Real Time Digital Audio Streaming Over The Internet Network

by

Shahidan Izham Yeop Ibrahim

Final Project Report

Dissertation submitted in partial fulfilment of

the requirements for the

Bachelor of Engineering (Hons)

(Electrical Engineering)

July 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

+
TK
5105.386
2004
a) Streaming technology (Telecommunication)
b) EEE -- Thesis

CERTIFICATION OF APPROVAL

REAL TIME DIGITAL AUDIO STREAMING OVER INTERNET NETWORK

by

Shahidan Izham Yeop Ibrahim

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirements for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved by,

[illegible]

(Ms Nasreen Badruddin)

Project Supervisor

lasreen Badruddin
ecturer,
lectrical & Electronic Engineering
cademic Block No 22
niversiti Teknologi PETRONAS
andar Seri Iskandar
31750 Tronoh, Perak Darul Ridzuan, MALAYSIA

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

July 2004

CERTIFICATION OF ORIGINALITY

This is to verify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

(Shahidan Izham Yeop Ibrahim)

ABSTRACT

The distributions of music over the internet have become the popular ways for the music lovers to exchange their music experience. Since the file can be downloaded easily, especially using the mp3 format, users don't have to wait for a long a time to listen to any songs of their interest. Most of the music companies also have taken a step in introducing their recording artist by allowing user to hear a preview of the songs through websites, using the audio streaming technique. Of course, they are several ways a user can choose to listen to music, but the streaming is basically to create an alternative way in broadcasting and to share the music experienced over the internet network. The objective of this project is to create a system where two computers can listen to a high quality audio through audio streaming in real-time. This project covers a network programming, multimedia application and some of the software engineering knowledge in implementing the audio streaming program. Using Visual Basic 6 as them main programming language, both the networking programming and GUI or Graphical User Interface of the program can be applied. The results produced are the full working program that enabled user to have a good quality audio streaming between two computers and also the project flow before and after the audio streaming. The founding from the results are discussed and concluded as a whole, where the aim of the studies have been fulfilled, thus making the project success. Finally, the author has made several recommendations to improve the relevant study and future works.

ACKNOWLEDGEMENT

Firstly, I would like to acknowledge my deepest gratitude to my supervisors, Ms Nasreen Badruddin for their expert guidance, valuable suggestions, enthusiastic support and personal concern during the duration of this final year design project. In addition to that, I would like to extend my sincere to the Programme Head of Electrical and Electronic Engineering Department, Dr. Md Noh Karsiti for allowing me to use all the facilities in completing this project.

Deepest appreciation also goes to the whole committee members of Final Year Project for the highly cooperation, collaboration and support that gave to each others. Last but not least, special appreciation goes to ever-loving parents who always on my side, supporting me on my up and down as well as giving me the encouragement to pursue my dream.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The needs of music nowadays are seem to be quite an essential for most of the people around the world. Starting with the broadcasting through the radio, the development of the music industry is growing really fast. And nearly in less than ten years, since the popularity of the internet and audio compression, the music distribution is no longer a big problem. People can listen to any songs of their interest and have a lot of choices since the number of the internet user are increasing rapidly everyday.

1.2 Background of study

This project is to implement a program that will perform an audio streaming between two computers. Visual Basic 6 is choose as the main programming language that will carry out the audio streaming process. It will support both the network programming and applying the GUI.

Technically, streaming works by first compressing a digital audio file and then breaking it into small packets, which are sent, one after another, over the Internet. When the packets reach their destination (the requesting user), they are decompressed and reassembled into a form that can be played by the user's system. To get a continuous play, the packets are buffered so a number of them are downloaded to the user's machine before playback. As those buffered or preloaded packets play, more packets are being downloaded and queued up for playback.

1.3 Problem statement

The high demand of internet multimedia leads to the enhancing the capability of multimedia itself. A lot of software programs have been created in order to fulfill the user needs in multimedia. Undeniably, the downloading program appears much more often than the streaming programs due to the fact that, it produces the same quality as the original file, technically. The streaming program on the other hand, helps user to save more time rather than have to wait for some time to download the file. It has the capability to play the file directly without have to wait, meaning user will have a real-time broadcasting direct from the server itself.

1.4 Problem identification

Audio streaming always allows only a snippet of songs to be heard by the user. It is also a fact that audio streaming usually produce a low quality audio file to the user, depends on the speed of the connection. This project requires an author to design an audio streaming program that will produce a good quality of audio. As in advance, the system will allow user to access the audio file from the remote computer.

In order to have good real-time digital audio streaming program, this few basic studies will be covered, and the problem arise from each studies will be covered throughout the project:

1. Internet network

   The connection between two computers needs to be defined and recognized first.

2. Data transfer

   Data transfer between computers is what differ the streaming technique and the downloading technique.

3. Real-time multimedia

   Real-time multimedia will be implemented as the title suggest

4. Audio format

   The audio formats consist of several known types. Each will be determined and the commonly used will be considered.

5. Graphical User Interface

   Graphical user interface is important to have a communication between the program and the user-end.

1.5 Significant of the project

This project is to implement an audio streaming program between two computers. The significance of this project is to create an effective way in sharing an audio file over the internet network. The existing audio streaming concept seems to be limited only to a low quality audio file and a limited size of files. Unlike to the existing audio streaming, user will be benefits to have a better audio quality and have the full length of files while streaming. This project also must be reliable enough for further enhancement, such as using more suitable interface and can be applied to a larger network connection. If this project is a complete success, it might also be applied in real industry, as long as it is stable and secured enough. It is going to play its part in the development of the file sharing over internet network.

CHAPTER 2

LITERATURE REVIEW

2.1 Literature Review

A lot of researches and experimenting have been done in order to make this project works. This project covers a few major parts that need a lot of understanding and knowledge. This section discussed some important analysis that will be covered throughout the process.

2.2 Audio and computers

In general, audio can be produced either naturally by means of a microphone or electronically by means using some form of synthesizer. In the case of synthesizer, the audio is created in a digital form and hence can be readily stored within the computer memory. A microphone however, generates a time-varying analog signal and in order to store such signal in the memory of a computer and to transmit them over the digital network, they must first be converted into a digital form using an audio encoder.

Once digitised, any form of audio can be stored within a computer. However, the amount of memory required t store digitised audio waveform can be very large, even for relatively short passage. It is for this reason that synthesized audio is often used in multimedia application since the amount of memory required can be between two and three orders of magnitude less than that required to store the equivalent digitised waveform version.

2.3 Networking connections

In order to have a connection between two computers, a network connection needs to be established. However they are a few types of connection can be made in order to make things works. In multimedia, they are five types of network connection, namely:

i. Telephone networks
ii. Data networks
iii. Broadcast television networks
iv. Integrated service digital service networks
v. Broadband multiservice networks

Each of the networks plays a different role and will have a different connection. A few consideration need to be made in order to choose the right network connection that can be established.

As with WAN, LAN is a communication network that interconnect a variety of devices and provides a means for information exchange among those devices. LAN make use of a broadcast network approach rather than a switching approach. With a broadcast communication network, they are no intermediate switching nodes. At each station, there is a transmitter/receiver that communicates over a medium shared by other stations. A transmission from any one station is broadcast to and received by all other station. Data are usually transmitted in packets. Because the medium is shared, only one station at a time can transmit a packet.

