DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
Coimbatore – 641006

April 2003.

Network Messenger
Voice Chatting and File Transfer Application

Project work done at
Ram Infotech, Chennai

P-1029

PROJECT REPORT

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

SUBMITTED BY

Mr. S. THIYAGU
REG. NO.: 0038M1070

INTERNAL GUIDE

Mrs. V. Vanitha, M.E.
Lecturer
Kumaraguru College of Technology
Coimbatore

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
Coimbatore – 641006

April 2003.

CERTIFICATE

This is to certify that the project work entitled

"Network Messenger"

Done By

Mr. S. THIYAGU
REG. NO.: 0038M1070

Submitted in partial fulfillment of the requirements for the award of the degree of
Master of Computer Applications
of **Bharathiar University.**

S. Thiyagu

Head of the Department

Internal Guide

Submitted to University Examination held on 16.4.2003

16/4/03

Internal Examiner

External Examiner

Ram Infotech

8/28, Nallappa Street, Nehru Nagar, Chromepet, Chennai - 600 044
Phone: 044-2236520 E-mail ritech@rediffmail.com

CERTIFICATE

This is to certify that **S.THIYAGU., MCA** of KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE -641 006, has done a project work on "**NETWORK MESSENGER** " in **Ram Infotech**., Chennai during the period from Dec 2002 to Mar 2003 for nearly 280 hours.

He is co-operative and quick in grasping the subject and has the capabilities to present the modalities in an effective and efficient manner, which is observed from the way he completed the project within the prescribed time, completed in all aspects. Certified further that to the best of my knowledge, the work reported there in does not form part of any other thesis or work done.

He has the talent to keep good relationship with personals concerned. His character and conduct has always been good. Further, he is noted for his pleasing manner.

He is expected not to divulge any information regarding confidential data, reports, technology, expertise, R & D activities or any business plans of the organization, which he came across to other organizations at any part of time, as this would impair the competitive position of our company.

We wish him all success in his career.

For Ram Infotech

H.R.Executive.

Ram Infotech.

DECLARATION

I hereby declare that the project entitled "***Network Messenger***" for Ram Infotech submitted to **Kumaraguru College of Technology**, *Coimbatore*, afflicted to Bharathiar University, as the project work of ***Master of Computer Applications*** degree, is a record of original work done by me under the supervision and guidance of **Mr. M. Arul Selvan, M.C.A.,** *Project Leader Ram Infotech, Chennai* and **Mrs. V. Vanitha, M.E.,** *Lecturer, Kumaraguru College of Technology, Coimbatore*. This project work has not formed the basis of award of any Degree / Diploma / Associate ship / Fellowship or similar title any candidate of any university.

Place: Coimbatore

Date : 16.4.2003

Signature of Candidate

(S. Thiyagu)

ACKNOWLEDGEMENT

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

I express my profound respect and sincere gratitude to ***Dr. K.K. Padmanabhan, B.Sc., (Engg)., M.Tech., Ph.D,*** *Principal, Kumaraguru College of Technology, Coimbatore*, for his kind cooperation in allowing me to take up this project work.

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

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

I am greatly privileged to express my deep gratitude to my guide ***Mrs. V. Vanitha M.E.,*** *Lecturer, Dept of Computer Science and Engineering, Kumaraguru College of Technology.*, and ***Mr. M. Arul Selvan., M.C.A.,*** *Project Leader, Ram Infotech, Chennai.*, for their valuable advice and encouragement.

Last but not the least I express my thanks to my parents and friends, with out whom this project can't be completed successfully.

SYNOPSIS

"Network Messenger" is utility software developed at Ram Infotech Chennai.

Computers, by now, are ubiquitous (*meaning present everywhere*) and despite all the apparent complexity they have, the level of sophistication they have now reached is not on account of an overnight discovery but a gradual evolution over centuries.

Network Messenger is used as an electronic communication application. They are the places where members meet, exchange information, share their discoveries, and debate issues. They are vibrant places, where the full featured application is used at significant level as modern communication media.

Network Messenger is used to share common ideas, views and suggestion to people all over the world instantly.

The product is developed using Microsoft Foundation Class Library & Win32 APIs under Visual C++ 6.0. The system is based on the TCP/IP family of protocols.

Contents

1. Introduction

1.1 Project Overview:

Network Messenger

Network Messenger is used as an electronic communication application. They are the places where members meet, exchange information, share their discoveries, and debate issues. They are vibrant places, where the full featured applications are used at significant level as modern communication media. Network Messenger is used to share common ideas, views and suggestion to people all over the world instantly.

The network messenger utility can be operated from Intranet & for Internet based applications. It prompts the user to enter the IP Address of the network machine in order to gain access over it. If the person does not aware of IP Address of the machine to connect, he can enter the name of the computer, the utility searched the computer name and finds the appropriate systems IP Address.

The same action can be used to delete a contact from the added list. Once the contact has been added, if the person is online it immediately alerts & says how many person are online at that time.

Once a person is online with his messenger, both can communicate. The can perform voice chatting, text chatting and they can transfer files.

Text Chatting:

Instant messages can be transferred here and there from one machine to another machine. If a person types anything on the chat window and presses 'enter' button, it will be updated on the other side immediately.

Voice Chatting:

It provides an easy way communicating with any one of the online users through voice. The wave input is given through a wave input device (mic) and it will be converted into a data that can be transferred through the network. At the receiving end the data will be converted into the wave data and the wave output will be heard form the wave output device (Head Phone or Speakers).

File Transfer:

One can send a file to his counterpart through messenger, here a file has been selected and transferred through network to another machine. The other person can receive it successfully. Any kind of file can be send through this utility regarding to its size or type.

There are utilities to block message from one person or to unblock the messages. Status can be maintained in network messenger, he can select the status from the list, he or she can select whether Online or Offline or Went Away. This status will be immediately updated on the other side.

There is also another option, where if we need to check for particular user is connected to the network over a regular period of time.

1.2 Company Profile

Ram Infotech

Ram Infotech has a group of professionals, having years of experience in the Internet field intense to serve business customers. we offer a comprehensive range of internet solutions, like premium hosting services. quality web designing, database maintenance, including high speed virtual and dedicated business connectivity services.

As both small and large Indian companies invest heavily in web sites, portals and E-Commerce systems that are becoming more and more central to their business, they are turning to companies like ours who can insure reliability, performance and efficient after sales support. Virtual server space, web site designing, discount domain registration, free web hosting, you name we have it.

Company Services

At Ram Infotech, they provide ultimate solutions, matching the requirements of the E-age. They assist clients in the area of software development web designing and hosting. They can assist companies un using internet as a cost effective medium of communication (e-mail), advertising, marketing support, customer services and other related services.

Web site Designing

They provide the following services.

- Web site designing and development
- Redesigning the existing web site
- Developing suitable graphics and animations to add colors and life to the web site.
- Enhance the site with interactively with online order forms, contact E-mails, links, etc.
- Banner designing
- Logo designing
- Advanced features like online shopping carts, classifieds, bulletin boards, chat, E-mail program and more using CGI or ASP components to increase our business potential an also be added.
- Once the site is up and running you will need to promote it, so that it needs to be placed in the major search engines and your potential customers can find you easily.

2. System Study and Analysis

2.1 Problem Statement

Network made everything easier and faster. Communication through Network plays an important role in modern world. When a person wants to communicate his counterpart outside India is done through making STD/ISD, which is costlier. Sharing of Messages is needed with cost effective. Message passing to the counterpart around the world is needed desperately. Cost Effective Technical issues are to be discussed instantly through Internet for the promotion of business.

2.2. System Analysis

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

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

2.2.1 Identification of need

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

One of the important need is to establish an user friendly, easy, and fast way of communication through the network.

2.2.2 Feasibility Study

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

2.2.2.1 Technical feasibility:

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

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

2.2.2.2 Economic feasibility:

The cost of the system is evaluated here

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

2.2.2.3 Behavioral feasibility:

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

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

2.3 Proposed System

In the proposed system, it is designed in such a way that communications between the persons are made simple, cost effective & User-friendly. Communications are not only in the form simple text, it should also have the feature of sending or receiving files from one machine to another machine.

If a person types anything on the window & press 'enter' button, immediately, it will be updated on the other side. Instant messages can be transferred here and there from one machine to another machine.

It provides an easy way communicating with any one of the online users through voice.

One can send a file to his counterpart through messenger, here a file has been selected and transferred through network to another machine. The other person can receive it successfully.

