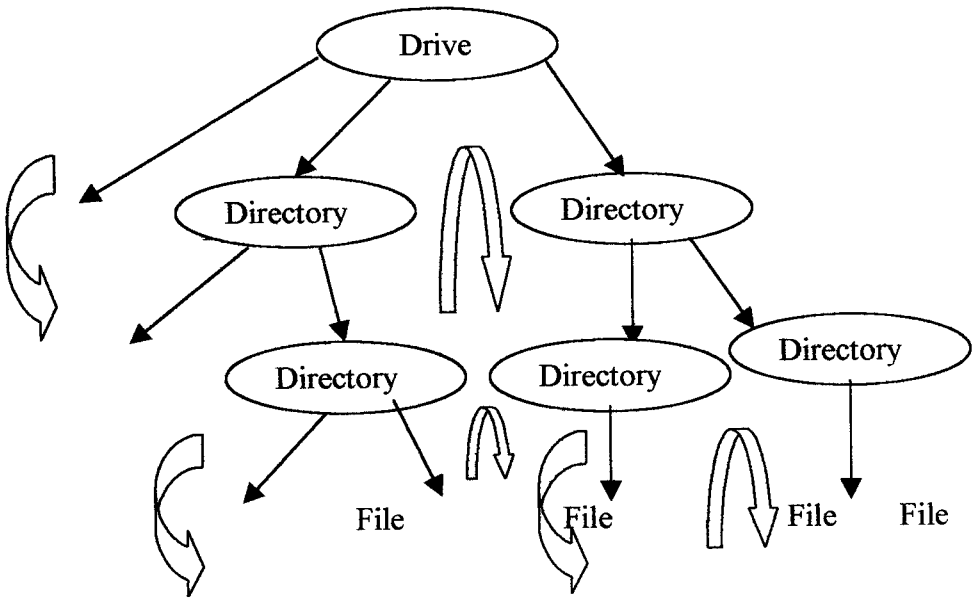


Figure 6.3

Finding files and directories after installation using Depth First Search

One file stores the existing files and the other file stores the directories which are added at the Installation time. These details are useful in the uninstallation part. For searching the files and directories "DEPTH FIRST SEARCH " is used [5]. Figure 6.3 shows the process of finding files and Directories.

PSEUDO CODE -for searching the files and directories

FIND FIRST FILE (Give Root directory)

Recursive function (DIRECTORY PATH)

Get the NEXT FILE

WHILE (NEXT FILE is not NULL)

IF NEXT File's attribute is a directory

STORE DIRECTORY PATH

CALL Recursive function(DIRECTORY PATH)

ELSE IF NEXT File's attribute is a file

STORE FILE PATH

END of WHILE

END of Function.

The files, directories and their paths (that exist in the Server machine) can be retrieved back after the Server installation. The file paths and directory paths are stored in two arrays(Named as New arrays).

The file paths and directory paths are retrieved back from the files which are stored before the Server installation. These details are stored in two arrays (Named as old array). So the new files and directories can be found by comparing these old and new arrays.

Finding new Directories

Two arrays are created to store the paths of the directories before installation and after installation. For identifying new directories the following algorithm is used.

ALGORITHM

OLD_ARRAY_DIR [] => contains the paths of directories before installation.

OLD_DIRS =>contains number of directories before installation.

NEW_ARRAY_DIR [] => contains the paths of directories after installation.

NEW_DIRS =>contains number of directories after installation.

1. [Initialize]

FILE_FOUND=false

2. [Search for new directories appended at Installation time]

FOR i=0 to NEW_DIRS

FOR j=0 to OLD_DIRS

IF OLD_ARRAY_DIR [j] = NEW_ARRAY_DIR [i]

THEN FILE_FOUND= true

BREAK from Inner for loop

IF FILE_FOUND= false

STORE the file path

Analysis of Algorithm

Consider, N = Number of directories before installation

M = Number of directories after installation

Order of this algorithm will be N.M

As soon as the directories are found, they are created in the Client machine. The Client machines are mentioned in the Second LIST box.

Also the details are stored in a file within the directory which is created recently with the 'SOFTWARE NAME'.

Finding new Files after Installation

Two arrays are created to store the paths of the files before installation and after installation. For identifying the new files the following algorithm is used.

ALGORITHM

OLD_ARRAY_FILE [] =>contains the paths of files before installation.

OLD_FILES =>contains the number of files before installation.

NEW_ARRAY_FILE [] =>contains the paths of files after installation.

NEW_FILES =>contains the number of files after installation.

ADDED_FILES =>contains number of new files added during installation.