There are several key distinctions between LANs and WANs. The scope of LAN is usually small, typically a single building or a cluster of buildings. This is what always be differentiate between LAN and WAN. The internal data rates of LAN are also much greater than those of WANs. Figure 1 below shows how LANs and WANs work. Local area network seems to be the best way to implement the project as the project only need two computers that are connected to each other, one as a server and the other one as the client. The graphical is showed in figure 2.1. [2]

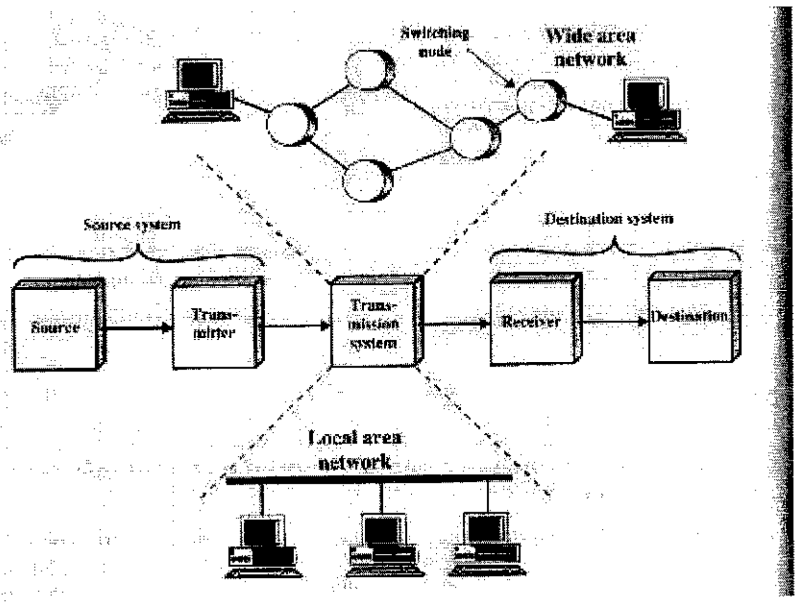


Figure 1 LAN and WAN connection

2.4 TCP/IP

TCP stands for Transmission Communication Protocol and IP stands for Internet Protocol. TCP and IP were developed by a Department of Defense (DOD) research project to connect a number different networks designed by different vendors into a network of networks. It was initially successful because it delivered a few basic services that everyone needs (file transfer, electronic mail, remote logon) across a very large number of client and server systems. Several computers in a small department can use TCP/IP (along with other protocols) on a single LAN. The IP component provides routing from the department to the enterprise network, then to regional networks, and finally to the global Internet. As with all other communications protocol, TCP/IP is composed of layers:

- **IP** - is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.
- **TCP** - is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.
- **Sockets** - is a name given to the package of subroutines that provide access to TCP/IP on most systems.

TCP/IP assigns a unique number to every workstation in the world. This IP number is a four byte value that, by convention, is expressed by converting each byte into a decimal number (0 to 255) and separating the bytes with a period. [14]

2.5 Graphical User Interface (GUI)

A GUI is a graphical, (rather than purely textual) user interface to a computer. The term came into existence because the first interactive user interfaces to computers were not graphical; they were text-and-keyboard oriented and usually consisted of commands that we had to remember and computer responses that were infamously brief. The command interface of the DOS operating system is an example of the typical user-computer interface before GUIs arrived. An intermediate step in user interfaces between the command line interface and the GUI was the non-graphical menu-based interface, which allows interaction by using a mouse rather than by having to type in keyboard commands.

Nowadays, major operating systems provide a graphical user interface. Applications typically use the elements of the GUI that come with the operating system and add their own graphical user interface elements and ideas. A GUI sometimes uses one or more metaphors for objects familiar in real life, such as the desktop, the view through a window, or the physical layout in a building. Elements of a GUI include such things as:

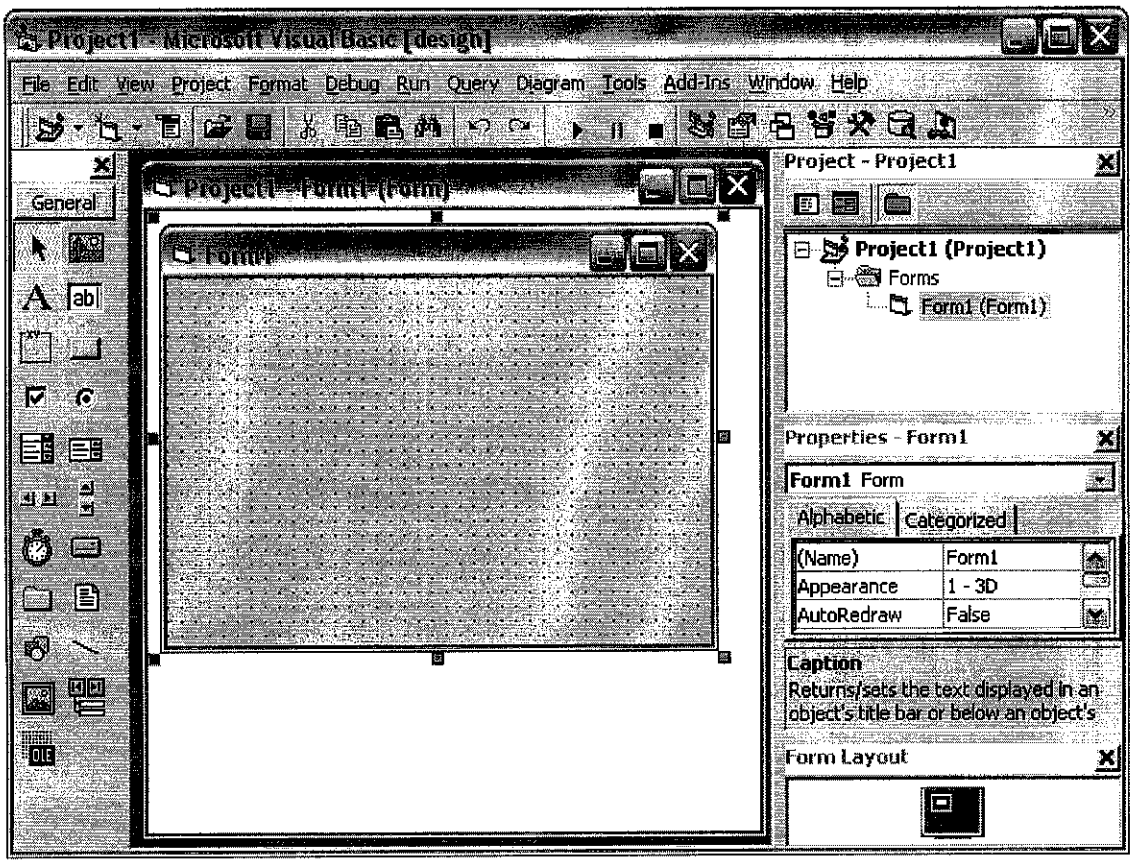
- windows
- pull-down menus
- buttons
- scroll bars
- iconic images
- wizards

With the increasing use of multimedia as part of the GUI, sound, voice, motion video, and virtual reality interfaces seem likely to become part of the GUI for many applications. A system's graphical user interface along with its input devices is sometimes referred to as its look-and-feel.

