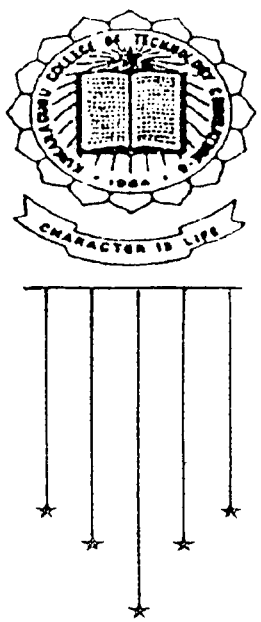
CAsyncSocket DERIVED CLASS TO CONNECT THROUGH PROXIES

PROJECT REPORT P-1091

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

MSc. (APPLIED SCIENCE –SOFTWARE ENGINEERING)
OF BHARATHIAR UNIVERSITY



Submitted by

SWAPNA MATHEW
Reg. no- 0037S0111

Guided by

Mr. R.K. Gnanamurthy, M.E.

Department of Computer Science and Engineering
Kumaraguru College Of Technology

(Affiliated to Bharathiar University)

Coimbatore-641006

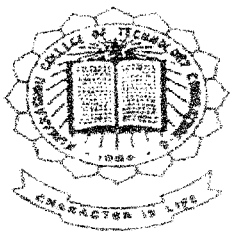
SEPTEMBER 2003

CERTIFICATE

Department of Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore – 641 006



This is to certify that the project work entitled

"CAsyncSocket derived class to connect through Proxies"

has been submitted by

Miss. SWAPNA MATHEW

(Reg.No: 0037S0111)

In partial fulfillment of the award of the degree of

Master of Science in Applied Science – Software Engineering of

Bharathiar University, Coimbatore

During the academic year 2003-2004

29/9/2003.

Guide

Head of the Department

Certified that the candidate was examined by us in the Project Work Viva Voce Examination held on 29-09-03.

29/9

Internal Examiner

29/9

External Examiner

DECLARATION

I here by declare that the project work entitled

"CAsyncSocket Derived Class To Connect Through Proxies"

done at

ELECTRONICS RESEARCH AND DEVELOPMENT CENTRE OF INDIA TRIVANDRUM

and submitted to

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University)

in partial fulfillment of the requirement for the degree of MSc(Applied Science-Software Engineering)

is a report done by me during the period of study in

Kumaraguru College of Technology, Coimbatore-641006.

Under the supervision of

Mr. R.K.Gnanamurthy,M.E.

Name of the Candidate	Register Number	Signature of Candidate
Swapna Mathew	**0037S0111**	

Date: 25-09-03

Software Training and Development Centre
3rd Floor, Chennankara Building
Vellayambalam, Thiruvananthapuram 695 010
Tel. 0471 - 2726531, 2726586, 2726012. Fax: 0471 - 2721209
E-mail: stdc@erdcitvm.org, Website: www.erdcitvm.org

A Scientific Society of the Ministry of Communications and Information Technology, Government of India

CDAC
The Supercomputing People
CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING

(Formerly ER&DCI)
Vellayamblam,
Thiruvananthapuram 695 033, India
Tel: +91- 471-272 3333
Fax: +91- 471-272 3456. 2722230
http://www.cdacindia.com

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"Proxy Server"** using **VC++** is a bonafide record of the work done at the Centre by **Ms. Swapna Mathew,** Kumaraguru College of Technology, Coimbatore, in partial fulfillment of the requirements for the award of the **MSc (SE)** degree from **Bharathiar University.** She has been working on the project in the Centre during the period June 2003 to September 2003.

Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis on the basis of which a degree or award was conferred on an earlier occasion to this or any other candidate.

Prof. R Sahadevan
Head, STDC

SOFTWARE TRAINING AND DEVELOPMENT CENTRE
A SCIENTIFIC SOCIETY OF THE MINISTRY OF COMM. & INFO. TECHNOLOGY
CDAC
GOVT. OF INDIA
T.C.15/180-184
TRIVANDRUM
Pin-695 010

Vinnie John Thottan
Scientific Officer

(Project Guide)

Thiruvananthapuram
26th September 2003

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I express my sincere gratitude to our principal **Dr.K.K.Padmanabhan :.(Engg),M.Tech, Phd**. I extend my gratitude to **Dr. Thangaswamy B.E.(hons),Ph.d,** beloved **H.O.D** for his constant support, encouragement and valuable internal dance.

I express my sincere thanks to internal guide **Mr R.K.Gnanamurthy,M.E.**for ›rall support that he Provided during the project work.

I am greatly indebted to **Mrs Saramma Chacko ,Joint Director, ER&DCI, vandrum** for providing me the golden opportunity to do my project at their anization, and sparing her time with me to give all details about the organization.

I am grateful to **Mr.Vinnie John Thottan, Scientific Officer** and project guide sparing his valuable time in successful completion of this project.

I also take this opportunity to thank **my parents** for their constant support and ›uragement.

Last but not the least I thank **GOD** for his blessings for the success of our project.

SYNOPSIS

SYNOPSIS

Internet is computer-based worldwide information's network. The internet is composed of a large number of interconnected networks connecting tens, hundreds, or thousands and more of computers, enable them to share information with each other and to share various resources such as powerful supercomputers and databases of information. The internet has made it possible for people all over the world to effectively communicate with each other.

A proxy server is used to safely add internet capabilities to the network environment. A proxy server is a system that operates between client applications such as web browser and a server. It intercepts all request sent to the server to see if it can fulfill the request. If not, it forwards the request to the server. The main feature used in the proxy server is TCP/IP Protocols. TCP Protocols are mainly used for the connection – oriented protocols. Once the connection is established, the connection is closed when the client closes the connection or the allocated time for the client is finished.

The main services provided by Proxy Server are:

- Site Restriction
- Cache Management
- Remote Administration
- Mail Client

Proxy servers have two main purposes: to improve performance and to filter requests. It allows sites to be censored from view. A specific time can be assigned to each client and it helps in restricting machines from accessing the network.

Remote administration helps to monitor the process on every machine, shutdown and restart any client.

The mail client using the Post Office Protocol 3 here helps in simple mail reading and mail sending options. POP 3 is a standard mail protocol, a set of formal rules that work behind the screen to determine how message data gets transmitted across a computer network. When using a POP 3 mail client, new mail messages are downloaded from your notes mail database on the server to your local workstation.

CONTENTS

CONTENTS

INTRODUCTION

1. INTRODUCTION

1.1 Overview of the Project

The project is mainly developed filter the requests from the client. It allows sites to be censored from view. A proxy server is used to safely add internet capabilities to the network environment. A proxy server is a system that operates between a client application such as web browser and a server. It intercepts all requests sent to the server to see if it can fulfill the request. If not, it forwards the request to the server.

The main feature used in the proxy server is TCP/IP Protocols. TCP Protocols are mainly used for the connection – oriented protocols. Once the connection is established, the connection is closed when the client closes the connection.

The main services provided by Proxy Server are:

- Site Restriction
- Cache Management
- Remote Administration
- Mail Client

Remote administration helps to monitor the process on every machine in the LAN. Also, remote administration helps to shutdown or restart any machine in the LAN.

The mail client using the Post Office Protocol 3 here helps in simple mail reading and mail sending options. POP 3 is a standard mail protocol. When using a POP 3 mail client, new mail messages are downloaded from your notes mail database on the server to your local workstation.

1.2 Organization Profile

CDAC (FORMER ER&DCI) is a scientific society, which was originally started in 1974 under the department of electronics, Govt.of India. It is the pioneer organization engaged in application oriented research and development in electronics and information technology, providing services to practically all government departments as well as private organizations.

The centre has developed strong expertise over the years in the area of computer communication, instrumentation, and control, artificial intelligence, power electronics etc. The resources available in ER&DCI include an IBM- ES-9000 MAINFRAME SYSTEM, VLSI DESIGN CENTRE and advanced CAD facilities.

The centre's latest venture is the Software Training and Development Centre (STDC) Addressing to the needs of software development – market on mainframe platform. This centre is equipped with the advanced IBM- ES-9000 MAINFRAME COMPUTER SYSTEM and related software. It provides training on MVS, DB2, CICS and JCL position to address the major commercial segment of mainframe users.

STDC undertakes software development on the IBM mainframe platform. It has a team of dedicated and well-trained software professional.

ORGANIZATIONAL SET UP

There are six ER&DCI's located in the country. They are at

1. CALCUTTA
2. MOHALI
3. LUCKNOW
4. ERNAKULAM
5. THIRUVANANTHAPURAM
6. DELHI

SYSTEM STUDY AND ANALYSIS

2. SYSTEM STUDY AND ANALYSIS

2.1 Existing System

Installing a full – fledged connection to the internet can cost quite a bit. At present every user can access the internet facility. They are able to access any sites they want to. There is no restriction in accessing the sites.