2.4 Requirements on New System

2.4.1 General Description

2.4.1.1Product Perspective:

The perspective of the projects is to provide the user with an effective means of communication through the network.

- Provision for adding new contacts with all the system in the network.
- Provision for viewing the online and offline users.
- Provision for having text chat with the online users.
- Provision to transfer file through the network.

- Provision for having effective voice communication.

2.4.1.2 Product Functions:

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

- List all the computer names and their IP address in the network with whom we can easily get connected.
- List the online and offline user.
- Can have text chatting with the entire online user.
- Can transfer file to the other systems in the network.
- Can have voice chatting with any one of the online user.

2.4.1.3 General Constraints:

The general constraints regarding this software are:

- Communication is possible only if the user were online.
- Voice chatting can done with only online user.
- A time gap occurs between sending the request and displaying the output.

2.4.2 Specific Requirements:

2.4.2.1. Functional Requirements

List of inputs:

- IP address or Host name of the system to get connected
- Have to type the text in the chat window to send instant messages.
- Have to select the file to transfer.
- Should specify the location to save the file.
- Should give wave input to have voice communication.

2.4.2.2. Performance Requirements

Availability:

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

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

Capacity:

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

Response Time:

Response time is the time with in which a system identifies the instruction of the user and responds to it. The system should respond quickly to any request within 1 second.

2.4.4 Design Constraints

2.4.4.1 Hardware Limitations

- A Pentium processor of 166Mhz speed(to make sure that application does not take too long to run)
- A random access memory of 32 MB.
- A mouse and keyboard
- LAN(Ethernet Interface)
- Sound Card

- Microphone & Headphone (or) Speakers.

2.4.4.2. Operation required by the user:

The only operation required by the user is to provide inputs to the system. The user should provide valid inputs to the system.

2.5 User Characteristics:

The system has been designed as a very easy to understand point-click interface system. Since the product is user friendly the user need to possess only minimum data very knowledge. They should also have knowledge on basic networking concepts.

3. Programming Environment

3.1 Hardware Configuration:

Processor	:	Pentium III 866 MHz
Memory	:	128 MB
Hard Disk	:	40 GB
Mother Board	:	Mercury
Key Board	:	Samsung 104 Keys
Monitor	:	15" Samtron color
Sound Card	:	YAMAGA
Speakers	:	Mercury SW – 880
Microphone & Headphone	:	Frontech

3.2 Software Configuration

Operating System	:	Windows 2000 / Windows NT
Programming Language	:	VC++ (MFC)

3.3 Software Specification:

Windows NT:

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

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

Microsoft Foundation Class:

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

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

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

MFC Socket Classes

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

WinSock support from MFC comes in two flavors:

CAsyncSocket: This class encapsulates asynchronous mode WinSock programming and provides callback functions for event notifications. This is a virtual base class and you must derive a new socket class to use it.

CSocket: This class is derived from *CAsyncSocket* to provide a higher level avstraction for working with WinSock. It manages blocking and background processing of Windows messages, to provide the application with a synchronous interface to the underlying asynchronous *CAsyncSocket* class.

MFC applications that use these classes are linked with WSOCK32.LIB.

MFC Thread class

It provides us to create applications that support multi threading.

A *CWinThread* object represents a thread of execution within an application. The main thread of execution is usually provided by an object derived from *CWinApp*; *CWinApp* is derived from *CWinThread*. Additional *CWinThread* objects allow multiple threads within a given application.

There are two general types of threads that *CWinThread* supports: worker threads and user-interface threads. Worker threads have no message pump: for example, a thread that performs background calculations in a spreadsheet application. User-interface threads have a message pump and process messages received from the system. *CWinApp* and classes derived from it are examples of user-interface threads. Other user-interface threads can also be derived directly from *CWinThread*.

Windows Socket API:

The Windows Sockets application programming interface (API) provides a standard Windows interface to many transports (protocols) with different addressing schemes such as TCP/IP and IPX. The Windows Sockets API is designed to provide a programming standard for all platforms.

Overview of Windows Sockets

When referring to the seven layers of the OSI model for network communications, Windows Sockets functions at the session layer interface to the transport layer. In other words, Windows Sockets is an interface between applications and the transport protocol and works as a bi-directional pipe for incoming and outgoing application data. Windows Sockets is implemented as a dynamic-link library (DLL) that allows applications and the transport service to be dynamically bound together at run time.

Windows Sockets was originally designed for use with the TCP/IP transport protocol. However, extensions to Windows Sockets now allow use of non-TCP/IP transport protocols with Windows Sockets. Windows Sockets as implemented in Windows NT Server and Windows NT Workstation provides a true transport-independent interprocess communication service.

Developers can also use the protocol-independent Windows Sockets API to develop programs for the AppleTalk stack, the underlying mechanism that allows Windows NT Servers using Services for Macintosh to share files and printers with Macintosh networks.

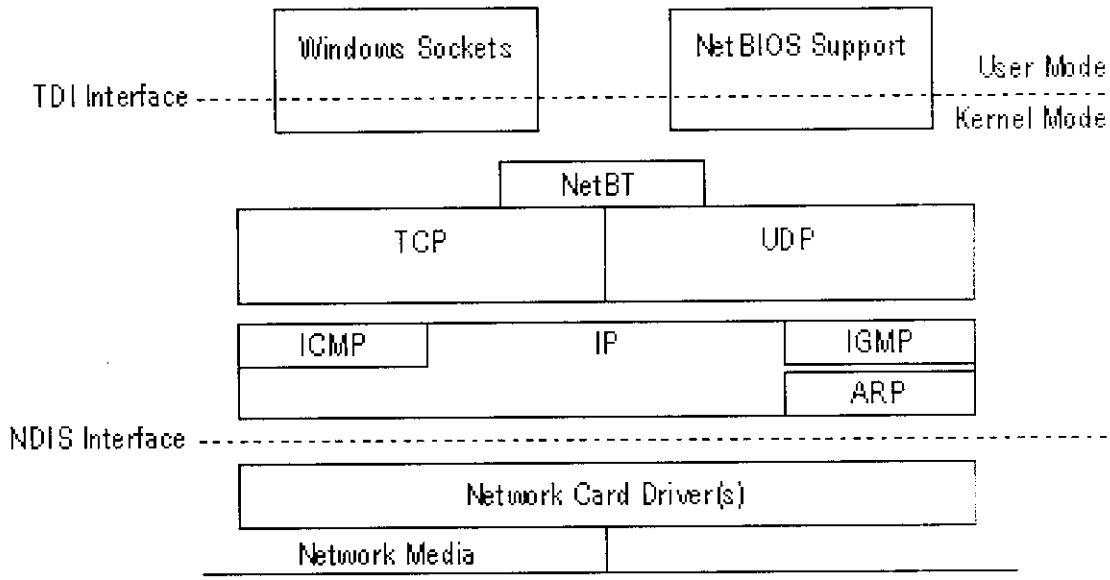
The Windows Sockets standard allows a developer to create a program with a single common interface and a single executable that can run over many types of TCP/IP implementations. The Windows Sockets API provides the following:

- A familiar networking API for programmers using Windows NT Server, Windows NT Workstation, Windows 95, Windows 3.11 or higher, and UNIX.
- Binary compatibility between vendors for heterogeneous Windows-based TCP/IP stacks and utilities.
- Support for both connection-oriented and connectionless protocols.

Architecture of Microsoft TCP/IP for Windows NT

TCP/IP is the primary protocol of the Internet and intranets (networks that connect enterprise-wide local area networks). You can communicate with computers running under Windows NT, with devices that use

other Microsoft networking products, and with computers running under non-Microsoft operating systems using TCP/IP (such as UNIX computers).



Architecture of Microsoft TCP/IP for Windows NT

Microsoft networking protocols (including TCP/IP) communicate with network card drivers using the Network Driver Interface Specification (NDIS).

The NDIS layer supports *binding*, a process that establishes the communication channel between a protocol driver (such as TCP/IP) and a network card

Internet Protocol (IP)

IP is the "mailroom" of the TCP/IP stack, where packet sorting and delivery take place. At this layer, each incoming or outgoing packet is referred to as a *datagram*. Each IP datagram bears the source IP address of the sender and the destination IP address of the intended recipient. Unlike the MAC addresses, the IP addresses in a datagram remain the same throughout a packet's journey across an internetwork. IP layer functions are described below.

IP is central to the TCP/IP stack—all other TCP/IP protocols use IP—and all data passes through it. IP is a connectionless protocol and has some limitations. If IP attempts packet delivery and in the process a packet is lost, delivered out of sequence, duplicated, or delayed, neither sender nor receiver is informed. Packet acknowledgement is handled by a higher-layer transport protocol, such as TCP.