The GUI familiar to most of us today in either the Mac or the Windows operating systems and their applications originated at the Xerox Palo Alto Research Laboratory in the late 1970s. Apple used it in their first Macintosh computers. Later, Microsoft used many of the same ideas in their first version of the Windows operating system for IBM-compatible PCs.

2.6 Microsoft Visual Basic Environment

The Microsoft Visual Basic is known to be among the most popular choice to create Windows GUI. In Visual Basic, new windows created are called forms. Elements (such as text boxes and buttons) that are placed inside a form are called controls. The Visual Basic allows event-driven programming, where the user's actions cause events, and each event in turn triggers a procedure that is associated with it. [10]



(Name)	Form1
Appearance	1 - 3D
AutoRedraw	False

Caption
Returns/sets the text displayed in an object's title bar or below an object's

Figure 2 Visual Basic Editor

The properties of creating an object model in Visual Basic are:

- Objects have properties and methods
- Forms and controls are examples of objects
- Properties describe an object. Examples: name, color, size, or how they will behave
- Methods are actions associated with an object. Examples: move, clear and print

In order to write a proper Visual Basic project, there are several important elements to learn and understand. The two vital steps are:

2.6.1 Planning

1. Design the user interface
2. Plan the properties
3. Plan the Basic code - procedures are associated with the events, actions written in pseudocode

2.6.2 Programming

1. Define the user interface - define objects
2. Set the properties
3. Write the Basic code

Table 1 Visual Basic normally used naming conventions

Object	Prefix
Form	frm
Command Button	cmd
Text Box	txt
Label	lbl
Option Button	opt
CheckBox	chk
Frame	fra
Horizontal Scrollbar	hsb

Vertical Scrollbar	vsb
Image	img
Picture Box	pic
Combo Box	cbo
List Box	lst
Shape	shp

2.7 Strings in VB

This project requires a lot of understanding in handling a string function. Some of the function includes separating, remove add, count, etc. Below is the list of some of the function used in handling a string in Visual Basic 6. [12]

Table 2 Example of strings function

Mid function and Mid statement	To extract characters from the middle of a string. Three arguments used: The original **string**, the place to **start** extracting, and the **number** of characters to extract. The Mid *statement* not only extracts the characters, but replaces them with the text specified. Since this is a statement (not a function), you won't get a result, rather the action is just completed for you.
Option Compare Text	A big part of working with strings is comparing one to another to see if they

	match. There are two ways to compare strings - case sensitive and case insensitive. By default, VB will use the Binary method to compare strings. This can be changed for individual modules by placing an Option Compare Text statement in the general declarations section. Option Compare Binary is also a valid statement, but it's never necessary to write this out.
String function	This function is used for producing a string with a certain number of repeating characters. It's two arguments are the number of characters and the character code to repeat
Trim, LTrim, and RTrim functions	These functions remove leading (LTrim) or trailing (RTrim) spaces from a string. It won't affect any spaces between words. Trim simply accomplishes both LTrim and RTrim. Users may inadvertantly type spaces into a Text Box, and you'd use these functions to account for that, but the largest use of Trim is with fixed-length strings, user-defined Types, and Random Access Files.
Left and Right functions	Used to extract a certain number of characters from the leftmost or rightmost portion of a string. The function requires two arguments: the original string and the number of characters to extract.

2.8 Connection

2.8.1 Winsock

Winsock aka Windows Sockets is a communications stack for communicating using the TCP/IP protocol. In more basic terms, it is a standardized interface for communicating with various types of machines, and enables you to perform multiple tasks at the same time through your connection (be it modem, ISDN, LAN, or WAN), such as reading mail and downloading a file at the same time, from different places. The Winsock driver (a DLL in the case of MS-Windows) provides the standardized interface for various "client" programs to run on your computer. A "client" program is the program which performs the type of task you want performed (like FTP, Telnet, IRC, News, Mail, Finger, Archie, WWW, Gopher, MUDs, etc). Winsock allows a variety of programs to operate on your machine without need for them to directly understand your hardware configuration (that is part of the job of the Winsock driver itself). In theory, you can get a number of client programs, and a Winsock driver for modem, use the client software, and later switch to an Ethernet or ISDN network connection (with another Winsock driver), and still be able to use all the software you already have.

2.8.2 Version of Winsock

Windows Sockets is an independent specification, which was created and exists for the benefit of application developers and network vendors and, indirectly, computer users. Each published (non-draft) version of this specification represents a fully workable API for implementation by network vendors and programming use by application developers.

2.8.3 Winsock version 1.1

WinSock version 1.1 has been the standard since its release in January of 1993, and has exceeded its authors' original intent to provide a powerful and flexible API for creating universal TCP/IP applications. Some have argued that it was an unsung hero in the Internet's phenomenal success, as it enabled the creation of a single network application like Netscape's browser, for example to run on the many TCP/IP stacks for PC's that existed at the time.

Since those early days, the PC network industry has changed completely: The many TCP/IP vendors have all but disappeared after their software was replaced by Microsoft's free TCP/IP implementations built into the operating systems. The Internet has continued to *explode* in popularity, as have the number of applications that depend on WinSock and they have changed as they diversified. Applications have become more demanding of Internet services. As a result, the Internet itself is changing, evolving. The *Grand Convergence* of phone, radio and television networks to the Internet Protocols (IP) is underway. So, of course, it comes as no surprise that WinSock the APIs have undergone change also.

The authors of Windows Sockets version 1.1 originally limited the scope of the API specification to TCP/IP primarily, but this focus did not preclude the possibility that WinSock could support other protocol suites. So they came up with WinSock Version 2.

2.8.4 Winsock version 2

Windows Sockets version 2.0 (Winsock 2) formalizes the API for a number of other protocol suites such as ATM, IPX/SPX, and DECnet. It allows them to coexist simultaneously. Winsock 2 also adds substantial new functionality. Most importantly, it does all this and still retains full backward compatibility with the existing 1.1 some of which is clarified further so all existing Winsock applications can continue to run without modification (the only exception are WinSock 1.1 applications that use blocking hooks, in which case they need to be re-written to work without them).

WinSock 2 goes beyond simply allowing the coexistence of multiple protocol stacks, in theory it even allows the creation of applications that are network protocol independent. A WinSock 2 application can transparently select a protocol based on its service needs. The application can adapt to differences in network names and addresses using the mechanisms WinSock 2 provides.

Here is a list of the new features that WinSock 2 provides:

1. Multiple Protocol support: WOSA architecture let's service providers plug-in and pile-on.
2. Transport Protocol Independence: Choose protocol by the services they provide.
3. Multiple Namespaces: Select the protocol you want to resolve hostnames, or locate services.
4. Scatter and Gather: Receive and send, to and from multiple buffers.
5. Overlapped I/O and Event Objects: Utilize Win32 paradigms for enhanced throughput.
6. Quality of Service: Negotiate and keep track of bandwidth per socket.
7. Multipoint and Multicast: Protocol independent APIs and protocol specific APIs.

8. Conditional Acceptance: Ability to reject or defer a connect request before it occurs.
9. Connect and Disconnect data: For transport protocols that support it.
10. Socket Sharing: Two or more processes can share a socket handle.
11. Vendor IDs and a mechanism for vendor extensions: Vendor specific APIs can be added.
12. Layered Service Providers: The ability to add services to existing transport providers.