Limitations of the Existing System

There is no software available for sharing, monitoring internet access and hence updating the details in the current user database.

Goals to be achieved

A reliable internet connection sharing software that monitor and record internet access log.

2.2 Proposed System

Due to the drawbacks and inability of the existing system there is a strong need for developing customized internet connection distribution software. The proposed solution "Proxy Server" will use connection oriented TCP/IP protocols. This will effectively mediate and distribute internet access in a Local Area Network and enforce access authority security refinements and share bandwidth. Proxy server should be installed in a computer that is connected to the web and client in the LAN. This client and server will communicate each other and share the bandwidth. Just 'Click!' 'Click!' no more administrating manually.

Objectives of the Proposed System

- It acts as an intermediary between a Web Browser and a server.
- It caches frequently accessed Web Pages over a desired period of time.
- It helps in remote administration.
- Also, it provides a simple mailing facility using POP3.

2.3 Study on Proposed System

The proposed system has the following modules.

- Site and User Restriction
- Cache Management
- Remote Administration
- POP3 Mail Client

Site and User Restriction

The proxy server can restrict some particular sites. Restricting the sites in the sense improving the security. In our software system, also user authentication has got its importance. As proxy servers are aimed at external network security, only authorized users are provided with access rights to the system. Here user authentication is provided with the help of Dialog boxes and controls at the entry point of the program. Only after the user is authenticated the software starts itself. Also it is possible to change the password. For this the administrator has to authenticate himself by providing the user name and password.

Cache Management

In some sites, certain pages are frequently used rather than the home page. Like in yahoo instead of home page, the sign up page is most commonly used. So we can provide options in settings to cache the sign up page directly instead of home page.

Remote Administration

Here a client program is run on every machine that is fully listening. This program may get the message from the proxy server through the socket, and it reacts according to the message received. The client program when it receives 1, it makes the windows to shutdown. If 2, then restarts the client and if it receives 3, this client program sends the details of the process that runs on the machine. These values 1, 2 and 3 are generated according to the options that the administrator will choose.

POP3 Mail Client

The mail client using the Post Office Protocol 3 here helps in simple mail reading and mail sending options. POP 3 is a standard mail protocol. When using a POP 3 mail client, new mail messages are downloaded from your notes mail database on the server to your local workstation.

2.4 Functional Requirements of Proposed System

The proxy server run on the net machine seeks the request from the client. It accepts the requests and opens a TCP Socket and reads the data and sends it to a particular client. TCP is a connection-based technology that behaves in a stream oriented manner. Proxy Server is created with an objective to create an access point to the internet that can be shared by all the users on a LAN. This software uses the abstraction socket as an interface between application programs and TCP/IP protocols.

Creating a Socket

The ***Socket*** system call creates sockets on demand. It takes three integer arguments and returns an integer result.

Result = socket (pf, type, protocol)

Here 'pf' specifies the protocol family to be used with the socket i.e., it specifies how to interpret the addresses when they are supplied.

'Type' specifies the type of communication desired. Here reliable stream delivery service is required and hence the type is SOCK STREAM. Then a specific protocol has to be selected from the protocol family.

Closing a socket

When a process finishes using a socket it calls ***close***. Close has the form:

Close (socket)

Here the argument socket specifies the descriptor of a socket of close. When a process terminates for any reason, the system closes all sockets that remain open.

Specifying a Local Address

Once a socket has been created, a server uses the ***bind*** system call to establish a local address for it. Bind has the following form:

Bind (socket, localaddr, addrlen)

Argument socket is the integer descriptor of the socket to be bound. Argument ***localaddr*** is a structure that specifies the local address to which the socket should be bound and the argument ***addrlen*** is an integer that specifies the length of the

address measured in bytes. Instead of giving the address merely as a sequence of bytes, the ***sockaddr*** structure is used when passing a TCP/IP address to the socket interphase.

Connecting sockets to destination address

Initially a socket is created in the unconnected state, which means that the socket is not associated with any foreign destination. The system call ***connect*** binds the permanent destination to a socket, placing it in the connected state. The ***connect*** system call has the following form:

Connect (socket, destaddr, addrlen)

Argument ***socket*** is the inter descriptor of the socket to connect. Argument ***destaddr*** is a socket address structure that specifies the destination address to which the socket should be bound. Argument ***addrlen*** specifies the length of the destination of address measured in bytes.

Sending data through a socket

Once an application program has established a socket, it can use the socket to transmit the data. The system call write takes three arguments:

Write (socket, buffer, length)

Argument ***socket*** contains an integer socket descriptor. Argument ***buffer*** contains the address of the data to be sent, and argument ***length*** specifies the number of bytes to be send.

The send system called has the following form:

Send (socket, message, length, flags)

Argument ***socket*** specifies the socket use, argument ***message*** gives the address of the data to be send, argument ***length*** specifies the number of bytes to be sent and argument ***flags*** controls the transmission.

Receiving data through a socket

Process call recv is used to receive data from a connected socket. It has the form:

Recv (socket, buffer, length, flags)

Argument ***socket*** specifies a socket descriptor from which data should be received. Argument ***buffer*** specifies the address in memory in to which the message should be placed, and argument ***length*** specifies length of buffer area. Finally, argument ***flags*** allow the caller to control the reception.

Specifies a queue length for a server

The system call ***listen*** allows server to prepare a socket for incoming connection. When the server invokes ***listen***, it also informs the operand system that the protocol software should enqueue multiple simultaneous requests that arrive at the socket. The form is:

Listen (socket, qlength)

Argument ***socket*** gives the descriptor of a socket that should be prepared for use by a server, and argument ***qlength*** specifies the length of the request queue for that socket. ***Listen*** applies only to socket that have selected reliable stream delivery service.

LIBRARY
Coimbatore 641 006

Server accepting connection

Once a socket has been established the server needs to wait for a connection. To do so, it uses system called ***accept***. A call to ***accept*** blocks until a connection request arrives. It has the form:

Newsock = accept (socket, addr, addrlen)

Argument ***socket*** specifies the descriptor of the socket on which to wait. Argument ***addr*** is a pointer to a structure of type *sockaddr* and ***addrlen*** is a pointer to an integer. When a request arrives, and sets ***addrlen*** to the length of address. Finally, the system creates a new socket and that has its destination connected to the requesting client and returns the new socket descriptor to the caller. The original socket still has a wild card foreign destination and it still remains open. Thus, the master server can continue to accept additional request at the original socket.

Network byte order conversion routines

Machines differ in the way they store integer quantities. The TCP/IP protocols define a machine independent standard for a byte order. There are functions that converts between the local machine byte order and the network standard byte order. To make the program portable, they must be written to call the conversion routines every time they copy an integer value from a network packet to the local machine.

Functions ***htons*** converts a 2 byte (short) integer in the host's local byte order to a 2-byte integer in network standard byte order. Programs invoke ***htons*** as a function:

Netshort = htons (localshort)

IP Address manipulation routines

Procedure ***inet_ntoa*** maps a 32 – bit integer to an ASCII string in dotted decimal format. It has the form:

Str = inet_ntoa (internetaddr)

Where argument ***internetaddr*** is a 32-bit IP address in network byte order, and ***str*** is the address of the resulting ASCII version.

Obtaining information about hosts

A process can retrieve information about a host given either its domain name or its IP address. When used on a machine that has access to a domain name server, the concern functions make the process a client of domain name system by sending a request to a server and waiting for a response.

Function ***gethostname*** takes a domain name and returns a pointer to a structure of information for that host. A call takes the form:

Ptr = gethostname (namestr)

Argument ***namestr*** is a pointer to a character string that contains a domain name for that host. The value returned, ***ptr***, points to a structure that contains the following information:

1) the official host name, a list of aliases that have been registered for the host,

2) the host address type (i.e. whether the address is an IP address)

3) the address length and

4) a list of one or more address for the host.

Function gethostbyaddr produces the same information as accepts gethostbyname. The difference between the two is that gethostbyaddr a host address as an argument:

Ptr = gethostbyaddr (addr, len, type)

Argument ***addr*** is the pointer to sequence of bytes that contains a host address. Argument ***len*** is an integer that gives the length of the address, and argument ***type*** is an integer that specifies the type of the address (eg that it is an IP address).

SOCKETS, PORTS AND ADDRESSES

The basic object used by application to perform most networks communication is called a socket. Sockets were first developed on UNIX at the university of California at Berkley. Sockets were designed so that most network communication between applications could be performed in the same way that these applications would read and write files. Sockets have progressed quite a bit since then, but the basis of how they work is still the same.