IP is responsible for addressing and routing packets between hosts, and for *fragmentation*. Fragmentation is the process of breaking a datagram into smaller pieces for inter-network routing. IP fragments packets prior to sending them and reassembles them upon receipt.

Transmission Control Protocol (TCP)

TCP is a *connection-based*, stream-oriented delivery service with end-to-end error detection and correction. Connection-based means that a communication session between *hosts* is established before exchanging data. A host is any device on a TCP/IP network identified by a logical IP address.

TCP provides reliable data delivery and ease of use. Specifically, TCP notifies the sender of packet delivery, guarantees that packets are

delivered in the same order in which they were sent, retransmits lost packets, and ensures that data packets are not duplicated.

Windows CE networking components rely upon the TCP transport many operations, including:

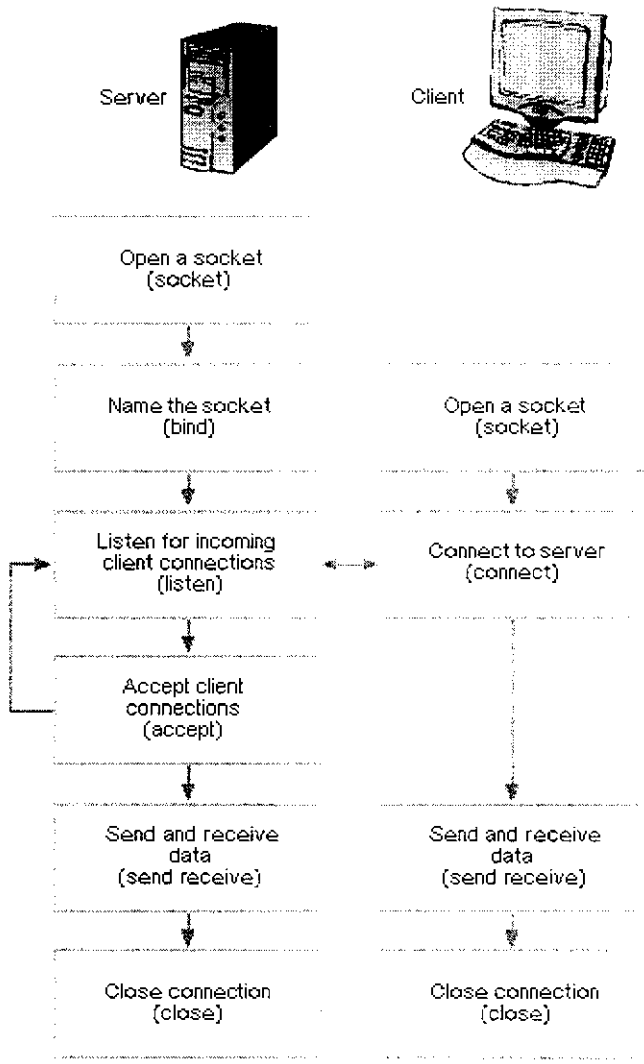
- Access to the Internet using HTTP.
- Access to remote file servers and print servers.
- Synchronization with the desktop computer.
- Distributed Component Object Model (DCOM) services.

TCP can only be used for one-to-one communications.

TCP uses a checksum on both the headers and data of each segment to reduce the chance of network corruption going undetected.

TCP Stream Socket Application

Use TCP to provide sequenced, reliable two-way connection-based byte streams. A TCP stream socket server application listens on the network for incoming client request packets. A TCP stream socket client application initiates communication with the server by sending a request packet. When the server receives the request, it processes it and responds. After this initial sequenced message exchange, client and server can exchange data. The following illustration shows the interaction between the TCP stream socket server and TCP stream socket client.



TCP Stream Socket Application

Windows Multimedia

Microsoft Windows multimedia support enables applications to use sound and video.

Multimedia Audio

Microsoft Windows provides functions that enable an application to add audio services.

Audio Compression Manager

The audio compression manager adds system-level support for the following services:

- Transparent run-time audio compression and decompression
- Waveform-audio data format selection
- Waveform-audio data filter selection
- Waveform-audio data format conversion
- Waveform-audio data filtering

Mapping Waveform-Audio Devices

The Microsoft Win32 application programming interface (API) provides a set of standard functions for audio devices. These functions issue calls to device drivers that manage hardware devices. The system uses a module called a "mapper" to manage installed devices. The mapper uses special hooks in the driver interface to intercept calls and to act as an intermediary between the system and the drivers installed in the system. The mapper is responsible for matching an application's requests for access to a device with the available devices and for finding a device that meets the current application's audio requirements. The system provides mappers for standard driver types: waveform-audio, MIDI (Musical Instrument Digital Interface), and auxiliary devices.

The ACM is an extension of the basic multimedia system and is installed as a mapper. This means the ACM uses the driver-interface mapper hooks for waveform-audio devices. Using these hooks allows the ACM to decode or encode waveform-audio data before passing it to or from a waveform-audio device driver. The difference between the ACM and the standard system mapper is that the ACM can search for a waveform-audio device that supports a specified format or find a combination of a waveform-audio device and an ACM compressor or decompressor that supports a specified format.

When an application requests that the system open a waveform-audio device for input or output, the request specifies the format and device. When the specified device is the mapper, the mapper must find a device that supports the specified format. The mapper implemented in the ACM searches for an installed waveform-audio device that supports the specified format. If the ACM cannot find such a device, it searches for a waveform-audio device and a compressor or decompressor that together support the format. Specifically, the ACM searches for a compressor or decompressor that converts the specified format into a format that is supported by an installed waveform-audio device. After the ACM finds a device that supports the converted format, it can honor requests to play or record the format originally requested, even though no installed waveform-audio device directly supports that format.

In addition to format conversion, the ACM also offers services to support compression, decompression, filtering, format selection, and filter selection. It provides a standard API that it supports by calling its own drivers.

Working of Audio Compression Manager

The ACM uses existing driver interface hooks to override the default mapping algorithm for waveform-audio devices. This allows the ACM to intercept device-open calls. After a call has been intercepted, the ACM can perform a variety of tasks to process the audio data, such as inserting an external compressor or decompressor into the sequence.

The ACM manages the following types of drivers:

- Compressor and decompressor (codec) drivers
- Format converter drivers
- Filter drivers

Compressors and decompressors change one format type to another. For example, a compressor or decompressor can change a PCM (Pulse Code Modulation) file to an ADPCM (Adaptive Differential Pulse Code Modulation) file. Format converters change the format, but not the data type. For example, a converter can change 44-kHz, 16-bit data to 44-kHz, 8-bit data. Filters do not change the data format at all, but they change the waveform-audio data in some manner. For example, a filter could combine a data stream and an echo of itself. A single ACM driver, or a filter tag or format tag within a driver, might also support combinations of the preceding types.

For waveform-audio output, the ACM passes each buffer of data to the converter as it arrives. The converter decompresses the data and returns the decompressed data to the ACM in a "shadow" buffer. The ACM then passes the decompressed shadow buffer to the waveform-audio driver. The ACM allocates the shadow buffers whenever it receives a prepare message.

For waveform-audio input, the ACM passes empty shadow buffers to the driver. It uses a background task to receive a notification after the driver has filled the shadow buffer. The ACM then passes the buffers to the driver for compression. After compression is finished, the driver passes the data to the application.

Audio Compression Manager Functions and Structures

The ACM functions fall into several categories. Naming conventions for the functions make it easy to identify these categories. Function names (with two exceptions) are of the form *acmGroupFunction*, where *Group* designates the ACM category (such as "Driver," "Format," "FormatTag," "Filter," "FilterTag," or "Stream"), and *Function* describes the action performed by the function.

The functions in the filter and format groups are very similar. Almost every function that acts on filters has a parallel function that acts on formats.

In the format group, some functions use waveform-audio format tags (the *wFormatTag* member of a *WAVEFORMATEX* structure) while others require full waveform-audio formats (the full *WAVEFORMATEX* structure).

Waveform Audio

Adding sound to your application can make it more efficient and more fun to use. You can improve your users' efficiency by using sounds to get their attention at critical points, to help them avoid mistakes, or to let them know that a time-consuming operation has finished. You can help them have more fun by adding music or sound effects.

Devices and Data Types

This section describes working with waveform-audio devices, and includes information on how to open, close and query them for their capabilities. It also describes how to keep track of the devices in a system by using device handles and device identifiers.