In a nutshell, Winsock 2 is Winsock 1.1 on steroids; it's a superset of 1.1's APIs and architecture. In addition to its new features, it also clarifies existing ambiguities in the 1.1 Winsock specifications and adds new extensions that take advantage of operating system features and enhance application performance and efficiency. Finally, Winsock 2 includes a number of new protocol-specific extensions. These extensions such as multicast socket options are relegated to a separate annex, since the main Winsock 2 protocol specification is protocol-independent

The specification for Winsock 2 is actually made up of several separate documents. Each of these documents is usually (though not always) available in the 3 different formats: MS Word for Windows (.doc), Text (.txt) and Postscript (.ps). Most are BIG (250+ pages each), so you are encouraged to print double-sided, if possible.

2.8.5 Technical requirements

Basically, Windows Socket needs a connection access to the Internet. The technical requirements are:

2.8.5.1 A suitable connection for Internet.

Connection to the Internet may take the form of a direct connection via a network card or a dialup account using a modem. A fast computer, 8MB or more of memory, and a high-speed modem for dialup connections (at least 14.4k) are also recommended.

2.8.5.2 A TCP/IP stacks (WINSOCK.DLL)

Some operating systems include stacks, such as Microsoft Windows 95 and IBM OS/2. For other operating systems, like Microsoft Windows 3.1/3.11 or Microsoft Windows for Workgroups 3.11, stack need to be added.

2.8.5.3 Differences between Winsock and TCP/IP

TCP/IP	**WINSOCK**
TCP/IP is a network protocol, meaning that it is at layers 3 through 4 in the OSI model. A network protocol provides services like addressing, data transport, routing, and logical connections across a network. Two computers must use the same network protocol if programs running on those computers are to communicate. Other common network protocols include Novell's IPX, 3Com/IBM/Microsoft's NetBIOS and Apple's AppleTalk. TCP/IP is the most popular network protocol today: virtually all computers support it.	Winsock is an API that lets a Windows program send data over any network transport protocol. There are several functions in Winsock that only work with TCP/IP (e.g. `gethostbyaddr()`), but there are newer generic versions of all these in Winsock 2 which allow you to use other transports.

CHAPTER 3

METHODOLOGY AND PROJECT WORK

3.1 Procedure Identification

As the title suggests, the procedures involved for implementation throughout the project work are divided into several systematic steps. Applying the waterfall model, it provides systematic approach in solving any encountered problem.

3.2 Analysis

Before the project can be started, a few scopes of studies are determined and narrow down for further analysis. Three major three parts covers in the analysis part. There are programming language, audio format and GUI. These three provide sample information and detail in proceeding to the next designing stage. Each element is studied and is performing by doing literature reviews and research on the Internet.

3.3 Design

The design stage for the system based on the general block diagram which represents the overall attributes of the system. The system block diagram is shown in Figure 3.1 below.

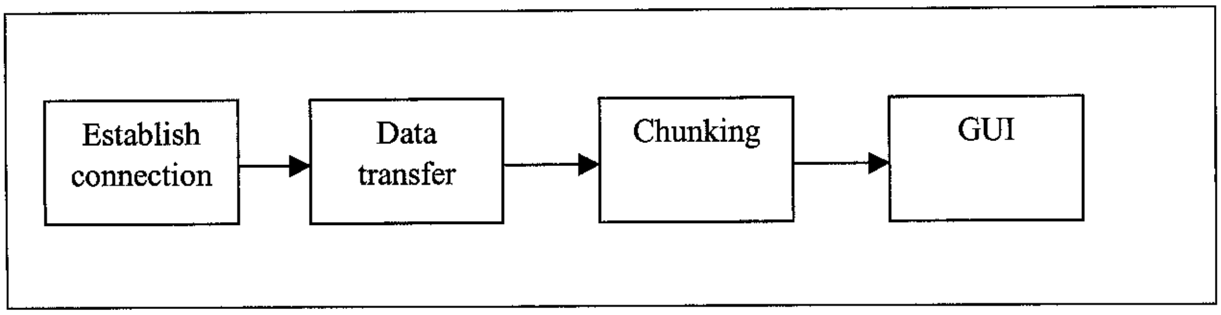


Figure 3 System block diagram

The basic structure of this system consists of:

1. The establishment of connection between two computers.
2. Data transfer from one computer to another.
3. Chunking, where each file from server will have to be split out into several parts before it can be transferred and played in client.
4. A GUI interface to enhance the capability of the audio streaming program and acting as a communication part from the user-end.

3.3.1 Connection

The connection between two computers is using the Winsock connection. The connection established will enable the two computers to communicate to each others using the TCP\IP connection. The connection also allows both computers to send and received data. The connection of Winsock is basically follows this simple steps.

First of all, the serve will listen to the port open, declares in VB. When the port is opened, the client will ask for a connection request. Server will then received the connection request. After that, it will close the socket and accepts the connection. The client will grant the accepted connection, and finally the connection established. Figure 3.2 explains how the connection recognized.

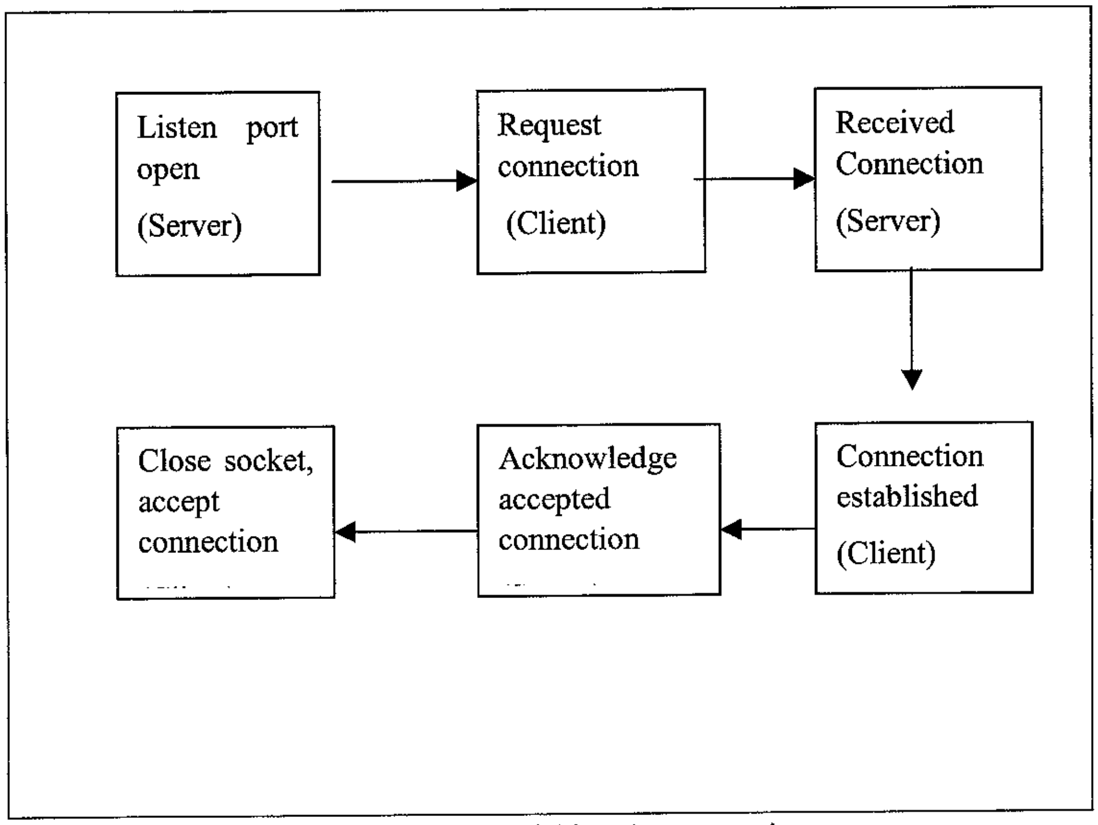

Listen port open (Server)
Request connection (Client)
Received Connection (Server)
Connection established (Client)
Acknowledge accepted connection
Close socket, accept connection

Figure 4 Establishing the connection

3.3.2 Data transfer

During the data transfer, each file in server will have to be split into three parts, namely beginning of file (BOF), middle file, and the end of file (EOF). The data filename will be stored as a string of filename, and then separated into three parts as mention. This is to indicate the client that the there will be three different files is coming. The BOF is to trigger the client that there is an incoming file from server, and the BOF to indicate that there is no more data left to received by client.