1. [Initialize]

FILE_FOUND=false

2. [Search for new files appended at Installation time]

FOR i=0 to NEW_FILES

START = i - ADDED_FILES

IF START < 0

THEN START =0

FOR j=START to i

IF OLD_ARRAY_FILE [j] = NEW_ARRAY_FILE [i]

```
                THEN FILE_FOUND= true
                    BREAK from Inner for loop
            IF FILE_FOUND= false
            STORE the file path
            ELSE
                FOR j=0 to OLD_FILES
            IF OLD_ARRAY_FILE [j] = NEW_ARRAY_FILE [i]
                    THEN FILE_FOUND= true
                    BREAK from Inner for loop
                IF FILE_FOUND= false
                STORE the file path
```

Analysis of Algorithm

This algorithm is efficient in searching the files.

Consider, N = Number of files before installation

M = Number of files after installation

P = Number of new files that are added newly (M-N)

Order of this algorithm will be M.(P)

Finding the Modified Files after Installation

Depth first search is used in finding the modified files after installation. Some files in the Window's platform will get changed at the installation time. The file attribute 'LAST MODIFIED TIME' helps to find the modified files.

The time before the Software installation is stored in the Secondary storage. (FILE) This time is retrieved (named as TIME1). By recursively navigating in the directories, each file's Modified Time is compared with TIME1. If any file is modified at the installation time, it can be found. The file should be transferred to the Clients at the same time.

6.3.3 ADMINISTRATION

This is used to control the Client machines from the Server machine. This module provides the following facilities:

- Getting the names of currently connected Client machines.
- Sending information to Clients.
- Reboot the Client machine.
- Logoff the Client machine.
- Shut down the Client machine.
- Locking the Client machine.
- Un-locking the Client machine.
- User details (Log on User Name, System Name).

Getting the currently connected Client machine name

The first step in the Client Installation is to identify the connected Clients. The signals are sent and received between the Server and the

Client. If data transfer takes place, then the Client machine names will be displayed to the user. Otherwise the Client is assumed that it is connected with the Server. The Server gets the IP address of the Client machines from the user for installation.

If a user wants to know the connected Clients, the Server machine should send a request to all the Client machines one by one. If the Client machine is not connected in the local network, the Client will not receive the request. After few seconds, the socket program returns "WSASTARTUP - ERROR" which means that the Client machine has not initialized the WSASTARTUP. It also means that the Client is not getting power supply [4]. So the user can know that the Client is not connected in the network.

If the Client program is not installed in the Client machine but the Server requests the Client for connection; the Server receives an error named 'WSATIMEOUT' which means that there is no such Client program within the socket.

If the Client program is running in the Client machine, the Client responds to the Server, by sending the Client name to the Server machine.

It happens at the time of searching the Clients. So the Server program will accept variety of responses. The Server displays the Client names in the first list box.

The user is allowed to choose the Clients. The selected Clients will be displayed in the second LIST box. The Installation is done for only those Clients that are in the second list box.

Sending information to Clients

This helps in sending different messages to different Clients by the Administrator. The set of Clients who are all obtained by searching the network within the range of I.P. Addresses are listed out. The necessary Clients can be selected and required information or message can be passed to them. For example, the Clients are made aware of the installation process by sending messages.

The selected Client machines and their IP addresses are stored in the Server program. Request is made to the specified Clients using the IP Address. As soon as the Client receives the request from the Server, it gets the message and displays it in a Message Box which appears on the top of the Client's screen.

The message transferring is used for

- Sending the messages about installation
- Locking the Clients
- Informing any information to Clients.
- Getting the Client attention.

Reboot the Client machine

Some Software needs to be rebooted in order to respond after installation, this is due to registry update. After installation the Administrator can handle the rebooting of Client Systems if necessary. For example, to restrict the access of any user, the Client can be rebooted.

This part is used, at the time of installing any Software to the Clients. Some Software will reboot the machine after installation to update the

System. After installing the Software in the Client the machine will not get updated.

Log off the Client machine

The Clients can be identified by means of System names or User Login name that are available in the list that is present in the Server. This helps the Administrator in identifying the list of Clients who have logged in. The Administrator can log off a particular Client or group of Clients. If the Administrator finds that a Client has entered illegally (Unauthorized person enters), the Client can be logged off by choosing that particular Client machine.

Shut Down the Client machine

The Administrator can see the list of Clients who have logged in. The Administrator can shutdown a particular Client or group of Clients. This also helps in shutting down the systems that are not shutdown by the Users.

Locking the Client machine

Locking a Client machine helps in blocking the user from performing any type of operations. The Key Board, Mouse and other types of inputs are blocked for the list of Clients selected [4].

The Desktop Window is captured and locked. This helps mainly during installation of Software from the Server to a set of Clients; they are locked in order to avoid disturbances. Selection of Clients is done from the list and locking operation is performed.