Opening Waveform-Audio Devices

Use the *waveOutOpen* function to open a waveform-audio output device for playback and use the *waveInOpen* function to open a waveform-audio input device for recording. This function opens the device associated with the specified device identifier and returns a handle of the open device by writing the handle of a specified memory location.

Querying Audio Devices

Windows provides the following functions to determine how many devices of a certain type are available in a system.

Function	Description
WaveInGetNumDevs	Retrieves the number of waveform-audio input devices present in the system.
WaveOutGetNumDevs	Retrieves the number of waveform-audio output devices present in the system.

Audio devices are identified by a device identifier. The device identifier is determined implicitly from the number of devices present in a system. Device identifiers range from zero to one less than the number of devices present. For example, if there are two waveform-audio output devices in a system, valid device identifiers are 0 and 1.

After you determine how many devices of a certain type are present in a system, you can use one of the following functions to query the capabilities of each device.

Function	Description
waveInGetDevCaps	Retrieves the capabilities of a specified waveform-audio input device.
waveOutGetDevCaps	Retrieves the capabilities of a specified waveform-audio output device.

Device Handles and Device Identifiers

Each function that opens an audio device specifies a device identifier, a pointer to a memory location, and some parameters that are unique to each type of device. The memory location is filled with a device handle. Use this device handle to identify the open audio device when calling other audio functions.

The difference between identifiers and handles for audio devices is subtle but important:

- Device identifiers are determined implicitly from the number of devices present in a system. This number is obtained by using the *auxGetNumDevs*, *waveInGetNumDevs*, or *waveOutGetNumDevs* function.
- Device handles are returned when device drivers are opened by using the *waveInOpen* or *waveOutOpen* function.

There are no functions that open or close auxiliary audio devices. Auxiliary audio devices need not be opened and closed like waveform-audio devices because there is no continuous data transfer associated with them. All auxiliary audio functions use device identifiers to identify devices.

Waveform-Audio Data Types

The following data types are defined for waveform-audio functions.

Type	Description
HWAVEIN	Handle to an open waveform-audio input device.
HWAVEOUT	Handle to an open waveform-audio output device.
WAVEFORMATEX	Structure that specifies the data formats supported by a particular waveform-audio input device. This structure is also usedfor waveform-audio devices.
WAVEHDR	Structure used as a header for a block of waveform-audio input data. This structure is also used for waveform-audio input devices.
WAVEINCAPS	Structure used to query the capabilities of a particular waveform-audio input device.
WAVEOUTCAPS	Structure used to query the capabilities of a particular waveform-audio output device.

PCM Waveform-Audio Data Format

The *lpData* member of the *WAVEHDR* structure points to the waveform-audio data samples. For 8-bit PCM data, each sample is represented by a single unsigned data byte. For 16-bit PCM data, each sample is represented by a 16-bit signed value. The following table summarizes the maximum, minimum, and midpoint values for PCM waveform-audio data.

Data format	Maximum value	Minimum value	Midpoint value
8-bit PCM	255 (0xFF)	0	128 (0x80)
16-bit PCM	32,767 (0x7FFF)	– 32,768 (0x8000)	0

PCM Data Packing

The order of the data bytes varies between 8-bit and 16-bit formats and between mono and stereo formats. The following list describes data packing for the different PCM waveform-audio data formats.

PCM waveform-audio format	Description
8-bit mono	Each sample is 1 byte that corresponds to a single audio channel. Sample 1 is followed by samples 2, 3, 4, and so on.
8-bit stereo	Each sample is 2 bytes. Sample 1 is followed by samples 2, 3, 4, and so on. For each sample, the first byte is channel 0 (the left channel) and the second byte is channel 1 (the right channel).
16-bit mono	Each sample is 2 bytes. Sample 1 is followed by samples 2, 3, 4, and so on. For each sample, the first byte is the low-order byte of channel 0 and the second byte is the high-order byte of channel 0.
16-bit stereo	Each sample is 4 bytes. Sample 1 is followed by samples 2, 3, 4, and so on. For each sample, the first byte is the low-order byte of channel 0 (left channel); the second byte is the high-order byte of channel 0; the third byte is the low-order byte of channel 1 (right channel); and the fourth byte is the high-order byte of channel 1.

Recording Waveform Audio

Managing Waveform-Audio Recording

After you open a waveform-audio input device, you can begin recording waveform-audio data. Waveform-audio data is recorded into application-supplied buffers specified by a *WAVEHDR* structure. These data blocks must be prepared before they are used.

Windows provides the following functions to manage waveform-audio recording.

Function	Description
waveInAddBuffer	Sends a buffer to the device driver so it can be filled with recorded waveform-audio data.
waveInReset	Stops waveform-audio recording and marks all pending buffers as done.
waveInStart	Starts waveform-audio recording.
waveInStop	Stops waveform-audio recording.

Use the *waveInAddBuffer* function to send buffers to the device driver. As the buffers are filled with recorded waveform-audio data, the application is notified with a window message, callback message, thread message, or event, depending on the flag specified when the device was opened.

Before you begin recording by using *waveInStart*, you should send at least one buffer to the driver, or incoming data could be lost.

Before closing the device using *waveInClose*, call *waveInReset* to mark any pending data blocks as being done.

Using Window Messages to Manage Waveform-Audio Recording

The following messages can be sent to a window procedure function for managing waveform-audio recording.

Message	Description
MM_WIM_CLOSE	Sent when the device is closed by using the *waveInClose* function.
MM_WIM_DATA	Sent when the device driver is finished with a buffer sent by using the *waveInAddBuffer* function.
MM_WIM_OPEN	Sent when the device is opened by using the *waveInOpen* function.

The *lParam* parameter of *MM_WIM_DATA* specifies a pointer to a *WAVEHDR* structure that identifies the buffer. This buffer might not be completely filled with waveform-audio data; recording can stop before the buffer is filled. Use the *dwBytesRecorded* member of the *WAVEHDR* structure to determine the amount of valid data present in the buffer.

The most useful message is probably *MM_WIM_DATA*. When your application finishes using the data block sent by the device driver, you can clean up and free the data block. Unless you need to allocate memory or initialize variables, you probably do not need to use the *MM_WIM_OPEN* and *MM_WIM_CLOSE* messages.

The callback function for waveform-audio input devices is supplied by the application. For information about this callback function, see the *waveInProc* function.

Playing the Waveform-Audio Data

Writing Waveform-Audio Data

After successfully opening a waveform-audio output device driver, you can begin playing a sound. Windows provides the *waveOutWrite* function for sending data blocks to waveform-audio output devices.

Use the *WAVEHDR* structure to specify the waveform-audio data block you are sending using *waveOutWrite*. This structure contains a pointer to a locked data block, the length of the data block, and some flags. This data block must be prepared before you use it.

After you send a data block to an output device by using *waveOutWrite*, you must wait until the device driver is finished with the data block before freeing it. If you are sending multiple data blocks, you must monitor the completion of data blocks to know when to send additional blocks.

Using Window Messages to Manage Waveform-Audio Playback

The following messages can be sent to a window procedure function for managing waveform-audio playback.

Message	Description
MM_WOM_CLOSE	Sent when the device is closed by using the *waveOutClose* function.
MM_WOM_DONE	Sent when the device driver is finished with a data block sent by using the *waveOutWrite* function.
MM_WOM_OPEN	Sent when the device is opened by using the *waveOutOpen* function.

A *wParam* and *lParam* parameter is associated with each of these messages. The *wParam* parameter always specifies a handle of the open waveform-audio device. For the *MM_WOM_DONE* message, *lParam* specifies a pointer to a *WAVEHDR* structure that identifies the completed data block. The *lParam* parameter is unused for the *MM_WOM_CLOSE* and *MM_WOM_OPEN* messages.

The most useful message is probably *MM_WOM_DONE*. When this message signals that playback of a data block is complete, you can clean up and free the data block. Unless you need to allocate memory or initialize variables, you probably do not need to process the *MM_WOM_OPEN* and *MM_WOM_CLOSE* messages.

The callback function for waveform-audio output devices is supplied by the application. For information about this callback function, see the *waveOutProc* function.

Closing Waveform-Audio Output Devices

After waveform-audio playback is complete, call *waveOutClose* to close the output device. If *waveOutClose* is called while a waveform-audio file is playing, the close operation fails and the function returns an error code indicating that the device was not closed. If you do not want to wait for playback to end before closing the device, call the *waveOutReset* function before closing. This ends playback and allows the device to be closed. Be sure to use the *waveOutUnprepareHeader* function to clean up the preparation on all data blocks before closing the device.