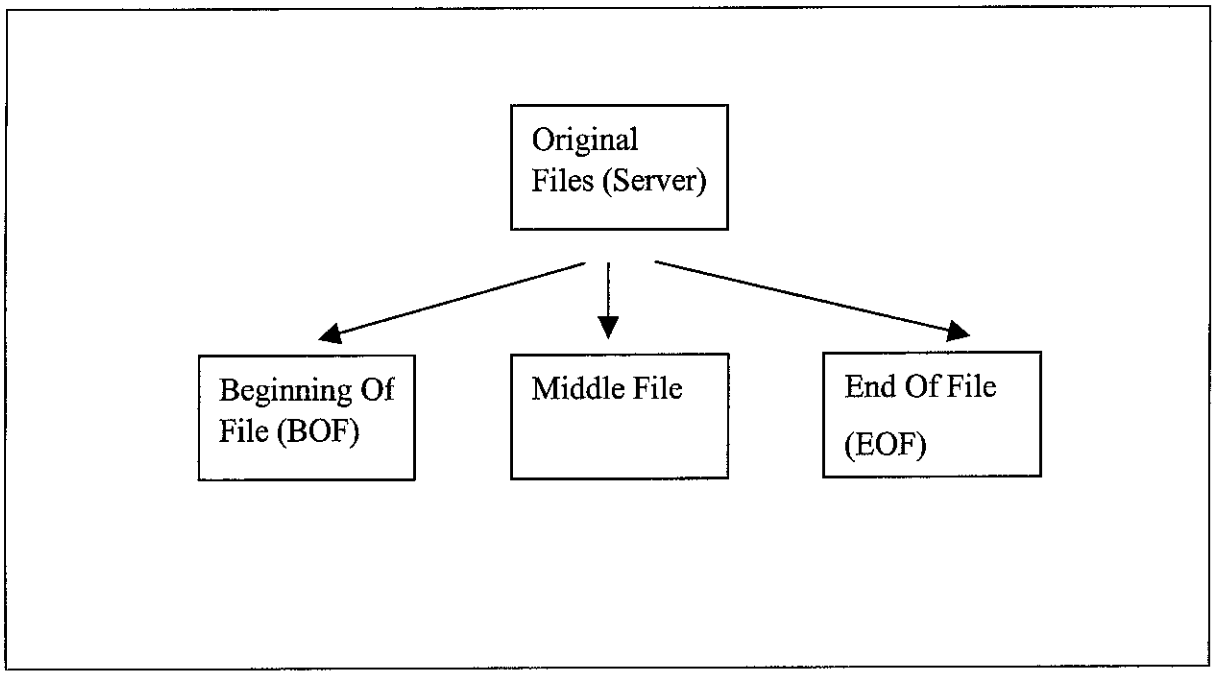


Figure 5 Separation of file

3.3.3 Chunking and streaming

The middle file send by the server is separated in chunk, and each chunk is stored into a temporary files namely temp1.mp3 and temp2.mp3. During this process, visual basic will open a binary files to stored the temporary files and allowing the mp3 files to be implemented in this program. Each chunk will then played, temp1 and temp2 accordingly.

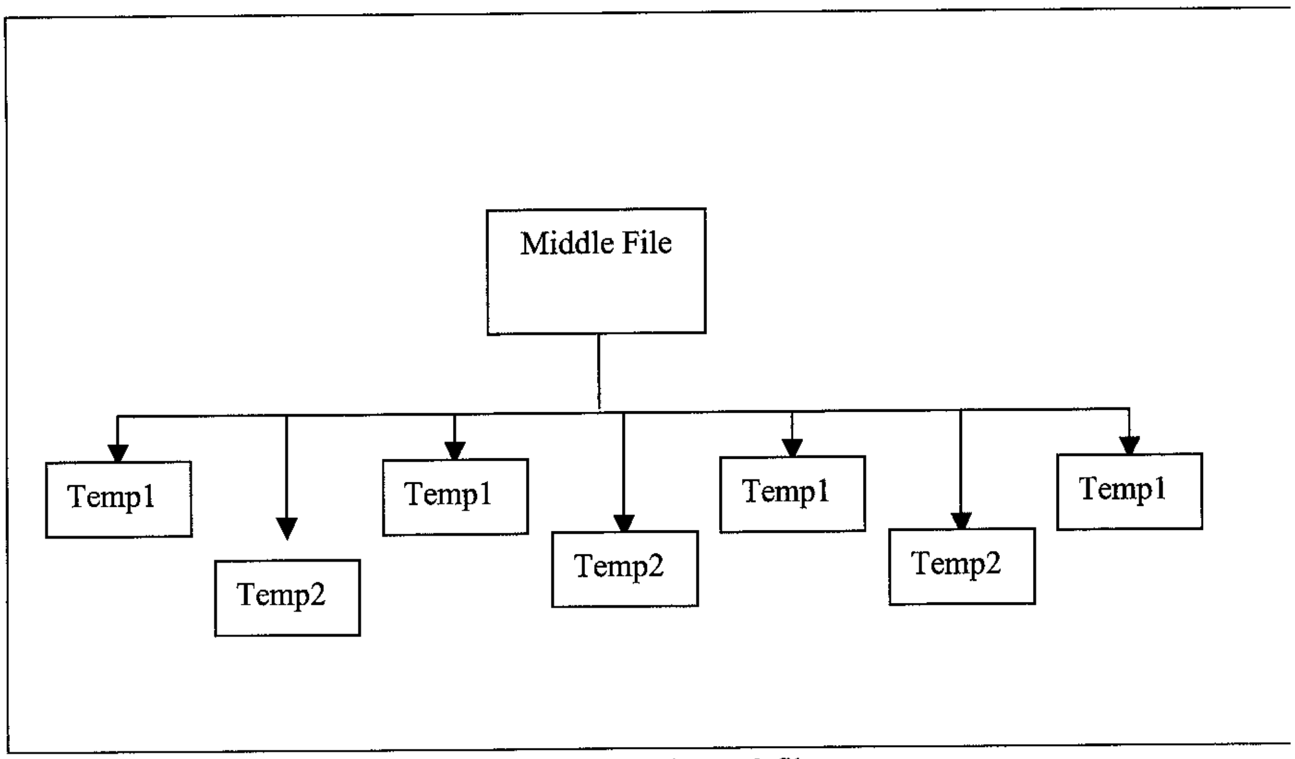


Figure 6 Temp1 and temp2 files

In order for the file to stream, each chunk received will be played by the client. As mention earlier, the middle file will be separated into chunks of file, which is temp1 and temp2. These files will be played immediately right after the BOF is received. Both files will be deleted right after they are played, and this process continues in real-time until the EOF is received by the client, showing that no more file is coming. Figure 3.4 and figure

3.5 explains the flow chart on how the program will be sunning both in client and server, respectively.