This is useful for restricting the Clients from their access. Whenever the Client's attention is needed, locking the Clients may do it. In any particular period also the Clients can be locked.

Un-locking the Client machine

Un-Locking is the process of releasing the lock. The Clients that are locked can be released by selecting the Clients from the list and Un-Locking them. The selection of releasing the lock is also optional, a list of Clients can be selected and just Un-locked. The controls of all input devices are regained [4].

User details

User details displays the Logged in name and the time at which he User has entered. The details are shown for the set of Systems selected in the list. Whenever the Administrator wants to know the Client details, the Server program will send request to the Client program. The Client program retrieves the Current user name from the registry. The Client program has stored the Current time, as soon as the Client login. The user name and the login time are combined and sent back to the Server .

6.4 CLIENT PROGRAM

The Client Program should run in each Client Machine of the network where the Software installation or any administration should take place. The user who is working in the Client machine cannot view the Client program. The main advantage is that the users are not affected by this program execution. This helps in parallel execution of the user's work and the Client program execution.

The Client program execution is hidden in the TASK MANAGER. This is used to avoid the user from removing the Client program from the

TASK MANAGER. The Client program services each request of the Server. The various services provided by the Client program are given below.

User name and Login time

As soon as the Client program starts running, it retrieves the current user name of the Client machine name from the registry. It also gets the current System time using an API. When the Server program requests for user details, the Client program sends the user name and login to the Server [1].

Log off

When the Server program sends a request to log off the Client System, the Client program accepts the request and uses an API to log off the Client System.

Reboot

When the Server program sends a request to Reboot the Client System, the Client program accepts the request and uses an API to Reboot the Client System.

Shutdown

When the Server program sends a request to Shut down the Client System, the Client program accepts the request and uses an API to Shut down the Client System [6].

Lock

When the Server program sends a request to Lock the Client System, the Client program accepts the request and Locks the Client System using an API [4].

Un Lock

When the Server program sends a request to Un-Lock the Client System, the Client program accepts the request and Un-Lock the Client System using an API.

Autoexec Backup

When the Server program sends a request to the Client System for taking a backup of the autoexec.bat file before installation, the Client program takes a backup of autoexec.bat file and stores it in another file.

Space checking

After The Server sends the information to the Client specifying the amount of space required to install the Software. The Client program checks whether the required space is available. If the space for the

Software installation is available then it sends "1" as the data else the Client sends "0" as the data.

Autoexec update

After installation of any Software in the Server machine the Server program checks whether there are any modifications in the autoexec.bat file. If there is any modification the Server writes the modified lines into a file and then transfers that to the Client machines. After transferring the file the Server sends a request to the Client for updating the autoexec.bat file. After receiving the request, the Client checks for the file and then reads the lines from that file and writes the line into the autoexec.bat file.

Message Display

When the Server program sends a request to the Client System for message display the Client machine receives the message and displays it in a message box in the Client System.

Registry update

The Server machine takes two copies of the local registry before installation and after installation. It finds new changes that are made in the registry. Then the Server machine sends the updated registry to the

Client machine. Also Server program sends the request to the Client program .As soon as the Client machine receives this request from Server machine, the Client machine updates that local machine's registry by exporting the updated registry file [1].

Testing

7. TESTING

Testing is a mandatory and continuous process for the successful completion of any project. Testing is done at various levels. Integration testing follows Unit Testing. Here, different dependant units are assembled and tested for any bugs that may surface due to the integration of the modules.

A final testing is done to access the correctness of the whole software as such. Here several test cases are taken and real life situations are created while testing. The program timings are also noted and compared with present or express results.

7.1 UNIT TESTING

In the installation module, during file transfer from the server to the client, data was getting transferred in a much slower rate. This was rectified and corrected by increasing the byte transfer rate.

When transferring the windows directory contents from the server to the clients, 'Unknown path error message' occurred. It was found that by default the windows directory were updated at the client side assuming

the same path as that of the server system. Since it is not necessary that the client systems should have the windows directory in the same path as that of the server, its source was detected and then updating was done.

Initially during file transfer the process of transfer wasn't knowledgeable as the screen appeared stand still since larger files naturally took a span of moments to transfer. Hence as an indication of file transfer a status bar was included.

7.2 INTEGRATION TESTING

While integrating Installation module and Administration modules many errors were listed due to the usage of individual sockets for each of the modules. Hence a common socket programming was written by introducing a home page.

The Regression testing, the last step in the Integration Testing was carried out to find out various complication that aroused in the individual modules.

7.3 CONDITION TESTING

The condition testing is a test case design method that exercises the logical conditions contained in a program module. The various condition testing strategies carried out were as follows,

- **Branch and Relational operator Testing**

This technique was performed to guarantee the detection of branch and relational operator errors in a condition.