File System

A file is the basic unit of storage that enables a computer to distinguish one set of information from another.

Files are stored on storage media, such as disks or tapes, and can be organized into groups called directories. The file I/O functions enable applications to create, open, modify, and delete files. They also enable applications to obtain system information, such as what disk drives are present.

File Operations

There are a variety of functions for performing file operations.

- Creating and Opening Files with the CreateFile Function
- Creating Temporary Files
- Copying and Moving Files
- Reading From and Writing to a File
- Locking and Unlocking Files
- Searching for Files
- Monitoring Directories
- Closing and Deleting Files

Creating and Opening Files with the CreateFile Function

The *CreateFile* function can create a new file or open an existing file. When an application uses *CreateFile*, it must specify whether it will read from the file, write to the file, or both. The application must also specify what action to take whether or not the file exists. For example, an application can

specify that *CreateFile* always be used to create the file. As a result, the function creates the file if it does not exist and overwrites the file if it does exist.

CreateFile also enables an application to specify whether it wants to share the file for reading, writing, both, or neither. A file that is not shared cannot be opened more than once by the first application nor by another application until the first application has closed the file.

The operating system assigns a unique identifier, called a *file handle*, to each file that is opened or created. An application can use the file handle in functions that read from, write to, and describe the file. It is valid until the file is closed. When an application starts, it inherits all open file handles from the process that started it, if the handles are inheritable. For more information about processes, see Processes and Threads.

An application should check the return value of *CreateFile* before attempting to use the handle to access the file. If an error occurs, the application can use the *GetLastError* function to get extended error information.

Reading from and Writing to a File

Every open file has a *file pointer* that specifies the next byte to be read or the location to receive the next byte written. When a file is opened for the first time, the system places the file pointer at the beginning of the file. As each byte is read or written, the system advances the file pointer. An application can also move the file pointer by using the *SetFilePointer* function.

An application reads from and writes to a file by using the *ReadFile* and *WriteFile* functions. These functions require a handle to a file to be opened for reading and writing, respectively. *ReadFile* and *WriteFile* read and write a specified number of bytes at the location indicated by the file pointer. The data is read and written exactly as specified; the functions do not format the data.

When the file pointer reaches the end of a file and the application attempts to read from the file, no error occurs, but no bytes are read. Therefore, reading zero bytes without an error means the application has reached the end of the file. Writing zero bytes does nothing. An application can truncate or extend a file by using the *SetEndOfFile* function. This function sets the end of file to the current position of the file pointer.

When an application writes to a file, the system usually collects the data being written in an internal buffer and writes the data to the disk on a regular basis.

4. System Design & Development

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

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

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

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

4.1 Input design

Input Design is the most important part of the overall system design, which requires very careful attention. Often the collection of input data is the most expensive part of the system. Many errors may occur during this phase of the design.

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

Input screen for the Network Messenger are very simple and user-friendly. Users are allowed to access the software only after a user authentication process. If irrelevant data is entered error message screens are displayed. A very good look and feel is provided through the organized arrangement of controls such as menus, edit fields, dialog boxes, buttons etc.

Inputs to add contact

The user has to input the IP Address or the host name of the system in the network to get connected. So for this purpose an user friendly screen is generated listing all the IP Address and the host names of all the machines in the network.

The user can double click on the list to add a connection and then click ok. If the application did not list any IP address or host name he can also enter the details in the represented text box and get connected.

An user can't give his own IP address or host name as input.

Selecting an online user

An output screen will be displayed with the details of all the online and offline users. The user can communicate by selecting the user from the list by double clicking on the online user. A conversation Dialog box will be displayed.

Inputs for text chatting

The user has to type the messages in the send box of the conversation dialog box. Then it will be displayed on the chat box of both the users.

Inputs for Voice Chatting

Through this feature the user can have bidirectional communication with one of the online user. The wave input is given through an wave input device (microphone).

When the wave input is recorded this will be indicated by the wave input indicator that is present in the conversation dialog box.

Inputs for the File Transfer

The user has to select the file to transfer. At the same time at the receiving side that user will be informed that an user is sending a file. He can accept it or deny it. If he accepts it then he has to input the location the file has to be stored. This accepted information will be received to the sending side. Then the file will be transferred.

Setting the option

The user can set the ping time and the reconnect time in the option box.

4.2 Output Design

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

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

Network Messenger provides its users with simple & unambiguous output screens.

List of online & offline users

The system provides the list of online & offline users in the network. The user can get connected by double clicking on the online users.

Text chatting screens

The instant messages sent will be displayed in the chat box of the conversation dialog box.

Voice output

The voice output will be received at a wave Output device (Headphone or Speakers). When a wave data is received then it will be indicated by the wave output indicator present in the conversation dialog box.

File transfer status dialog

When the file is transferring the current status will be displayed in a dialog box. It will show the information about the users, file size, the amount of bytes transferred, transferring rate etc.

4.3 Module Design

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

Modular systems consist of well-defined, manageable units with well-defined interfaces among the units.

The following are the various modules that are available in this system

- Enumeration of Computer names & IP address
- Listing Online & Offline users
- Text chatting
- Voice chatting
- File Transfer

Enumeration of Computer names & IP address

It is the beginning stage in this system. It retrieves the names of the computers in the network and their associated IP addresses. By using this information we can get connected the systems in network.

Listing of Online & Offline Users

The next stage in the system is to list online and offline users in the network. An user becomes an online user if he can respond the request made by the system. Otherwise he will be an offline user.

The request will be send to the offline users to get connect to this system to have communication. If he accepts he will become an online user. The system will check frequently whether a particular user is remaining online

and update the list. Frequently the request will be send to the offline users to get connected at regular intervals.

Text chatting

Instant messages can be sent to all the online users. When the text is typed in the text area then it will be updated in the widow of the connected user. The user can change the attributes of the text. He can change the font, font size and the color of the font.

Voice chatting

The user can have bidirectional voice communication with any one of the online users. For this purpose his system should have an sound card, wave input device (Microphone) and a wave output device (Headphone of Speakers).

The wave input received from the wave input device will be converted into the data that can be send through the network. Then it will be sent.

At the receiving side the voice data is again converted into a wave data and played in a wave output device.

File Transfer:

The user can send a file to an online user. The user has to select the file for the transmission. Then it will be send to the other user.

At the receiving end the user can accept it or deny it. If he accepts it he have to specify the location to save that file. Then the file will be copied at the specified location.

4.4 Process Design

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

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

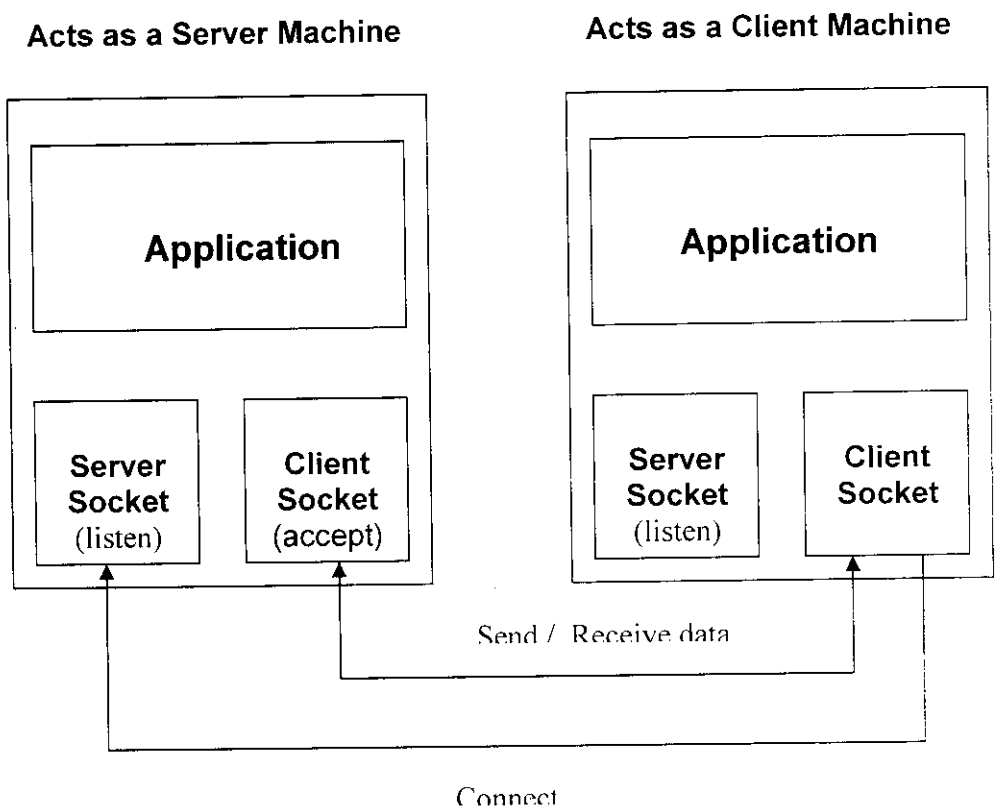
Design tools:

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

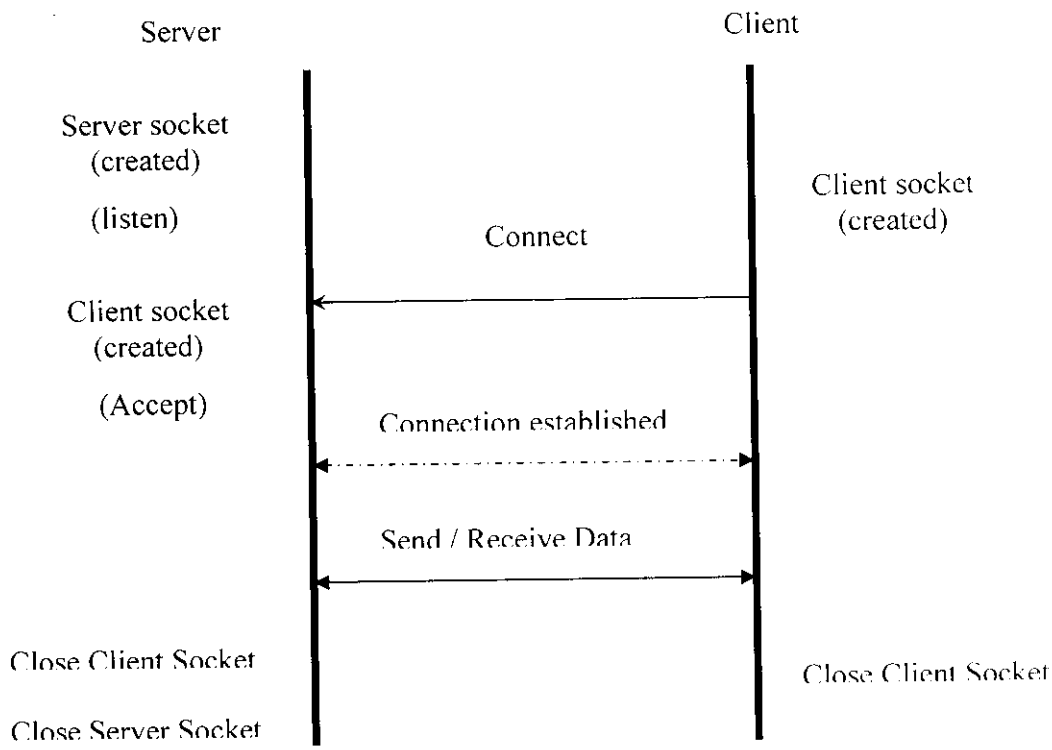
- Flowcharts
- Computer Run Chart
- Computer Procedure Flowchart/Process Diagrams
- Network Chart
- Interactive System Flowchart
- Decision Tables

Process diagram

Overall diagram:

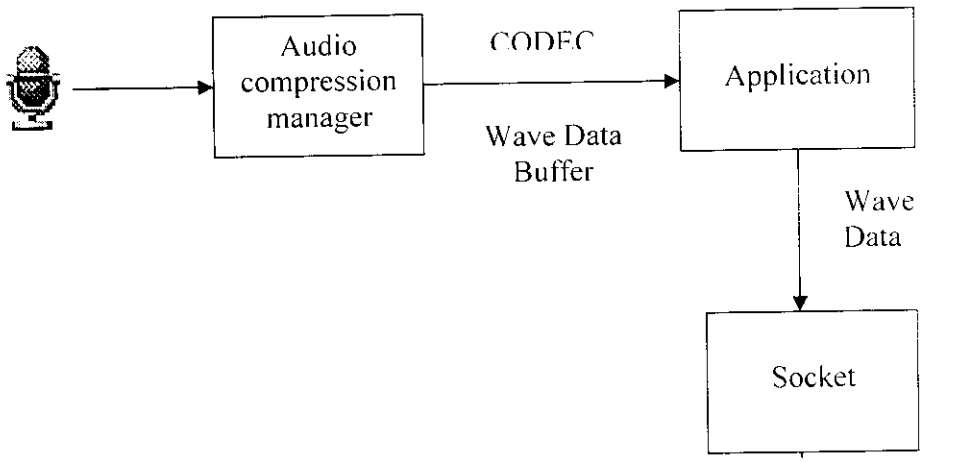


Text Chatting Process

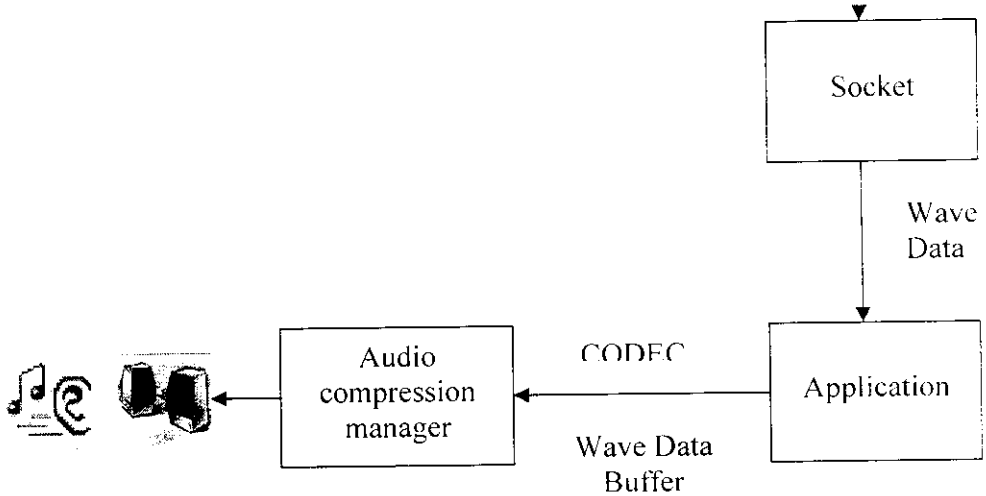


Voice Chatting

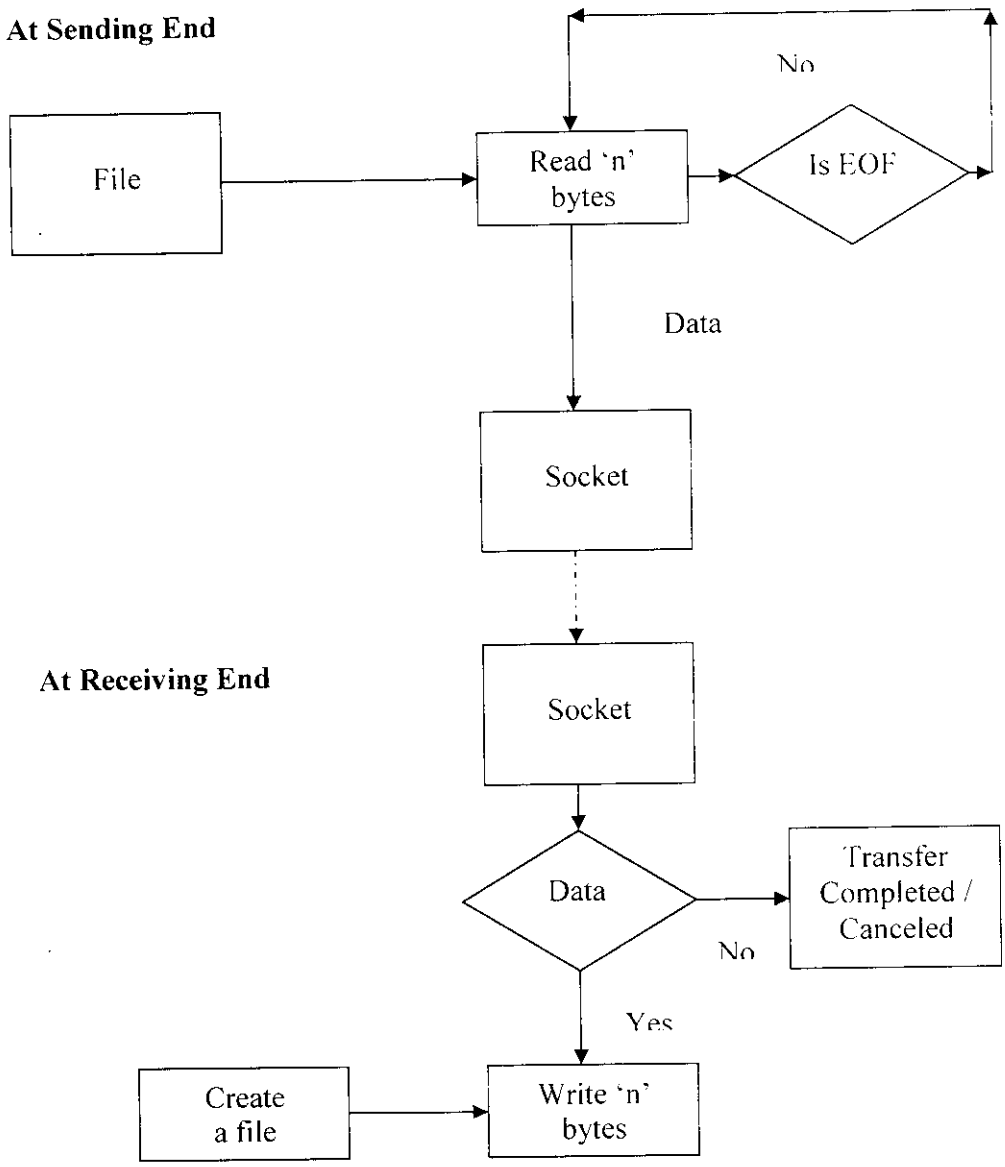
Recording



Playing



File Transfer



5. SYSTEM IMPLEMENTATION AND TESTING

5.1 System Implementation

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

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

There are three types of implementation:

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

The newly developed Network Messenger has to be implemented successfully. Training aids for staffs have been provided in the form of help screens. Since organizations system undergoes continual change, the application will undoubtedly have to be maintained. Modifications and changes

will be made to the user requirement in order to keep pace with the changing environment.

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

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

5.2 System Testing

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

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

Testing Methods

Unit Testing

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

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

Sub - System Testing

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

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

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

System testing

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

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

Interface testing

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

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

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

Procedural interfaces: these are interfaces where one sub-system requests a service from another sub-system by passing a message to it. A return message includes the results of executing the service. In this stage, the module-level interfaces and the system-level interfaces were tested thoroughly for errors.

Test Cases For System Testing

System testing is largely functional in nature. The focus is on invalid and valid cases, boundary value and special cases.

Seq. No.	Test case	Conditions being checked	Expected output
1	IP Address data field empty	Add new contact	Print message and stop
2	Wrong IP Address	Add new Contact	Print message and stop
3	Sending instant message	Try to make connected with the offline user	Print message and stop
4	Online user disconnected	While having communication	Becomes an off line user
5	No Sound Card support	Try to have voice communication	Print message and stop
6	No wave input device	Try to have voice communication	Print message and stop
7	No wave output device	Try to have voice communication	Print message and stop
8	Speaker in off state	Try to have voice communication	Print message and stop
9	No disk space	While transferring file	Print message and stop
10	Cancel File Transfer	While transferring file	Print message and stop

Refinements based on feedback

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

6. Conclusion

The proposed system "Network Messenger" was found to be very effective. It was able to achieve its intended goals.

The system was developed with quality consciousness and more importantly it pertained to the standards of the company. Another issues of significance is that this system was built rigidly, and exhaustive testing strategies were adopted. The testing was completely successfully for all the requirements and specifications, and the developer gave the sign off. As per schedule, the objectives of the proposed system have been realized.

I believe that I have put all my effort to implement this project sincerely

7. Scope for Future Enhancement

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

- The Audio Mixer features can be added to improve the quality of the sound produced
- Video Conferencing module can be added to the system
- User details can be collected and updated in a database and can verify their logon details etc..,

8. Bibliography

- Elias M Awad, "System Analysis And Design", Galgotia Publications, Fourth Edition 1999

- Roger S Pressman, "Software Engineering", Addison-Wesley, Fifth Edition 1999

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

- Antony Jones and Jim Ojlund, "Network Programming for Windows", Microsoft Press, 2000

- Lewis Napper, "WinSock 2.0" IDG Books Worldwide, Inc, 2000

- Andrew S. Tanenbaum "Computer Networks", Prentice – Hall of India, 1999.

- Chuck Sphar "Learn Microsoft Visual C++ 6.0 Now" Microsoft Press, 2001.

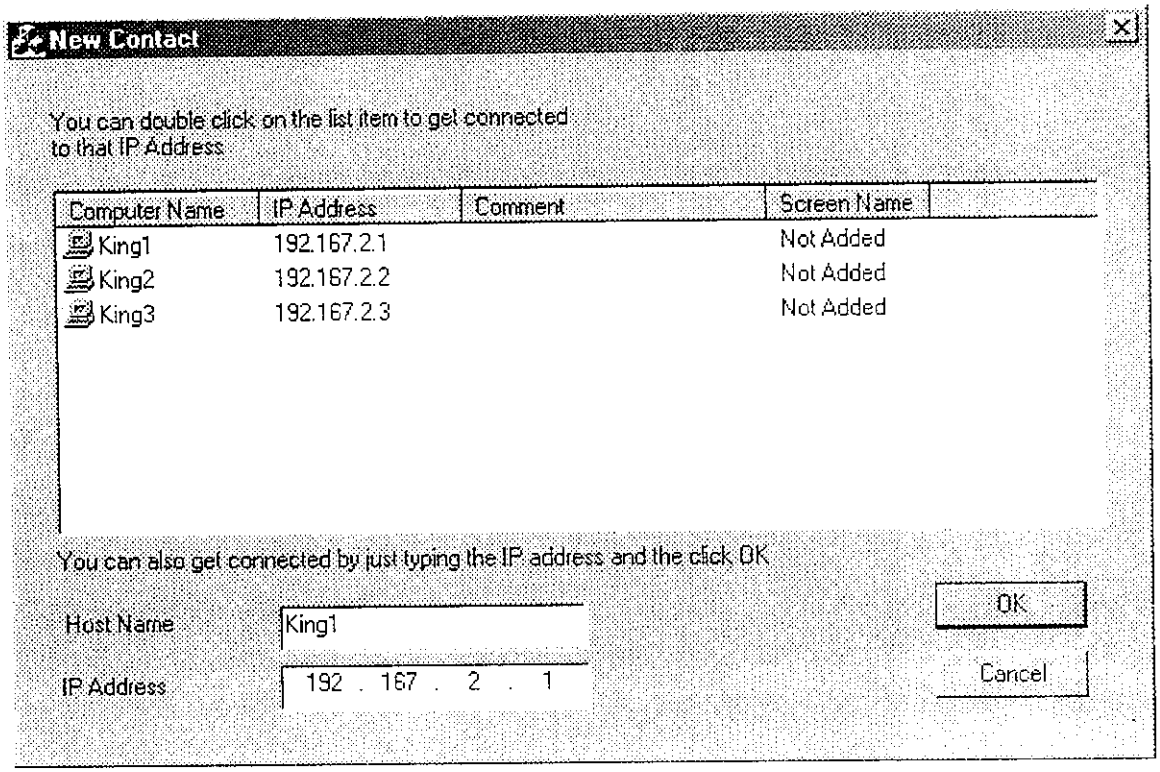
Reference:

- www.microsoft.com/msdn

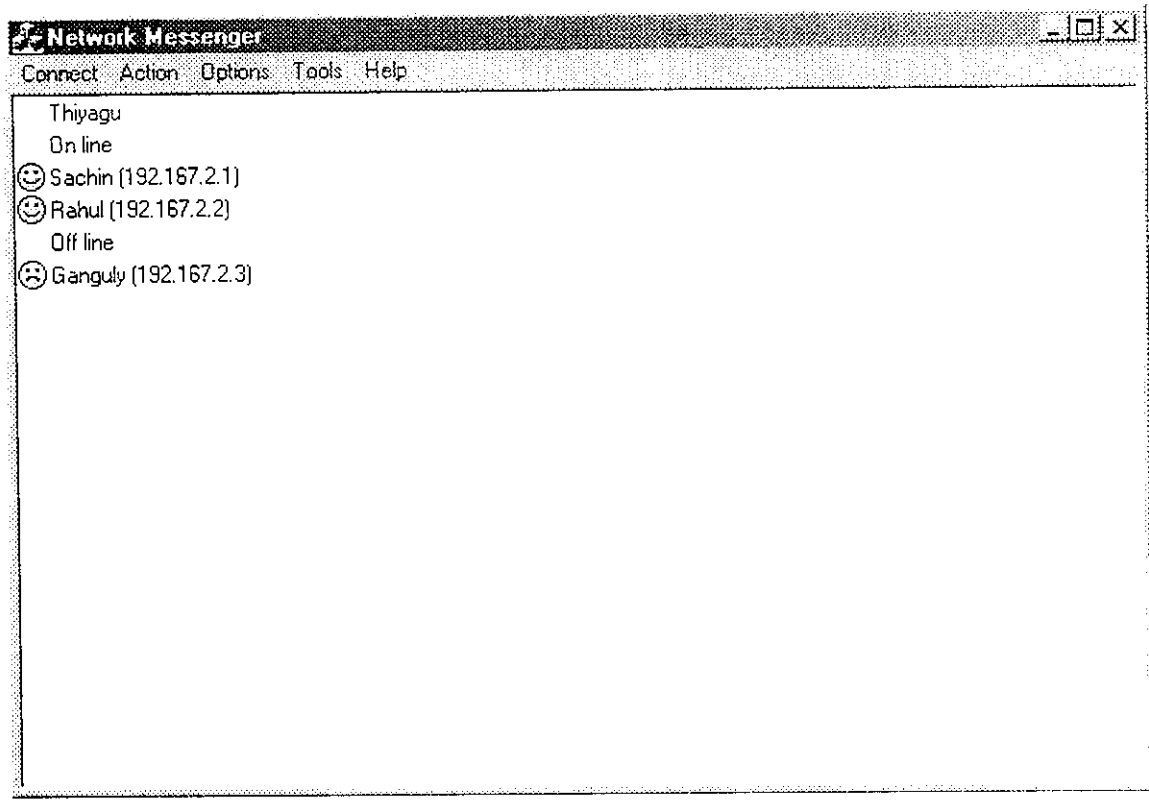
9. Appendix

9.1 Screens

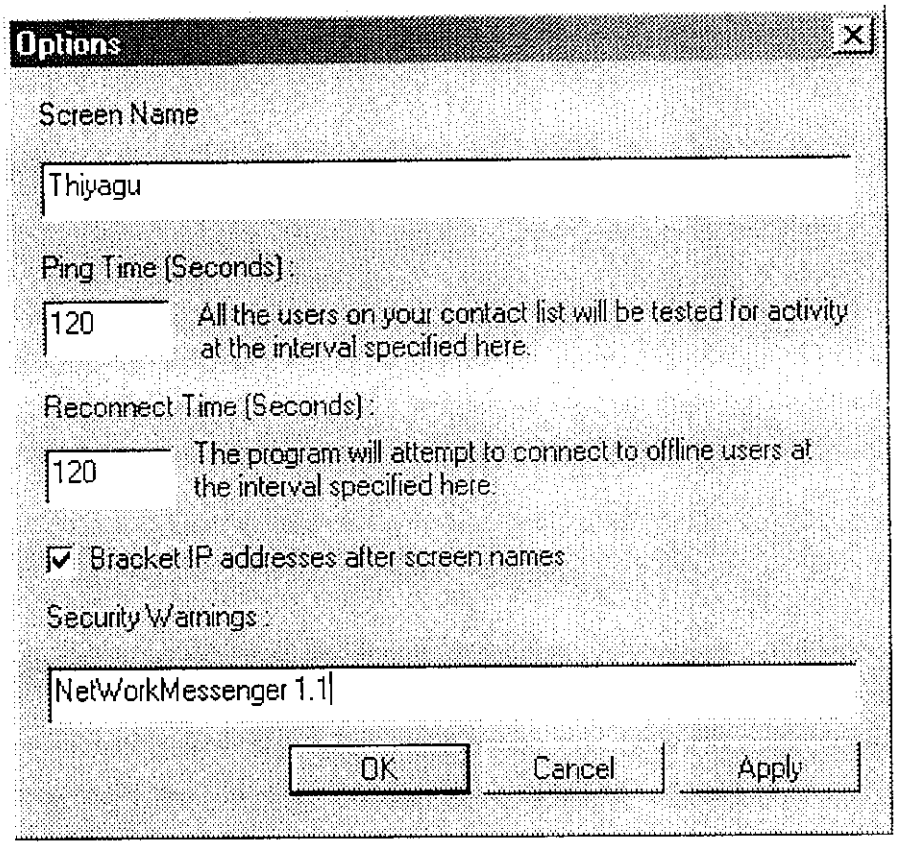
Listing Of Network Resources



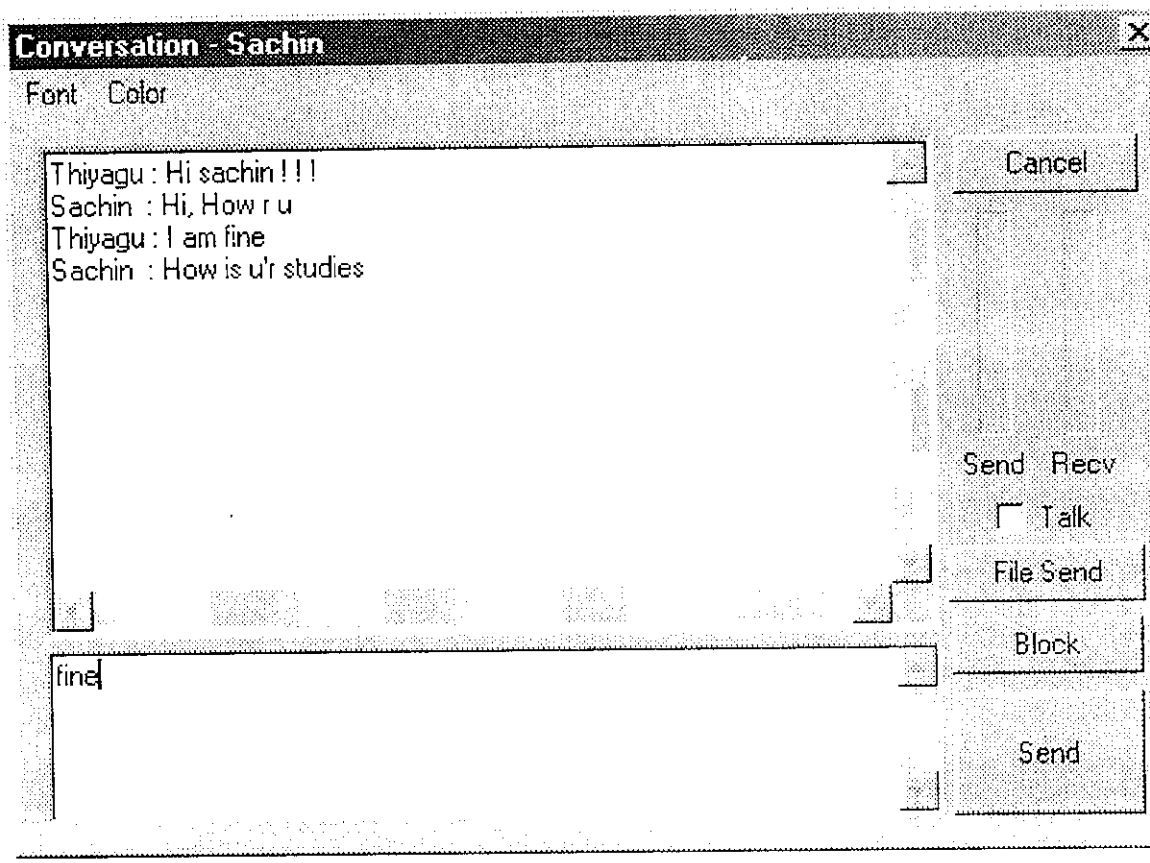
List of Online & Offline users



Options


Options
Screen Name
Thiyagu
Ping Time (Seconds) :
120
All the users on your contact list will be tested for activity at the interval specified here.
Reconnect Time (Seconds) :
120
The program will attempt to connect to offline users at the interval specified here.
Bracket IP addresses after screen names
Security Warnings :
NetWorkMessenger 1.1
OK
Cancel
Apply

Conversation Dialog



File Transfer Status Screen