During the days of windows 3.x, before networking was built into the windows operations system, you could buy the network protocols required by the network communication from numerous different companies. Each of a company had a slightly different way that an application performed network communication. As a result,

any application that performs network communication had a list of different networking software that the application would work with. Many application developers were not happy with the situation as a result, all the networking companies including Microsoft, got together and developed Winsock (Windows sockets) API. This provides all application developers with a consistent API to perform all network communication, regardless of the networking software used.

Making a socket connection to another application does not require a different set of information than opening a file. To open a file we need to know the file's name and location. To open a socket connection we need the computer on which the application is running and the port on which it is listening.

A port is like a phone extension, and the computer address is like the phone number. If we call someone at a large office building we may dial the same office number, but we need to specify the extension number. Likewise, ports are used to route network communication. As with phone number, there are means of looking up the port number, if you don't already know what it is, but this requires your computer to be configured with the information about which port the connection application is listening on. If specified wrong computer address or port number, we get connection to different application, with making the phone call, someone other than the person called may answer at all if there is no application listening at other end.

CLIENT / SERVER COMMUNICATION – TCP/IP

Client Side

1) Create a TCP socket

2) Connect the socket to a given address and port

3) Send or receive data to/ from the socket via stream

4) Close the socket.

Server Side

1) Create a listener socket for a given port

2) Listen for incoming request from clients

3) Accept incoming request and pass it to a new TCP socket

4) Send or receive data to/from the socket via stream

5) Close both sockets.

SOCKET PROGRAMMING

Definition of socket:

A socket is communication endpoint, an object through which a window sockets application sends or receives packets of data across a network. A socket has a type and is associated with a running process, and it may have a name. Currently, sockets generally exchange data only with other sockets in the "communication domains", which uses the internet protocol suite.

Two socket types are available:

- ***Stream Sockets***

  Stream socket provide for a data flow without record boundaries – a stream of bytes. Stream is guaranteed to be delivered and to be currently sequenced and unduplicated.

- ***Datagram Sockets***

Datagram socket support a record oriented dataflow that does not guaranteed to be delivered and may not be sequenced as send or unduplicated.

Description of socket handle:

Each MFC socket object encapsulates a handle to a window socket object. The data type of this handle is a SOCKET. A SOCKET handle is analogous to the HWND for a window. MFC socket classes provides operation on the encapsulation handle.

Uses for sockets:

Sockets are highly useful in atleast three-communication context:

- Client Server Models
- Peer-to-Peer scenarios, such as chat applications
- Making remote procedure calls (RPC) by having the receiving application intercept a message as a function call.

Socket classes:

Class CSOCKET derives from CAsyncSocket and inherits its encapsulation of the Windows Sockets API. A socket object represents a higher level of abstraction of the window socket API than that of a CAsyncSocket objects. Cocker works with classes Csocket files and Carchive to manage the sending and receiving of data.

A Csocket objects provides blocking, which is essential to the synchronous operation of Carchive. Blocking function such as receive, send, receive from, send to and accept (all inherited from CAsyncSocket), do not return

WSAEWOULDBLOCK error in Csocket. Instead, these functions wait until the operation completes. Additionally the original call will terminate the error WSAEINTR if CancelBlockingCall is called one of this function is blocking.

To use a socket object, call the constructor, and then call Create to create

the underlying SOCKET handle (type SOCKET). The default parameter of CREATE creates a stream socket, but if you are not using the socket with CArchive object, you can specify a parameter to create a datagram socket instead, or bind to a specific call to create a server socket. Connect to a client socket using CONNECT on the client side ACCEPT on the server side. Then create a socket file object and associate it with the CSOCKET FILE object in the CSOCKETFILE constructor next, creates a CArchive object for sending and one for receiving data (as needed), then associate them with the CsocketFile object in the CArchive constructor. When communication are complete, destroy the CARCHIVE, CSOCKETFILE and CSOCKET objects.

CsocketFile

A CsocketFile object is a Cfile used for sending and receiving data across a network via Windows Sockets. To serialize data you insert it to the archive, which call the socket file member function to write data to the socket object. To federalize data, you extract from the archieve.

The windows Socket specification defines a binary compactable network programming interface for Microsoft windows. Windows sockets are based on the UNIX Sockets

implementation.

The specification include both BSD-style socket routines and extension specific to windows. Using windows sockets permits your application to communicate across any network that confirms to the windows socket API.

MAKING A CONNECTION

Once we create a socket we are ready to open a connection with it. Three steps go along with a single connection. Two of these steps take place on the server, the application listening for the connection and the third step take place on the client, the one making the call.

For the client opening the connection is a simple method of calling the connect method, Computer name or network address and the port of application are the parameters to be passed.

Once the connection is made an event is triggered to let the application know that the connection is made. For the server the application must tell the socket to listen for incoming connections by calling the listen method. The listen method takes only a single argument. This parameter specifies the no: of pending connections that can be queued waiting for the connection to be completed.

When another application is trying to connect to the listening application, an event is triggered to let the application know that connection request is there. The listening application must accept the connection request by calling the accept method. This method uses second Csocket variable, which is connected to the other application. Once a socket is placed into the listen mode it stays in listen mode. Whenever connection request are

received the listening socket creates another socket which is connected to the other application. This second socket should not have the Create method called for it because the Accept method creates socket.

SENDING AND RECEIVING MESSAGES

To send a message to a socket connection we use the Send method. This method requires two parameters and has a third, optional parameter that can be used to control how the message is send.

If the message is in a Cstring variable we can use the LPSTR operation to pass the Cstring as the buffer. The second parameter is a length of the buffer.

When the data is available to be received from the other application the event is triggered on receiving application. To get the message, the Receive method must be called. This method takes the same parameter as the send method. The first parameter is a pointer to a buffer into which the messages may be copied. The second parameter is the size of the buffer.

CLOSING THE CONNECTION

Once all the application have finished communicating with each other the communication can be closed by calling the close method.

SOCKET EVENTS

The primary reason that we create our own descendant class of CASyncSocket is to capture the events that are triggered when the messages are received. All the functions of CASynSocket use same definitions. They are declared as protected. The functions have a

single parameter.

Some of the **functions**

OnAccept-> This function is called on a listening socket to signal that a connection request from another application is waiting to be accepted.

OnClose-> This function is called on a socket to signal that the application on the other end of the connection has closed its socket.

OnConnect-> This function is called on a socket to signal that the connection with another application has been completed and that the application can now send and receive messages through the socket.

OnRecieve-> This function is called to signal that the data has been received through the socket connection and that the data is ready to be retrieved by calling the receive function.

OnSend-> This function is called to signal that the socket is ready and available for sending data. This function is called right after the connection has been completed.

DETECTING ERRORS

Whenever any of the CASyncSocket member function return an error, either FALSE for the most functions or SOCKET_ERROR on the Send and Receive function, the GetLastError method can be used to get the error code. This function returns only error codes. All the WinSock error codes are defined with constants, so that the use of the

constants, so that the constants in the code determines error message to display for the user.

PROGRAMMING ENVIRONMENT

3. PROGRAMMING ENVIRONMENT

3.1 Hardware Requirements

Server:

Processor Name: Pentium I I I

Memory (RAM): 128 MB

Hard Disk Drive: 4 GB

Monitor: VGA Color (Resolution 640x480)

Network Interface: 100 Mbps Ethernet Card

Key Board: Standard 101/102 or Microsoft Natural Keyboard

Floppy Drive: 1.44 MB

Mouse: Logitech Serial Mouse

Printer: Inkjet

Client:

Personal Computer that runs internet explorer 4.0 or higher/

Nescape Navigator 4.5 or higher

Internet connection:

Internet connection using Modem / cable Modem

3.2 Software Requirements

Operating system: Windows 98

Software Tool: Microsoft VC++

Back–End Tool: Microsoft Access

3.3 Software Description

Visual C++ - 6.0

Microsoft Visual C++ version 6.0 is available in three editions: Standard, Professional, and Enterprise.

Win32 API

An acronym for Application Programming Interface, the set of commands that an application uses and carries out lower-level services performed by a computer's operating system.

A set of routines used by an application to direct the performance of procedures by a computer's operating system. For computers running a graphical user interfaces, an API manages an application's windows, icons, menus and dialog boxes. An API is simply a set of functions that you can use to work with a component, application or operating system. Typically an API consist of one or more DLLs that provide some specific functionality.

DLLs are files that contain functions that can be called from any application running in windows. At runtime, a function in DLL is dynamically linked into an application that calls it. No matter how many applications call a function in DLL, that function exits in only a single file on the disk, and the DLL is loaded only once in

memory.

The API you've probably heard about most frequently is the Windows API, which includes the DLLs that make up the Windows Operating System. Every Windows application interacts with the Windows API directly or indirectly. The Windows API ensures that all applications running under Windows will behave in a consistent manner.