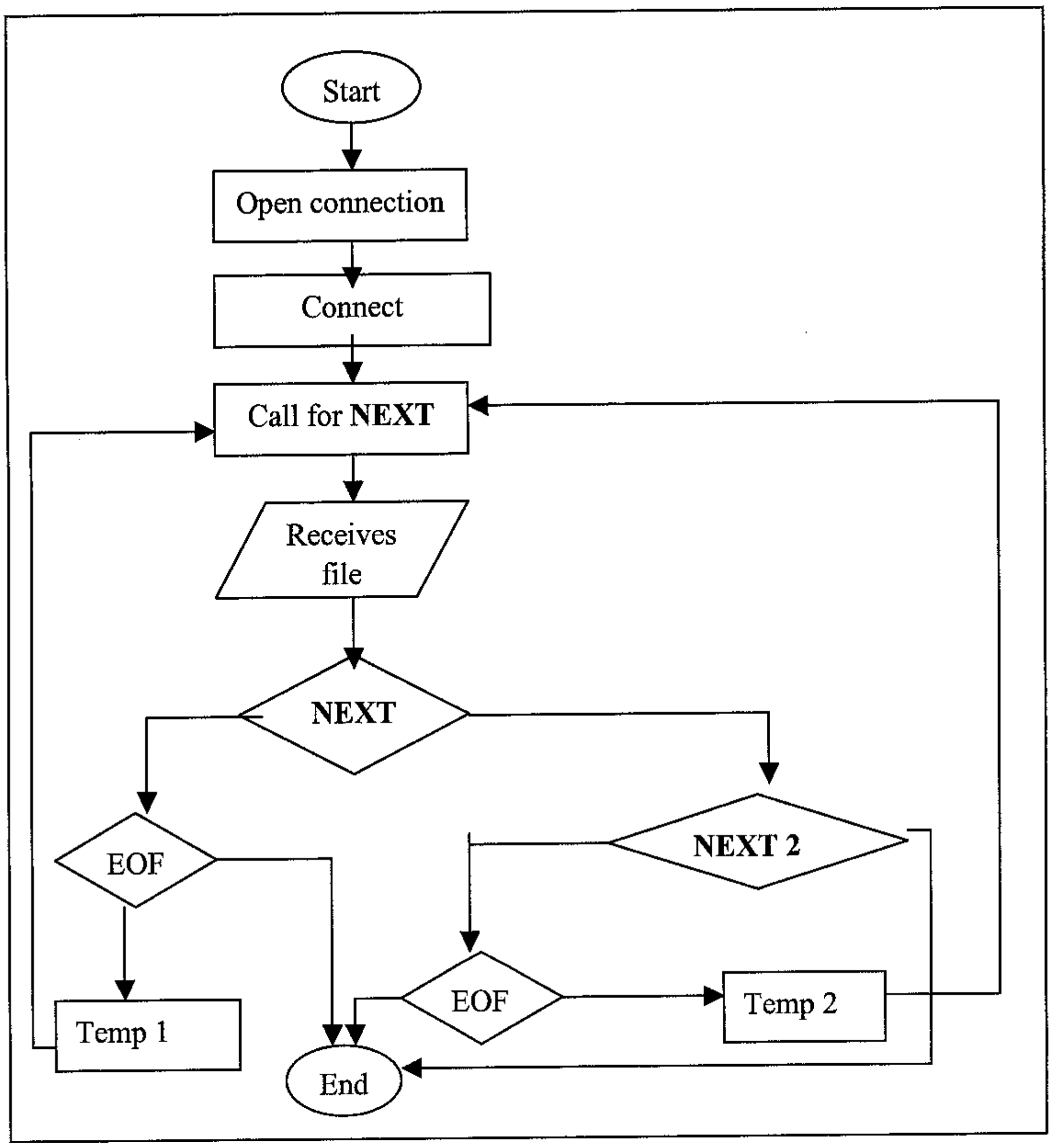


Figure 7 Transferring and streaming in client side

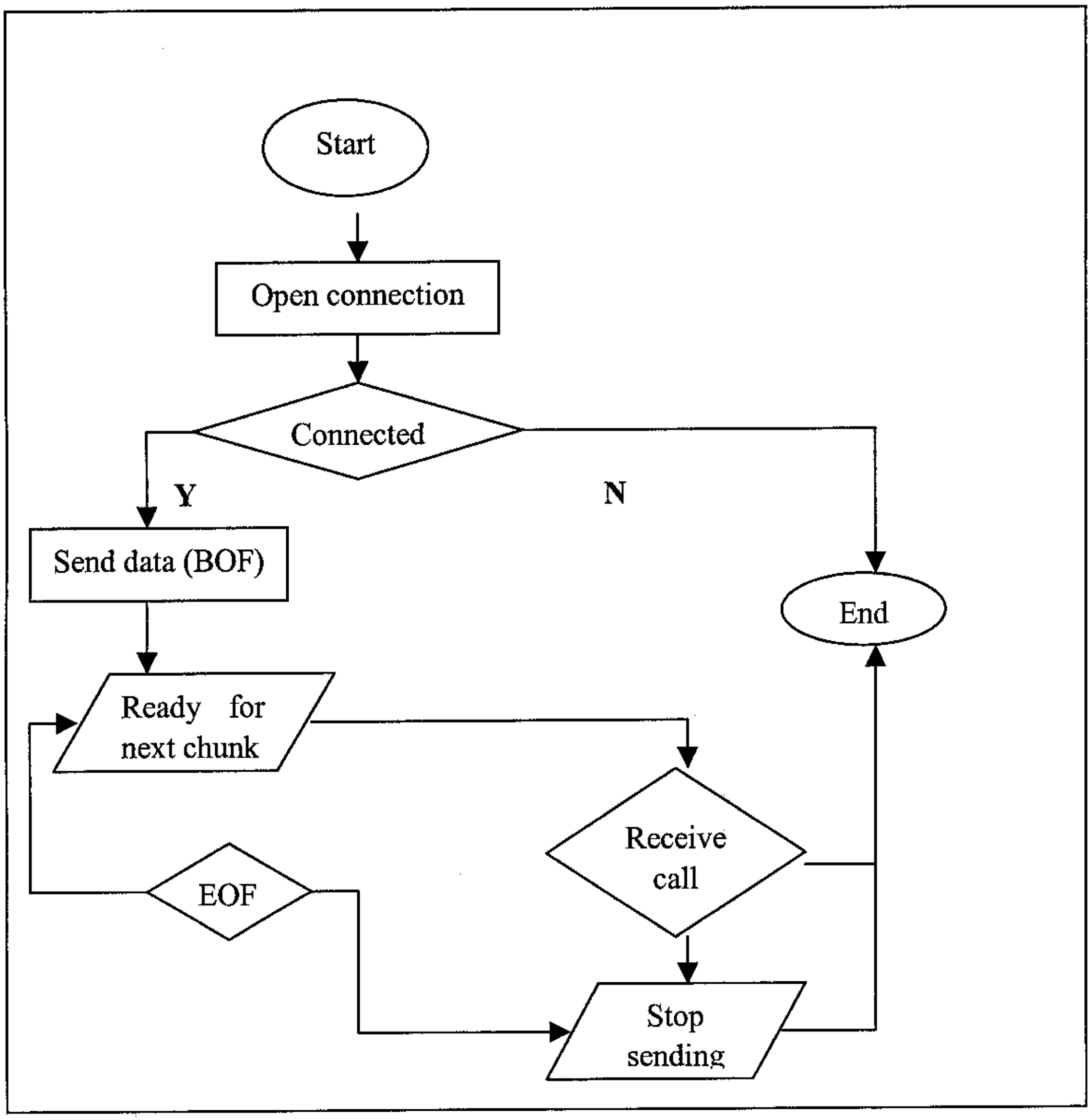


Figure 8 Transferring and streaming in server side

3.3.4 GUI

3.3.4.1 Server

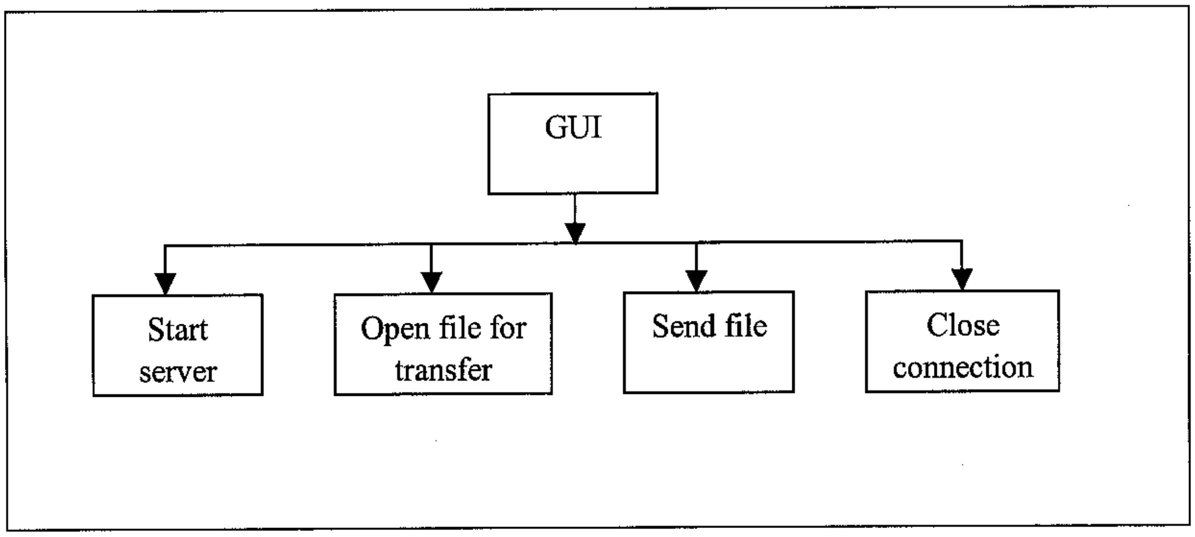


Figure 9 server GUI

In the server side, four main actions happen for GUI. It started with the start server, the command button for open a files, command button for sending a files and the close connection button, to terminate the program.

3.3.4.2 Client

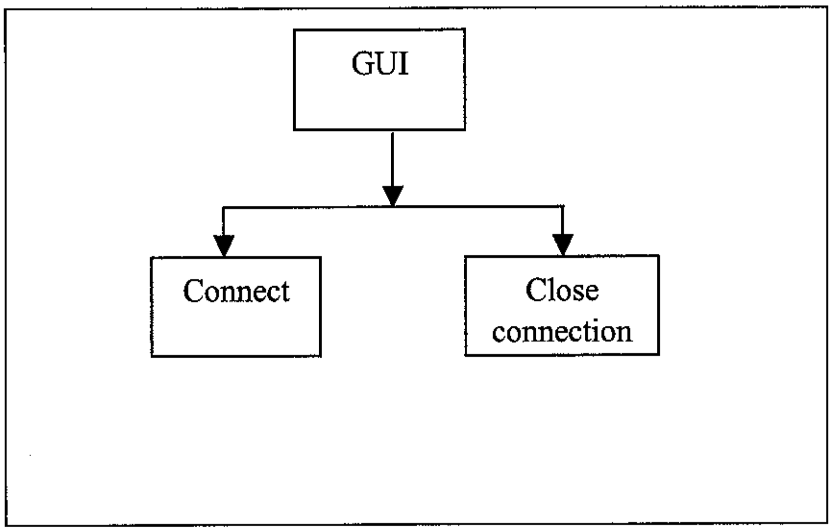


Figure 10 client GUI

In the client side, basically only two main functions happen, connect and the close command button, as in the figure3.7 above.

3.4 Coding

The coding stage will apply each of the steps in design stage and written in language that can be understood in Visual Basic. Applying the knowledge from the analysis done during the first part of the project, each function and strings in Visual Basic is applied to produce the desired result.

3.5 Testing and debugging

After completing the whole process, analyze, design and coding, testing process come into place. This is to ensure that each part of the coding process is working properly and in order. The program is tested on each stage in order to have the desired output. By doing each part separately, it is easier to test for any logical errors found in the program. Any problems encounter will be solved by experimenting with trial and error technique to make sure that the program works accordingly.

3.6 Tools Required

In this project, the main program is written in Visual Basic 6. It supports both the graphical user interface and the network programming. Among the advantages of using the VB6 is:

1. Simple programming language used.