- **Loop Testing**

All simple loops, nested loops, concatenated loops and unstructured loops were tested, bugs were detected and corrected.

7.4 EXPERIMENTAL RESULTS

This describes the tests that are conducted for Installation and Administration processes.

7.4.1 ASSUMPTIONS

The different assumptions for Installation and Administration are explained as follows

- The Systems in the network should be given full access rights.
- The Software can be installed only to c: of the Client Systems.

- The Software can be installed only in windows platform.
- The Client Software should run in the Client System at the time of installation.
- The Client System to which the Software has to be installed should be switched on at the time of installation.
- The IP-ADDRESS and SYSTEM NAME should be Unique for the Clients.

7.4.2 EXPERIMENTS CONDUCTED

The different tests that are conducted for Installation and Administration are explained as follows.

Mathcad7.0 Software is installed to 10 Clients. At the initial stage of the Software installation, nine of them are connected to the Server. They are selected for installing the Software. The 10^{th} Client is allowed to enter into his login during installation process. Among the selected Clients one of the Clients is deficient of space. During the installation process, it is found that the Software is installed to all the eight Clients and a message is popped informing about the insufficient space for that particular Client. The 10^{th} Client is found connected and the Software is installed to him too. As the Software made the System

reboot during installation process for updating of Registry, all the 10 Clients are rebooted from Server to get their Registry updated.

The JDK 1.3 Software is copied from the CD - Drive and dumped in the Server System. This operation does not modify the registry hence the deference of registry before and after dumping the software is nil. Then a group of Clients are selected and the file transfer operation is initiated. The Clients are verified and found that the above mentioned operation is done successfully.

Installation of MSN Messenger Software to a certain number of Clients is performed here. The Client Software is installed to the Clients. A System in which the Server Software is running becomes the Server. For instance, consider that 10 Clients are connected to the Server. The MSN Software is installed in the Server. Then all the 10 Clients are selected in the Client installation part for installing that Software. Now, the whole Software is found installed in both the Clients and the Server successfully.

Hence, the experiments that are made proved the Correctness of the project and is worth useful in any kind of installation of Software.

CONCLUSION

The project "Centralized Installation and Administration" can be used where there are large numbers of clients and different software are to be installed often. This project could be used to monitor the operation of clients and can be logged off their session by the administration when they go beyond their restrictions.

Having utilized VC++ to develop this project, the profundity with which most of the operations were supported gave clearer insight into the tool. VC++ supported many components in flexible handling of registry, which was a part in revealing the successful implementation of the project.

As when compared to the existing system of software installation over multiple clients in a network, the developed software was found to be much faster and the results were accurate and precise.

This endeavor has triggered my ideas to perform even more upgrades in future.

FUTURE OUTLOOK

In this project, installation of software can be done only in 'c' drive of the client and server System. This can be enhanced by providing the selection of the logical drives that are present in the client systems. This makes the Software more general.

The files are transferred by means of an API, which requires the logical drive to be shared. The security for System files is less guaranteed since the drive is shared. So, the file transfer can be modified with data transfer by means of socket programming.

Keeping track of every application those are running in the client system and displaying the processes in the server machine. If the Administrator finds any client using a restricted application, that process can be destroyed. This helps the Administrator effectively.

Taking a snapshot of the client system at various time intervals. This is another facility that can be added in the Administration part of this project. Actually the client software will take the snapshot of the client, which will be sent to the server. This helps in monitoring the client machines.

BIBLIOGRAPHY

[1] Williams.A.I "Windows 2000, Advance API & System Programming" by DreamTech Press. Fifth Edition-2000.

[2] James.L.Conger "The Waite Group's -Windows API BIBLE" by Galgutia Publications Pvt. Ltd. , 2001

[3] Alok K Sinha "Network Programming in Windows NT" by Thomson Press (INDIA) Ltd. First Edition 1999.