MFC

The Microsoft Foundation Class Library (MFC) is an "application framework" for programming in Microsoft Windows. Written in C++, MFC provides much of the code necessary for managing windows, menus and dialog boxes: performing basic input/output; storing allocation of data objects; and so on. All you need to do is add your application specific code into this framework. And, give the nature of C++ class programming, it's easy to extend or override the basic functionality the MFC supplies.

MS-Access

A database is a collection of information, which contains records and fields. Record is a collection of different types of information about same subject. A field is a category of information.

MS-Access is a relational database management system in short, RDBMS. An RDBMS utilizes two or more table, containing data arranged in rows and columns, to cross-reference and define relationships between data.

SYSTEM DESIGN

4. SYSTEM DESIGN

A system is a combination of resources working together to convert input to usable outputs. The most creative and challenging phase of the system life cycle is system design. System design is a process of developing specifications for a candidate system that meet the criteria established in the system analysis. Design requires a full understanding of the problem; there is a need for analysis of requirement and resources.

The acceptable design is likely to be a compromise among a number of factors particularly cost, reliability, accuracy, security, control, integration, expandability. It is the process of planning a new system to replace the existing system. The ideas developed from the system analysis are used for the new system.

The first step is to determine how the output is to be produced and in what format. Samples of the output and the input are also presented. Second, input data and the database have to be designed to meet the requirements of the proposed output. The operational phases involve program construction and testing.

4.1 Input Design

Input design determines the format and validation criteria for data entering in the system. It is the second task of the system design phase of the system processes. It is a process of converting a user – oriented description of the inputs to a computer based format. It is the point of contact for the user with computer system. Several activities have to be carried out as a part of the overall input process. They are data recording, data transcription, data verification, data control, data transmission, data validation and data correction.

The following features have been incorporated into the input design of the proposed system.

- ***Easy Data Entry***

Data entry screens have been designed in a manner much similar to the source documents. Appropriate messages are provided which prompts the user in entering the right data. Erroneous data inputs are checked at the end of each screen entry.

- ***Data Validation***

The input data is validated before starting to minimize errors in data entry. Validation can be done with respect to the data types and appropriate error messages are provided in encouraging errors.

- ***User Friendliness***

User is never left in a state of confusion as to what is happening,

instead appropriate error and acknowledgment messages are sent. Error maps are used to indicate the error codes and specific error messages.

- ***Consistent Format***

Every screen has a status line, which displays the operation that can be performed after data entry. They are normally done at the touch of the key. The form title clearly states the purpose of the form. The layout is such that the person filling the form can process in one direction and does not need to move back and forth the form.

- ***Interactive Dialogue***

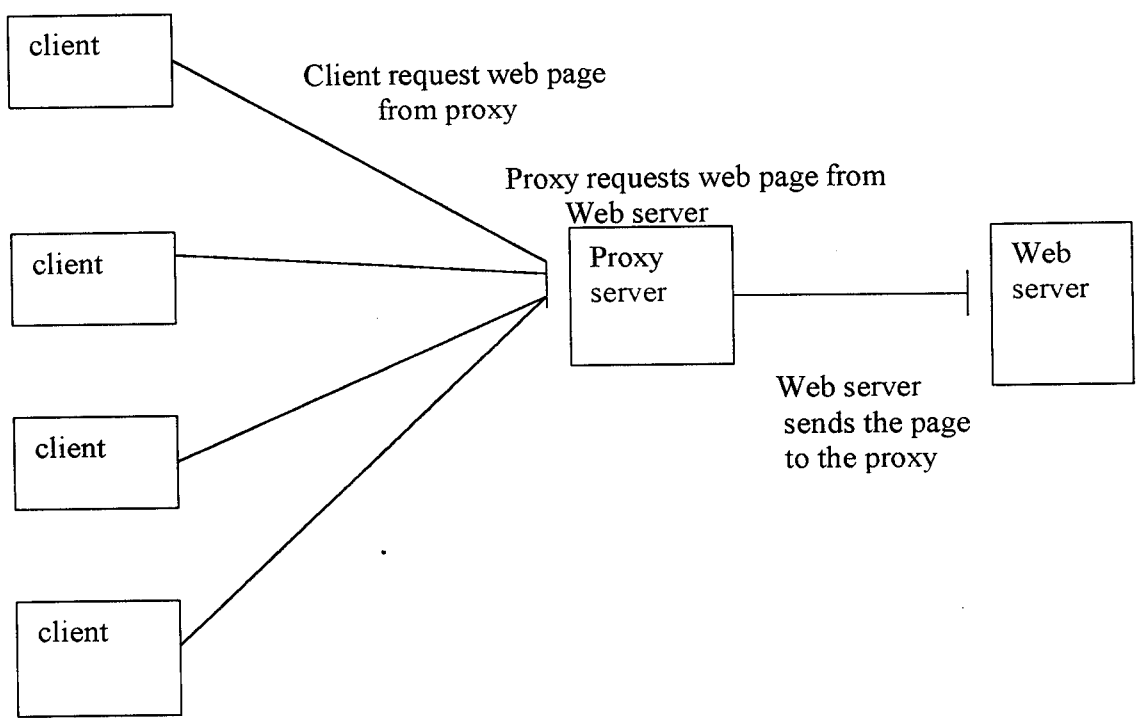
The system engages the user in an interactive dialogue. The system is able to extract missing or admitted information from the user by directing the user through appropriate message, which are displayed.

4.2 Output Design

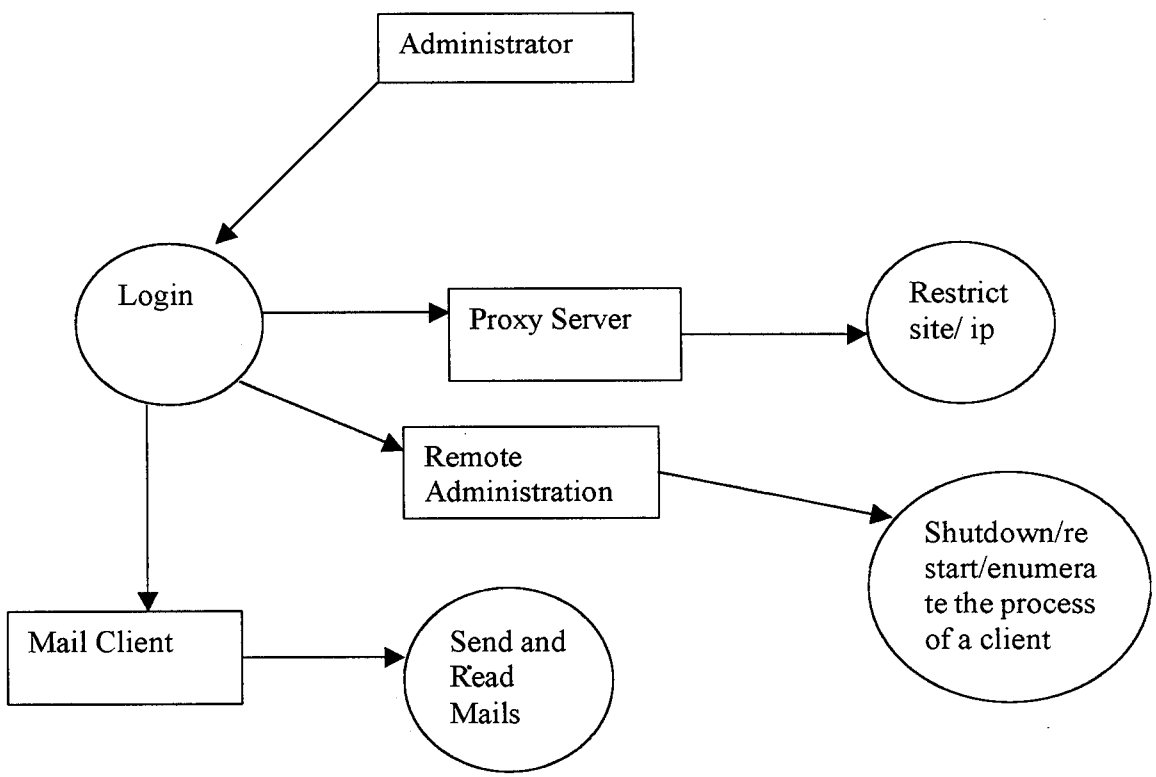
Output design is the first phase of the system design process. The output design should improve the system's relationship with the user and help in Decision-making. It is primarily required to communicate the user's input from the client side to the server where a program will be run to process the inputs. Once the output requirements are determined, the system designer can decide what to include in

the system and how to structure it so that the required output can be produced. For the proposed software it is necessary that output reports be compatible in format with the manual reports.