2. Enhanced interface.
3. Supports network programming
4. Supports multimedia player.

Other tools include:

- Software Used
    1. Operating System - Windows 98 or higher
- Hardware Used
    1. Processor - Pentium 3 or higher
    2. RAM – 64 Megabyte or higher
    3. Disk space - 1 Gigabyte or higher
    4. Monitor - 14 inch with 1024 x 768 resolution
    5. Network adapters and cables – 100 Mbps
    6. Graphic card - any standard
    7. Mouse/Keyboard - any standard

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results and Discussion

In creating an audio streaming program, there are a certain stages of implementing it. Starting with the connection between two computers, the data is then transferred via TCP/IP. Instead of downloading the whole file, the file that arrived in client will be stored, played and deleted continuously to make it stream. The process continue until the whole file in strings is completed sending.

4.2 Establishing the connection

The connection between two computers is using the Winsock function. The connection established enables the two computers to communicate to each others. The connection also allows both computers to send and received data. The connection using Winsock is using a TCP/IP connection. TCP/IP is a transport layer for the computers to communicate to each other. In order for the connection to be established, the IP address must been stated first. Before connecting to the server, the client must assign which server and port that it wants to connect to. The user from client will be assign to put in any IP address which they are desire to connect with, as if they have to know what is the address of the server. IP number 127.0.0.1 and port 10101 is used as a sample, as shown.