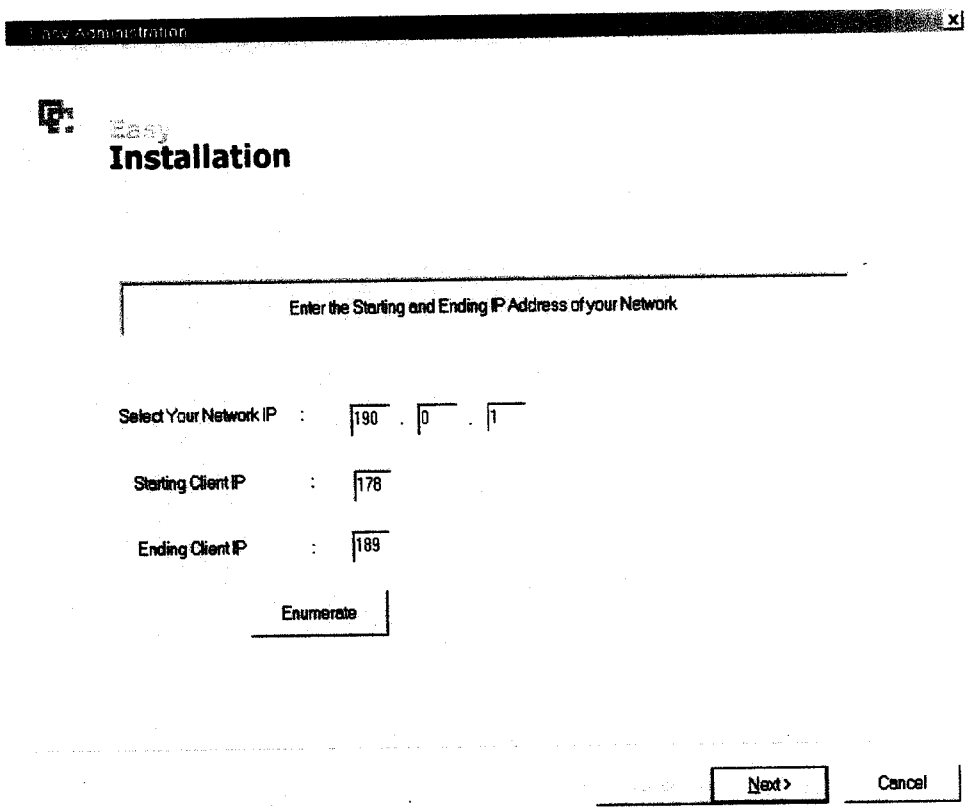
[4] Charles Petzold "Programming Windows" by WP Publishers and Distributors. (P) Ltd. 2000.

[5] Microsoft Visual studio, MSDN library - October 2000 (Software).