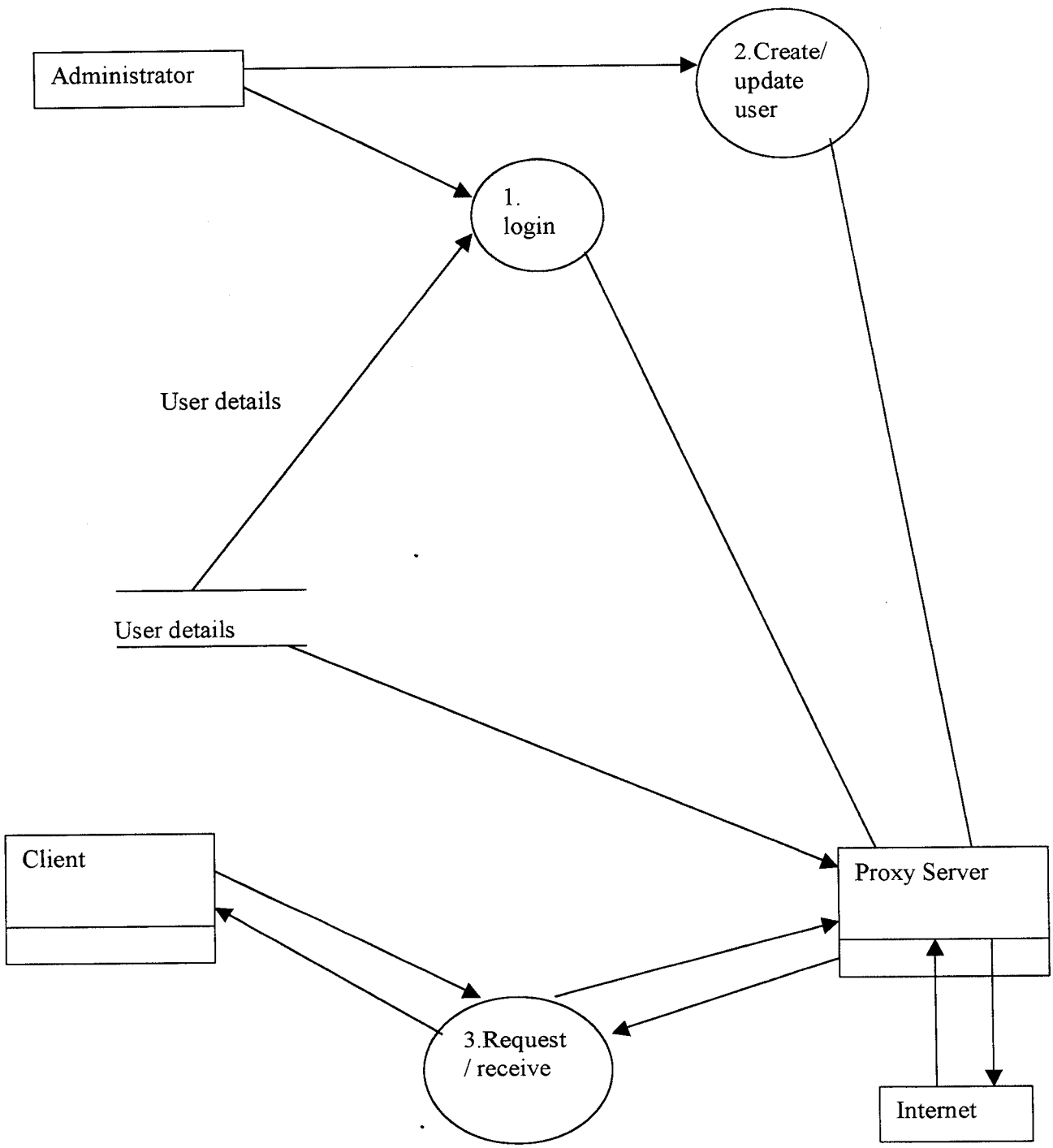
4.3 Process Design



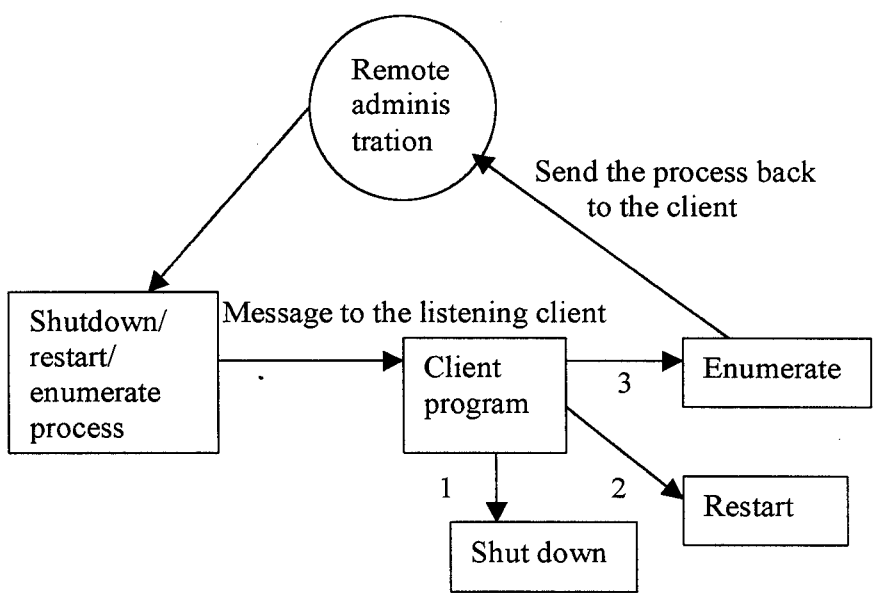
CONTEXT ANALYSIS DIAGRAM



GENERAL DATA FLOW DIAGRAM



DATA FLOW DIAGRAM OF PROCESS REMOTE ADMINISTRATION



4.4 Database Design

LOGIN

FIELD NAME	DATA TYPE	DESCRIPTION
UserName	Varchar2(25)	User Name
Password	Varchar2 (25)	Password
Administrator	Boolean	To differentiate administrator and other users

CACHE

FIELD NAME	DATA TYPE	DESCRIPTION
URL	Varchar2 (25)	URL of the site to be cached
COUNT	Integer (3)	Count no. of times the site is cached

PORT

FIELD NAME	DATA TYPE	DESCRIPTION
URL	Varchar2 (25)	URL that is to be restricted

SYSTEM TESTING AND IMPLEMENTAION

5. SYSTEM TESTING AND IMPLEMENTATION

5.1 SYSTEM TESTING

The software testing is the process that commences once the program is created and the documentation, related data structure are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete.

SOFTWARE TESTING FUNDAMENTAL

Testing presents an interesting anomaly for software engineers. Earlier in the software process, the engineer attemps to build the software from an abstract concept to a tangible implementation. Now comes the testing. The engineer creates a series of test cases that are intended to "demolish" the software that has been built. In fact testing is one step in the software engineering process that could be viewed as destructive rather than constructive.

Testing requires that the developer discard preconceived notions of "correctness" of software just developed and overcome a conflict of interest that occurs when error's are uncovered.

STRATEGIES AND TECHNIQUES

Testing is an activity associated with any process that produces a product. It is used to determine the status of the product during and after the build or do component of the process. Software testing is an integral part of the software

development process, which comprises the following four components:

PLAN	Device a test plan
DO	Execute the test plan
CHECK	Check the result of test plan execution
ACT	Take necessary action.

Vested interested parties in software testing

Customer: The party or department that contracts for the software to be developed.

User: The individual or group that will use the software once it is placed into production.

Tester: The individual or group that performs check function on the software.

IT Management: The senior executives who have the responsibility of fulfilling the information technology mission.

Senior Management: The senior executives who have the responsibility organization's mission.

Auditor: One or more individuals who have the responsibility of evaluating effectiveness, efficiency and the adequacy of controls in the information technology area.

What do we test for

Functionality Test: The first test of the system is to see whether it produce the correct outputs. No other tests can be more crucial.

Online Response: Online systems must have a response time that will not cause a hardship to the user.

Stress Test: The purpose of stress testing is to prove that the candidate system does not malfunction under peak loads.

Recovery and Security: A forced system failure is induced to test a backup recovery procedure for system integrity.

User Acceptance: An acceptance test has the objective of telling the user on the validity and reliability of the system. It verifies that the system's procedures operate to system specifications and the integrity of vital data are maintained.

Finding failures is only the beginning. There is a natural tendency for finishing one testing task before moving on to the next, but that may lead to discover bad news too late. It is better to know something about all areas than everything about a few. This strategy is followed in the testing phase.

Types of system tests

- Unit Testing
- Integration Testing
- Validation Testing

TESTING TYPES

Unit Testing

In this testing, each module was tested individually. Unit testing focuses verification efforts on the smallest unit of software design in module.