SCREEN SHOTS

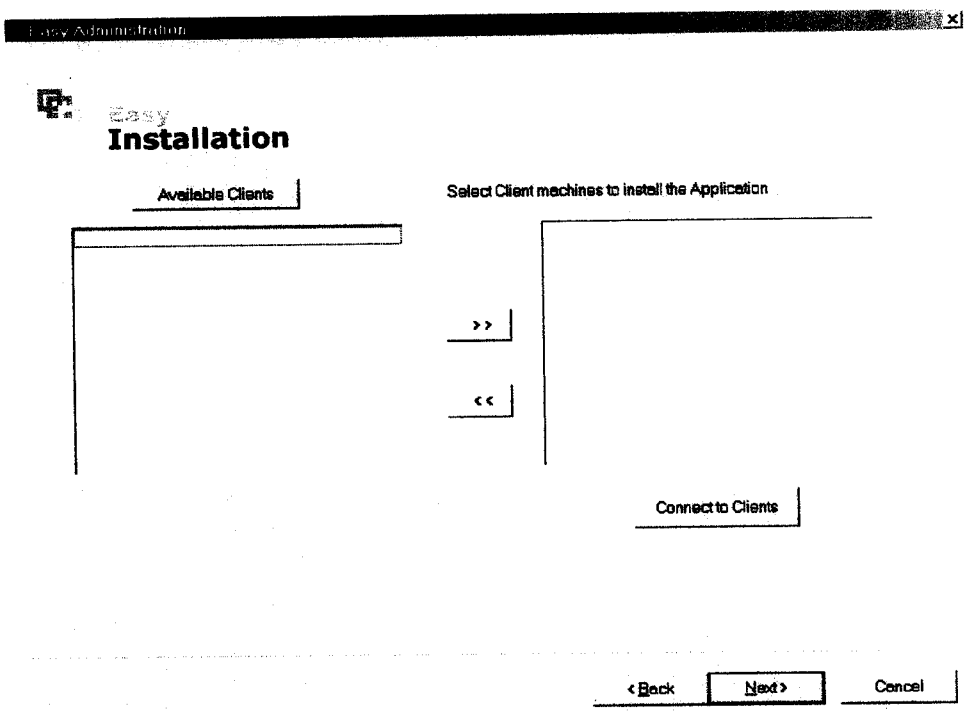
Screen 1

Getting the range of IP address of the computers in the LAN



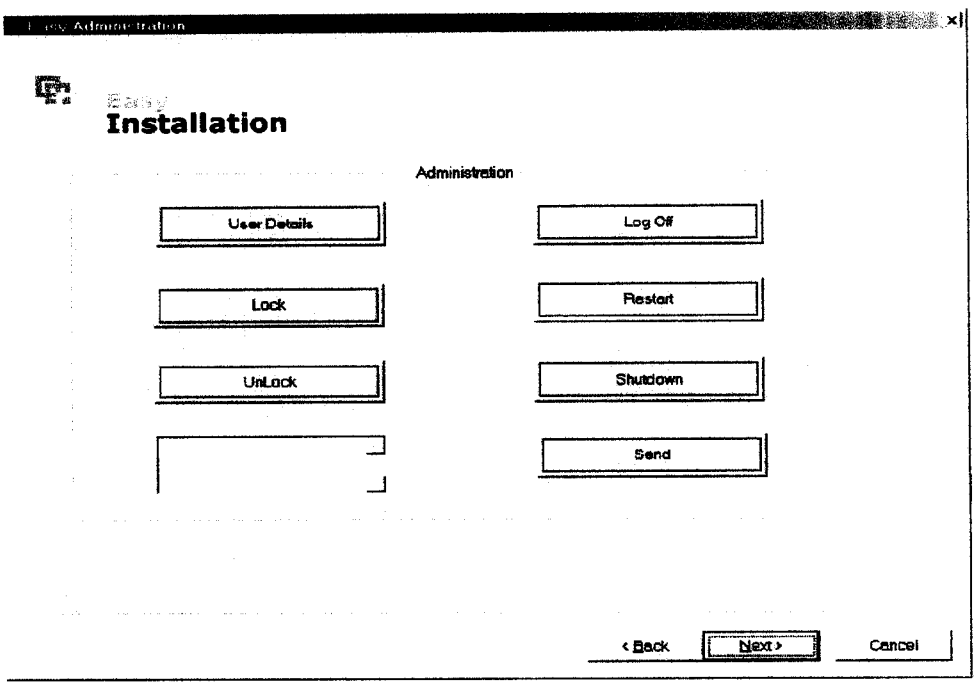
Screen 2

List of available Clients in the LAN and selecting them



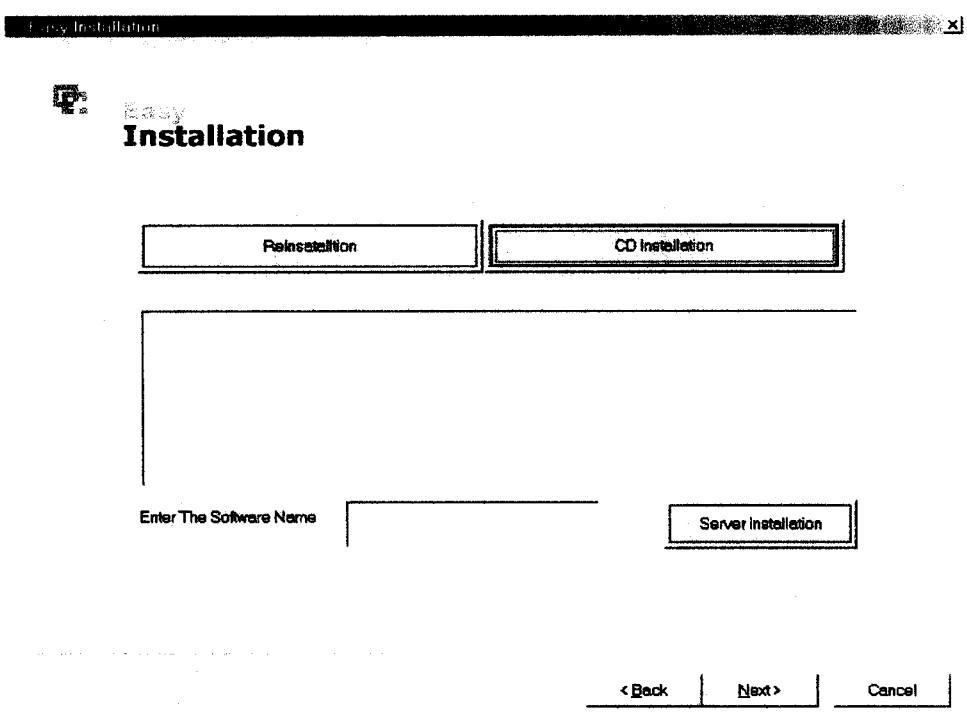
Screen 3

Performing Administration operations on Selected Clients



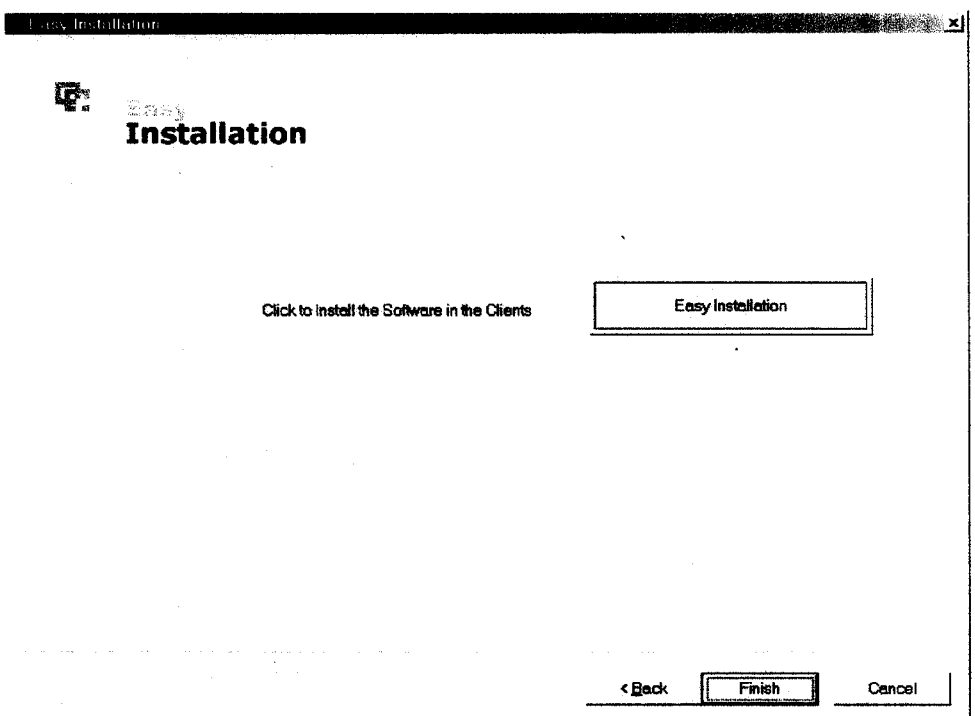
Screen 4

Selecting the Software that has to installed



Screen 5

Installing the Software in all Client Machines



SAMPLE SOCKET PROGRAM

SERVER PROGRAM

```
#include<afxwin.h>
#include<winsock.h>
#define PORT 5100

class cMainWin:public CFrameWnd
{
  public:cMainWin();
  afx_msg void OnLButtonDown(UINT flags,CPoint loc);
  DECLARE_MESSAGE_MAP();
};

cMainWin::cMainWin()
{
        Create(NULL,"WINDOWCREATION");
}

void cMainWin::OnLButtonDown(UINT flags,CPoint loc)
{
        AfxMessageBox("DDD");


        char data[]="I am server";
        char IP_ADDR[]="192.168.0.3";



        // SOCKET PROGRAM DATA'S DECLARATION

        WORD wVersionRequested = MAKEWORD(1,1);
        WSADATA wsaData;
        SOCKADDR_IN saServer;
        SOCKET          theSocket;
        LPHOSTENT lpHostEntry;
        char jet[128];
        int nRet;

        // NETWORKING START'S ON CLIENT SIDE

        //********************** START UP


        nRet = WSAStartup(wVersionRequested, &wsaData);

        if (wsaData.wVersion != wVersionRequested)
        {
        //Sock_pgm=false;
          return;
```

```
}

//********************** SOCKET

theSocket = socket(AF_INET,                 // ADDRESS FAMILY
                   SOCK_STREAM,             // SOCKET TYPE
                   IPPROTO_TCP);            // PROTOCOL
          if (theSocket == INVALID_SOCKET)
                   {
                   //       Sock_pgm=false;
                            return;
                   }


//********************** HOST NAME

          gethostname(jet,sizeof(jet));

          lpHostEntry = gethostbyname(jet);

          if (lpHostEntry == NULL)
                   {
                   //       Sock_pgm=false;
                   return;
                            }

          //********************** FILL IN THE ADDRESS STRUCTURE