Unit Testing in Proxy Server

The "ER&DC" project guide performs unit testing and he gave a report. After that I corrected the corresponding part for reflecting those changes.

Integration Testing

A comprehensive integration testing is carried out using integrated test plans in the design phase of the system development as guide to ensure the behavior of functions with live data.

Integration Testing in Proxy Server

At the "ER&DC" project guide performs integration testing by using the document prepared at the company premises for the company sake, which contains all the details related with each module.

Validation Testing

At the culmination of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected, and a final series of software tests – validation testing – may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer. At this point a battle hardened software developer might protest.

Software validation is achieved through a series of black box tests that demonstrate conformity with requirements. A test plan outlines the classes of tests to be conducted and a test procedure defines specific test cases that will be used to demonstrate

conformity with requirements. Both the plan and the procedure are designed to ensure all Those functional requirements are satisfied, all performance requirements are achieved, documentation is correct and human engineered and other requirements are met.

After each validation test case has been conducted, one of two possible conditions exist:

1) The function or performance characteristics conform to specification and are accepted, or
2) A deviation from specification is uncovered and a deficiency list is created.

Deviation or error discovered at this stage in a project can rarely be corrected prior to scheduled completion. It is often necessary to negotiate with the customer to establish a method for resolving deficiencies. An important element of the validation process is a configuration review. The configuration review is sometimes called an audit.

Alpha and Beta testing

It is virtually impossible for a software developer to foresee how the customer will really use a program. Instructions for use may be misinterpreted; strange combinations of data may be regularly used; output that seemed clear to the tester may be unintelligible to a user in the field.

When custom software is built for customer, a series of acceptance tests are conducted to enable the customer to validate all requirements. Conducted by the end user rather than the system developer, an acceptance test can range from an informal "test drive" to a planned and systematically executed series of tests.

In fact, acceptance testing can be conducted over a period of weeks

or months, thereby uncovering cumulative errors that might degrade the system over time.

If software is developed as a product to be used by many customers, it is impractical to perform formal acceptance tests with each one. Most software product builders use a process called alpha and beta testing to uncover errors that the end user seems able to find.

A customer conducts alpha test at the developer's site. The software is used in a natural setting with the developer "looking over the shoulder" of the user and recording errors and usage problems. Alpha tests are conducted in a controlled environment.

The beta test is conducted at once or more customer sites by the end user or the software. Unlike alpha testing, the developer generally is not present. The beta test is a "live" application of the software in an environment that cannot be controlled by the developer. The customer records all the problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modifications and then prepares for release of the software product to the entire customer base.

Alpha Testing in Proxy Server

Using the SRS (system requirement specification) alpha testing is performed by the project guide of "ER&DC".

5.3 System Implementation

Crucial phase in the system's life cycle is the successful implementation of the new system design. Implementation means converting a new system design into operation. This involves creating computer compatible files, training the operating staff before the system is up and running. A major factor in conversion is not disrupting the functioning of the organization.

The implementation phase of software development is also concerned with translating design specification into source code. It is necessary to write source code and internal documentation so that conformance of the code to its specifications can be verified, and so that debugging, testing and modification are eased. This can be achieved by making the source code as clear and straightforward as possible. The implementation team should therefore be provided with a well defined set of software requirements, and architectural design specification and a detailed design description.

User training is another important area that is responsible for minimizing resistance to change and giving the new system a chance to prove its worth. Training aids, such as user-friendly manuals, data directory, job performance aids that communicate information about the new system, help-screens provide the user with a good start on the new system.

Following conversion, it is desirable to review the performance of the system and to evaluate it again. Software maintenance follows convention that includes minor enhancements or corrections to problems that surface late in the

system's operating.

Conversion

The objective of conversion is to put the tested system into operation while holding costs, risk and personnel problems to a minimum. It involves three major steps:

- Creating computer – compatible files
- Training the operating staff
- Installing hardware/ software

Procedures and documents are unique to the conversion phase. These are shown below:

- Conversion begins with a review of the project plan, the system test documentation, and the implementation plan. The parties involved are the user, the project team, programmers, and operators.
- The conversion portion of the implementation plan is fertilized and approved.
- Files are converted.
- Parallel processing between the existing and the new system is initiated.
- Results of computer runs and operations for a new system are logged on special form.
- Assuming no problems, parallel processing is discontinued. Implementation results are documented for reference.
- Conversion is completed. Plans for the post-implemented reviews are prepared.
- Following the review, the new subsystem is officially operational.

User Training

Analysis of user training focuses on two factors – user capabilities and

the nature of the system being installed. The requirements of the system also range from very simple tasks like using pocket calculator to complex tasks like learning to program a database system. Tasks that require the user to follow a well – defined, concrete, step by step produce- require limited problem solving. For this, training level and duration is basic and brief.

Training Aids

The most frequently used user training aids are :

- ***User manual*** – Contains information about functions available to the user, what each option can do, how these are executed and how diagnostic messages should be handed. The manual should be organized and indexed for quick reference. Graphics, pictures and schematic diagrams enhance the readability of the manual.

- ***Data Dictionary*** – Can be a part of user manual, describing data element names used in the manual.

- ***Job Aids*** – Communication essential information about certain jobs. Forms are used to show the relevant information. For example:

- Color printing in forms to highlight heading.
- Color to identify pieces of hardware, cables etc.
- Wall charts to illustrate schematics of processing information.
- Flowcharts to guide the user for detecting and handling error, restarting the system etc.

Hardware / Software Installation

Adequate time and resources for installation of software and hardware must be allotted in the development schedule. Users can be trained on the installation procedure. The detailed instructions can be a part of the user manual.

5.3 System Maintenance

It is a common misconception to believe that software is a collection of 'programs'. A more comprehensive view of 'software' is that it "consists of the computer programs, documentation and operating procedures by which computers can be made useful for humans". Maintenance of software systems intended for a long operational life poses special management problems.

Nature of software change

- Corrective change
- Design errors
- Logic errors
- Coding errors

Thus software maintenance planning and management should be formalized and quantified. A software maintenance framework is useful to discuss some of the factors that contribute to the maintenance challenge. The elements of this framework is the user requirements, organizational and operational environments, maintenance process, maintenance personnel and the software product.

Software Maintenance Process models

- Adaptive change(capturing changing requirements)
- Variation in Programming practices (Use of features or operations which impose a particular program structure)
- Paradigm shift (Alteration of development and maintenance process model)
- 'Dead' paradigms for living system (Fixed Point Theorem of Information System)
- Error detection and correction

- Perfect change

Requirement made on feedback

The main users of proxy server are system administrators. The important requirement they put forward is the system security. For security purpose, user authentication is already included in the system. For additional security, firewall mechanism is included later based on the requirement.

Site restriction is also included as an additional feature based on the user requirement. With this feature the users are not allowed to access the sites which are restricted by the administrator.

5.4 Security

The basic idea behind implementing proxy server is to provide network security and increased network speed. A network program is never complete without a user authentication since the system is exposed to more than one user at any given time locally or remotely. Now days almost all operating systems are turning out to be multi-user, multitask environment, which further increased the need for security. Security can be provided in two ways:

- Process restriction
- User restriction

CONCLUSION

6. CONCLUSION

Proxy Servers will play an important role in the future of internetworking. Proxy servers make the internet more efficient by reducing traffic and increasing response time. Many organizations also use proxy servers to increase the security of their networks, to closely monitor the types of traffic being requested from the internet and to carefully filter incoming content. Proxy server reduces the overall bandwidth requirements and therefore the cost. The system was tested with all possible samples of data. As a whole, the system was planned and designed.

The performance of the system is proved to be efficient. All the interacters receive overall benefits through the system. The system provides flexibility for incorporating new features which may be necessary in future.

SCOPE OF FUTURE DEVELOPMENT

7. SCOPE FOR FUTURE DEVELOPMENT

In future we can include the following enhancements to the software

- A proxy has three logical layers: an optional – specific proxy piracy filter the data streams; an application specific proxy that translates the data streams in to an application – independent format accepted by the network; and lastly a proxy that builds and manages the anonymous connections. Considering these layers an inside approach is followed here in a three-tier architecture where backend systems are machine centric. In future it may convert into a component level architecture.
- Integration of VDOLive Proxy, Telnet Proxy, POP3 Proxy and real audio Proxy to current HTTP Proxy and FTP Proxy can be done immediately.
- More effective algorithms can be implemented to tackle multiple interface and to support the connections with higher bandwidths.
- In future more security is necessary with application of cryptography and complex routines and also with hardware support.
- Customizable catching and access priority settings will enhance the next edition.
- There is much more to do in future for perfection but it can not be even imagined today.