          saServer.sin_family = AF_INET;
          saServer.sin_addr.s_addr=inet_addr(IP_ADDR);
          saServer.sin_port = htons(PORT);  // PORT NUMBER FROM COMMAND LINE

          //********************** CONNECT TO THE SERVER

 nRet = connect(theSocket,                        // SOCKET
          (LPSOCKADDR)&saServer,          // SERVER ADDRESS
          sizeof(struct sockaddr));// LENGTH OF SERVER ADDRESS STRUCTURE
                            if (nRet == SOCKET_ERROR)
                            {
                                     int error=WSAGetLastError();
                                     //       Sock_pgm=false;
                                     closesocket(theSocket);
                                     return ;
                            }

          // SENDING DATA TO CLIENTS

                   char szBuf[240];
                   memset(szBuf, 0, sizeof(szBuf));
                   strcpy(szBuf, data);

                   MessageBox(hwnd,"SEND1","HELLO",1);
```

```
                    nRet = send(theSocket,szBuf,strlen(szBuf),0);


                    if (nRet == SOCKET_ERROR)
                    {
                    //        Sock_pgm=false;
                    closesocket(theSocket);
                            return;
                    }

                    if(true)//Receive)
                    {
                    memset(szBuf, 0, sizeof(szBuf));
                    nRet = recv(theSocket,szBuf,sizeof(szBuf),0);

                    if (nRet == SOCKET_ERROR)
                    {
                    //        Sock_pgm=false;
                    closesocket(theSocket);
                            return;
                    }
            }
                    MessageBox(szBuf,"HELLO",1);
            //      Sock_pgm=true;
                    closesocket(theSocket);
                    return;


}

BEGIN_MESSAGE_MAP(cMainWin,CFrameWnd)
 ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()


class cApp:public CWinApp
{
//cMainWin *min;
public :BOOL InitInstance();
};

BOOL cApp::InitInstance()
{

 m_pMainWnd=new cMainWin;
 m_pMainWnd -> ShowWindow(m_nCmdShow);
 m_pMainWnd -> UpdateWindow();
 return true;
}

cApp App;
```

CLIENT PROGRAM

```
#include<afxwin.h>
#include<winsock.h>
#define PORT 5100

class cMainWin:public CFrameWnd
{
  public:cMainWin();
  afx_msg void OnLButtonDown(UINT flags,CPoint loc);
  DECLARE_MESSAGE_MAP();
};

cMainWin::cMainWin()
{
        Create(NULL,"WINDOWCREATION");
}

void cMainWin::OnLButtonDown(UINT flags,CPoint loc)
{
        AfxMessageBox("DDD");


//      char data[]="I am server";
//      char IP_ADDR[]="169.254.213.169";


                //Data Declaraton for Socket program
                WORD wVersionRequested = MAKEWORD(1,1);
                WSADATA wsaData;
                SOCKET          remoteSocket;
                int nRet;
                __int64 TotalFree = 0;
                __int64 TotalBytes = 0;
                __int64 FreeCaller = 0;



                //Networking starts on server
                // **********statup
                        nRet = WSAStartup(wVersionRequested, &wsaData);
                        SOCKET          listenSocket;

                // **********socket creation

                listenSocket = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);

                if (listenSocket == INVALID_SOCKET)
                        {
                                closesocket(listenSocket);
                        }
```

```
// ********** Fill in the address structure

        SOCKADDR_IN saServer;
        saServer.sin_family = AF_INET;
        saServer.sin_addr.s_addr = INADDR_ANY;// Let WinSock supply address
        saServer.sin_port = htons(PORT);          // Use port from command line

// ********** bind the name to the socket


nRet = bind(listenSocket,(LPSOCKADDR)&saServer,sizeof(struct sockaddr));

        if (nRet == SOCKET_ERROR)
                {
                        closesocket(listenSocket);
                }

        // **********host name

        int nLen;
        nLen = sizeof(SOCKADDR);
        char szBuf[240];
        nRet = gethostname(szBuf, sizeof(szBuf));

                if (nRet == SOCKET_ERROR)
                        {
                                closesocket(listenSocket);
                        }

// **********listen


nRet = listen(listenSocket,SOMAXCONN); // Bound socket   // Number of connection request queue
                        if (nRet == SOCKET_ERROR)
                        {
//                      MessageBox(hwnd,"Listening error","error",MB_OK);
                                closesocket(listenSocket);
                        }


// ********** Wait for an incoming request

remoteSocket = accept(listenSocket,             // Listening socket
                        NULL,                   // Optional client address
                         NULL);
                if (remoteSocket == INVALID_SOCKET)
                        {
                                closesocket(listenSocket);
                        }

                memset(szBuf, 0, sizeof(szBuf));
                nRet = recv(remoteSocket,szBuf,sizeof(szBuf),0);
```

```
                    MessageBox(szBuf,"Hello",1);
                    send(remoteSocket,"Hello i am client",sizeof(szBuf),0);



                closesocket(remoteSocket);
                closesocket(listenSocket);
                WSACleanup();

                return;


}

BEGIN_MESSAGE_MAP(cMainWin,CFrameWnd)
 ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()


class cApp:public CWinApp
{
//cMainWin *min;
public :BOOL InitInstance();
};

BOOL cApp::InitInstance()
{

 m_pMainWnd=new cMainWin;
 m_pMainWnd -> ShowWindow(m_nCmdShow);
 m_pMainWnd -> UpdateWindow();
 return true;
}

cApp App;
```