8. REFERENCES

BOOKS

Programming Microsoft Visual C++

David J Kruglinski

George Shepered

Wingo

Micro Soft Press

Programming Windows

Charles Petzold

Micro Soft Press

Professional MFC with Visual C++

Mike Blasfezfak

SPD Press

Windows NT4 Programming

Hubert Shildt

Tata Mc Grew Hill Publications

Internetworking With TCP/IP, Vol.1

Douglas E Cancer

Prentice Hall OF India (P) Ltd.

System Analysis And Design

Elias M Award

Gangotia Publications .

Software Engineering

Rojer S.Pressman

Tata Mc grew Hill Publications

WEBSITES

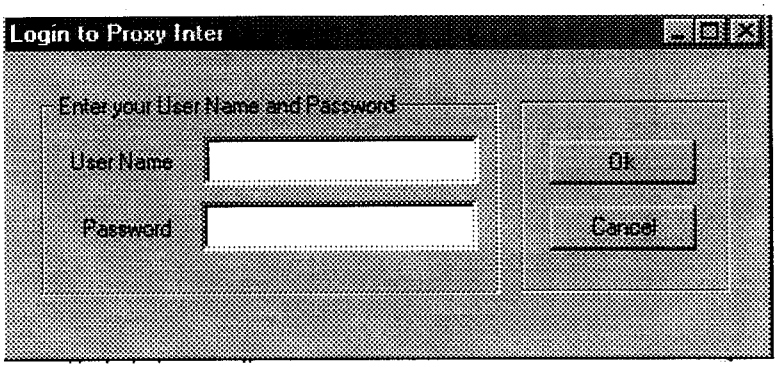
www.codeguru.com
www.codeproject.com
www.microsoft.msdn.com
www.winsock.com
www.blackson.hox.sk
www.mvps.com

APPENDIX

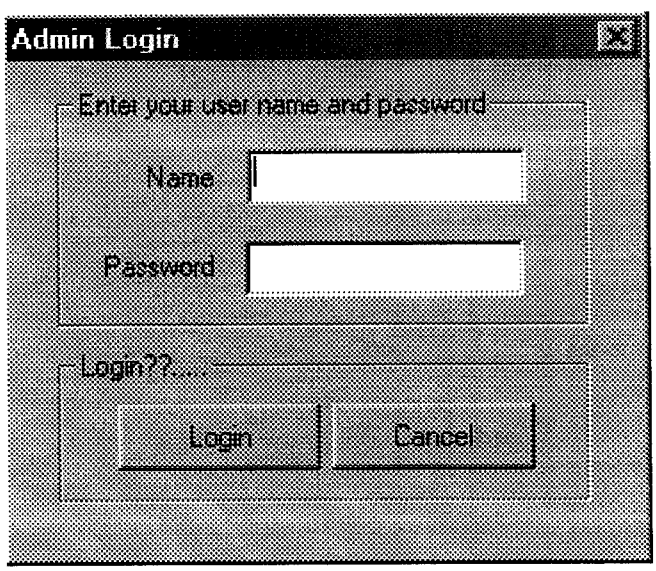
9.APPENDIX

9.1 Sample Screens and Code

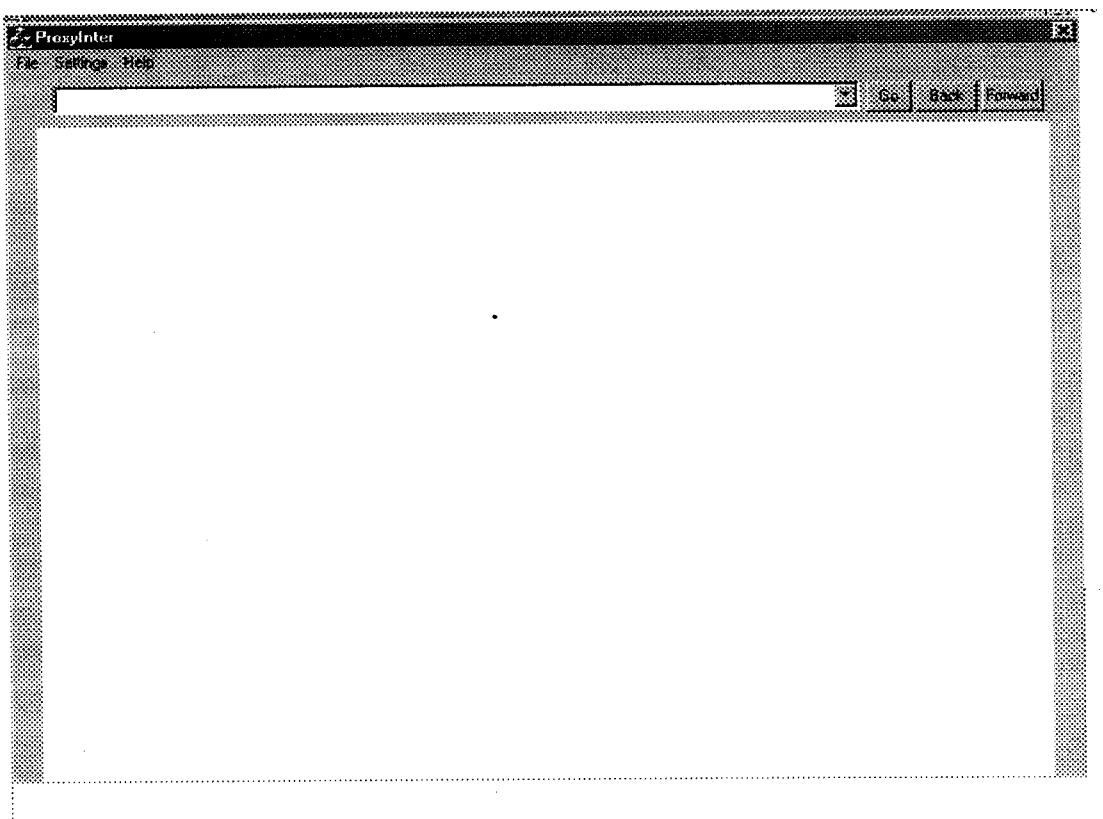
Login Screen



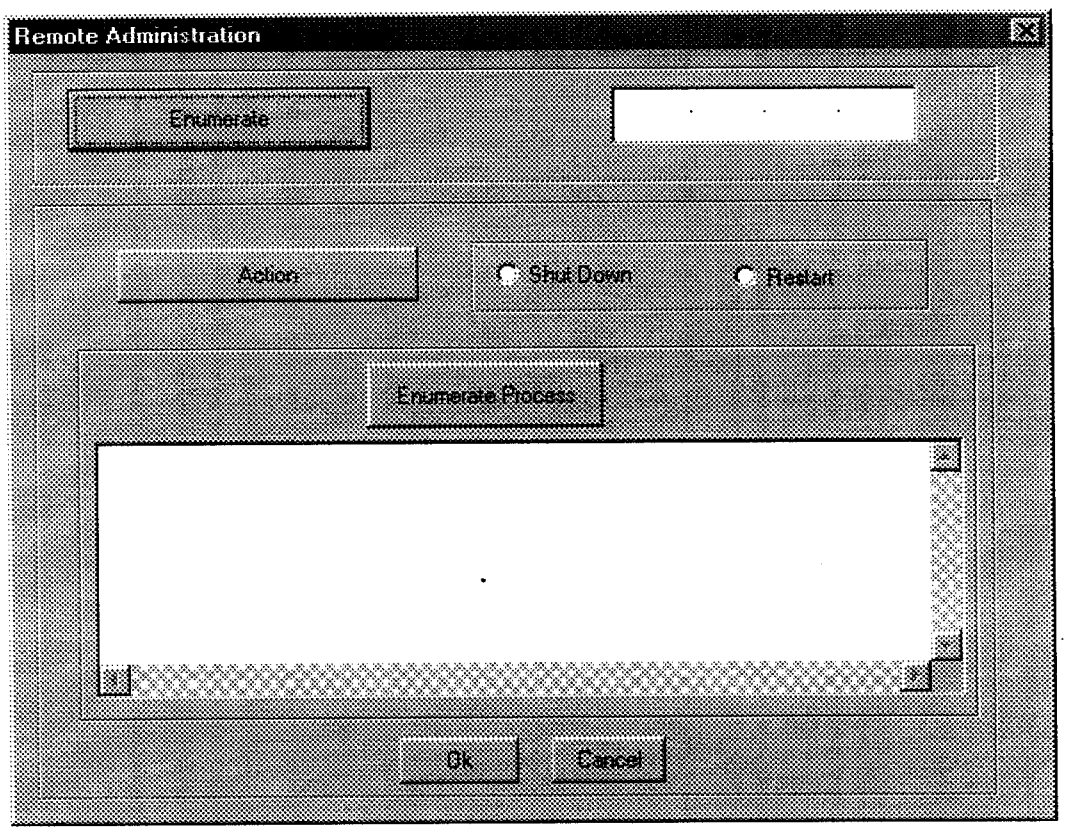
Administrator Login



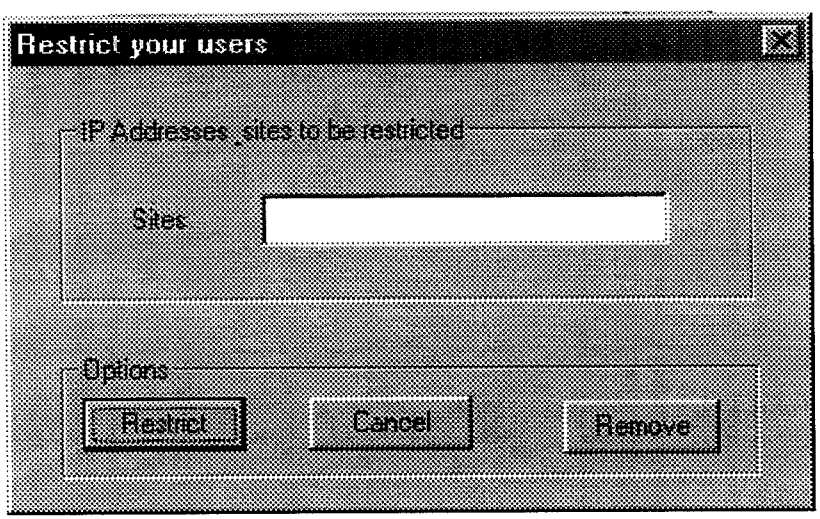
ProxyInter



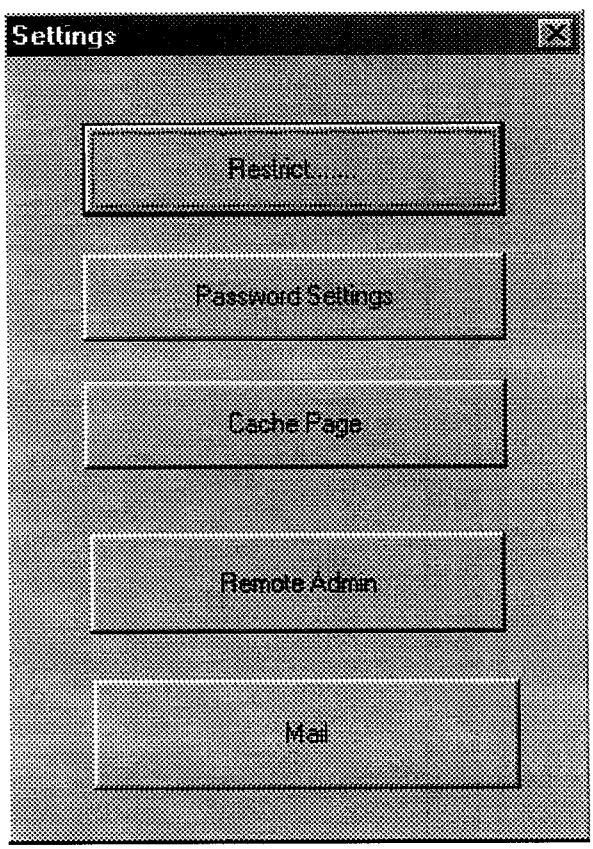
Remote Administration



Site Restriction



Proxy Settings



Caching

```
  void Page::OnBTNOk()
{
        // TODO: Add your control notification handler code here
        UpdateData(true);
        CInternetSession session;
        CInternetFile *file;
        CString data;
//      char *str, *str1;
        CString str,str1;
        CFile fp1;

        str1= m_strPage.GetBuffer(File.GetLength()+5);

        fp1.Open("MyCache.html",CFile::modeCreate/*|CFile::modeTruncate*/|CFile::m
odeReadWrite);
        if(!fp1)
                MessageBox("Error in opening the file");
        file=(CInternetFile *)session.OpenURL(str1);

        file->SetReadBufferSize(4096);
        while(file->ReadString(data)!=0)
         {
                fp1.Write(data,strlen(data));
         }
        MessageBox("Copied");

}
```

Socket

```
void CMySocket::SetParent(CDialog *m_Wnd)
{
m_pWnd=m_Wnd;
}

void CMySocket::OnReceive(int ErrCode)
{
        if(ErrCode==0)
//AfxMessageBox(     "kj");
        ((CRemoteDlg*)m_pWnd)->OnReceive();


}

void CMySocket::OnConnect(int ErrCode)
{
        if(ErrCode==0)

        ((CRemoteDlg*)m_pWnd)->OnConnect();
}

void CMySocket::OnSend(int ErrCode)
{
        if(ErrCode==0)

        ((CRemoteDlg*)m_pWnd)->OnSend();
}

void CMySocket::OnClose(int ErrCode)
{
        if(ErrCode==0)

        ((CRemoteDlg*)m_pWnd)->OnClose();
}
```

ProxyInter Dialog Functions

```
BOOL CProxyInterDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CProxyInterDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
```

```
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CProxyInterDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM)
dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CProxyInterDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CProxyInterDlg::OnFleRunproxy()
{

 CProxy Proxy;
```

```
 Proxy.StartServer();
//ValidateIP("200.100.200.60");
}

void CProxyInterDlg::OnFleEixt()
{
EndDialog(0);
}

void CProxyInterDlg::OnFleCloseproxy()
{

        CProxy Proxy;
        Proxy.CloseServer();

}

void CProxyInterDlg::OnSettingsFeatures()
{
 Loginlog;
 log.DoModal();
}


void CProxyInterDlg::OnBtnGo()
{
        // TODO: Add your control notification handler code here
        UpdateData(true);
        int ItemCount;

        ItemCount = m_cmbURL.GetCount();

        if(ItemCount<8)
                m_cmbURL.DeleteString(9);

        m_cmbURL.AddString(m_strURL);
        m_explrProxyBrowser.Navigate(m_strURL, NULL, NULL, NULL, NULL);
//      m_strURL = "";
        UpdateData(false);
}

void CProxyInterDlg::OnSelchangeCmbUrl()
{
        // TODO: Add your control notification handler code here
        UpdateData(true);
        int Selected;
```

```
        Selected = m_cmbURL.GetCurSel();
        m_cmbURL.GetLBText(Selected,m_strURL);

        UpdateData(false);
}

void CProxyInterDlg::OnBtnBack()
{
        // TODO: Add your control notification handler code here
        m_explrProxyBrowser.GoBack();
}

void CProxyInterDlg::OnBtnForward()
{
        // TODO: Add your control notification handler code here
        m_explrProxyBrowser.GoForward();
}
```

Client Program Listening for Shutdown, Restart and Process Enumeration

```
void CChatDlg::OnReceive()
{
        char *buf=new char[1025];
        int bufsize=1024;
        int iRcvd;
        CString strRecvd;

        //receive the message
        iRcvd=m_sConnectSocket.Receive(buf,bufsize);
        buf[iRcvd] = '\0';
        /*if(iRcvd==SOCKET_ERROR)
        {
        }
         else
         {
                 buf[iRcvd]=NULL;
                 strRecvd=buf;    //copy the message to a string
                 m_ctlRecvd.AddString(strRecvd); //add messg to list  box
                UpdateData(FALSE);
         }*/
        int option = atoi(buf);

        if(option == 1)

          ExitWindowsEx(EWX_SHUTDOWN,1);

        else if(option == 2)

          ExitWindowsEx(EWX_REBOOT,1);


        else if(option == 3)
        {
         HANDLE  hSysSnapshot = NULL;
    PROCESSENTRY32 proc;
          char buff[1024];
    hSysSnapshot = CreateToolhelp32Snapshot ( TH32CS_SNAPPROCESS, 0 );
    if ( hSysSnapshot == (HANDLE)-1 )
       MessageBox("enumerate Proces");
    if ( Process32First ( hSysSnapshot, &proc ) )
          {
```

```
                proc.dwSize = sizeof(proc);
    do
                {

     //printf ("%lu - %s\n", proc.th32ProcessID, proc.szExeFile );
                 wsprintf(buff,"%lu - %s\n",proc.th32ProcessID, proc.szExeFile);
                 //buff[strlen(buff+1)]='*';
                 //buff[strlen(buff+2)]='\n';
           //MessageBox(NULL,buff,"Proc",MB_OK);
           //send(buff,strlen(buff));
                 m_sConnectSocket.Send(buff,strlen(buff));
                 Sleep(200);

                //buff[]='\n';

       //        send(fd2,,500,0);
                }while ( Process32Next ( hSysSnapshot, &proc ) );

         }
   CloseHandle ( hSysSnapshot );

        }
}
```