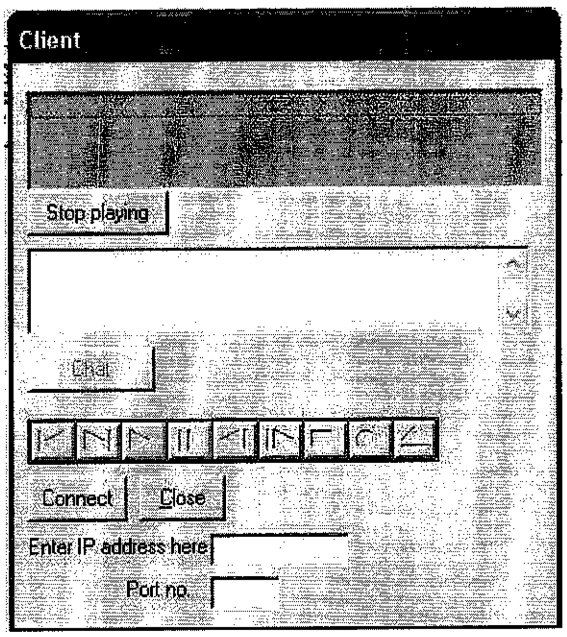


Figure 11   IP address and port to be filled in

Server will open a port connection, and listen to the incoming connection from any location to the particular IP address. An error might occur when it takes a longer time to connect. This might be an error happened in connection, or either one of the client or server is not responding.

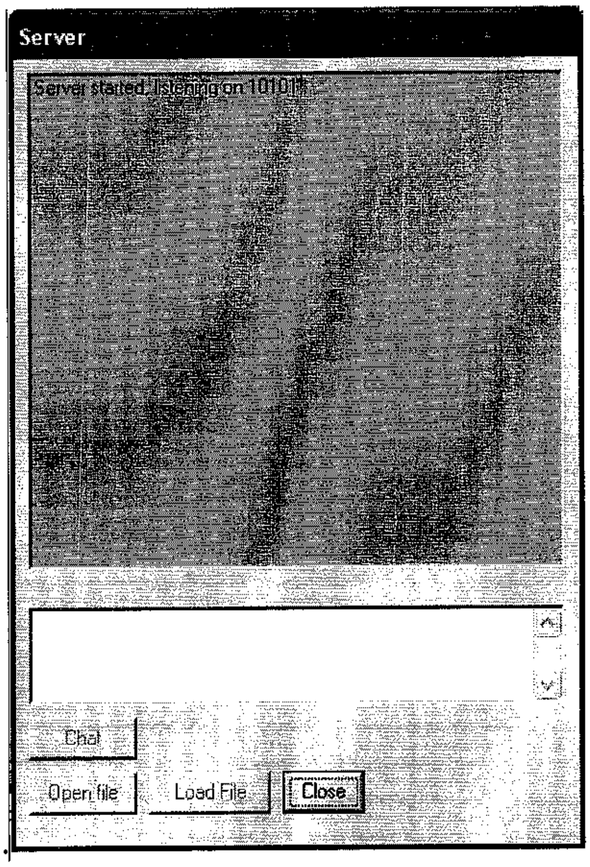
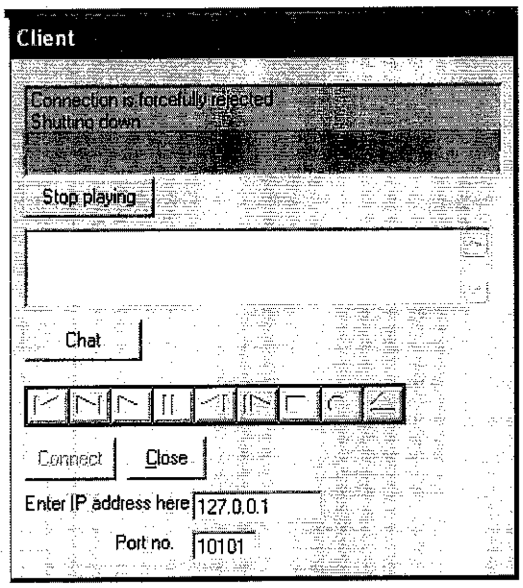

Server started. listening on 10101

Figure 12   Port opened by the server.


Connection is forcefully rejected
Shutting down

Enter IP address here 127.0.0.1
Port no. 10101

Figure 13   Error messages-connection when server is not responding

The code below explains the activity of for both client and server, when establishing the connection.

```
Private Sub StartServer()
  Call lstEvents.AddItem("Server started, listening on 10101")
  With Winsock
    Call .Close
    .LocalPort = 10101
    Call .Listen
  End With
 End Sub
```

Code 1: Server starts by listening to the port open, closes the socket and accepts the connection

```
Private Sub cmdConnect_Click()
  Call lstEvents.AddItem("Connecting to 127.0.0.1...")
  With Winsock
    Call .Close
    .RemoteHost = "" & txtIP.Text
    .RemotePort = "" & txtPort.Text
    Call .Connect
  End With
  Delay 1
  cmdConnect.Enabled = False
  Call lstEvents.RemoveItem(0)
  End Sub
```

Code 2: When connecting, client will assign the IP address and port of the server, waiting for the signal from server. If accepts, the connection established.

4.3 Data transfer

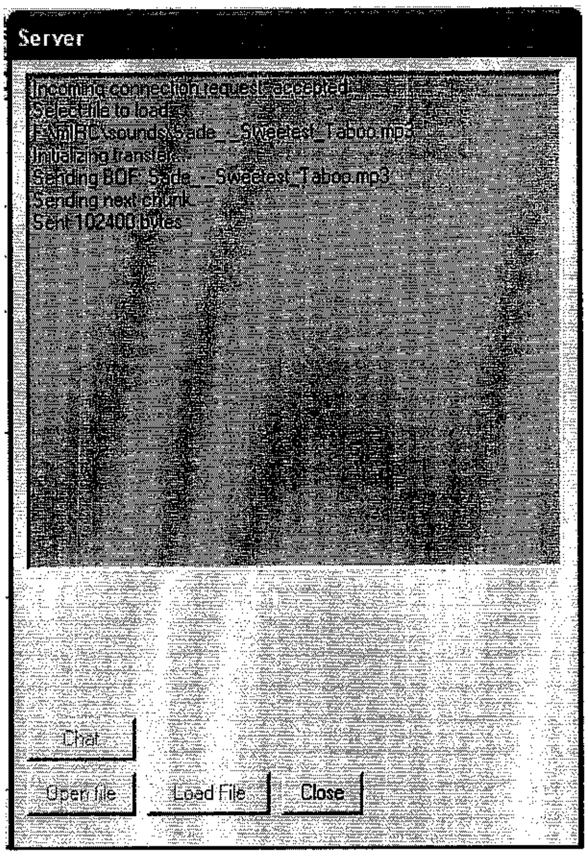
In transferring the file to the other end, Visual basic will read the filename as a set of strings. Therefore, when choosing the file, each file will be separated in three different parts, the beginning of file (BOF), middle of file and the end of file (EOF). The server will separate each of this file and starts sending to the client when acknowledged.

When the SendFile button is click, the file loaded in server will start the sending process. Starting with the BOF of the data, the rest of the file is then sending until no more data in the strings left. The code below shows the action of the SendFile button when clicked.

```
Private Sub SendFile(ByVal strFile As String)

  strFilename = strFile

  Call lstEvents.AddItem("Sending BOF: " & strFilename)

   blnTransferring = True

  Call Winsock.SendData("BOF" & strFilename)

  DoEvents

End Sub
```

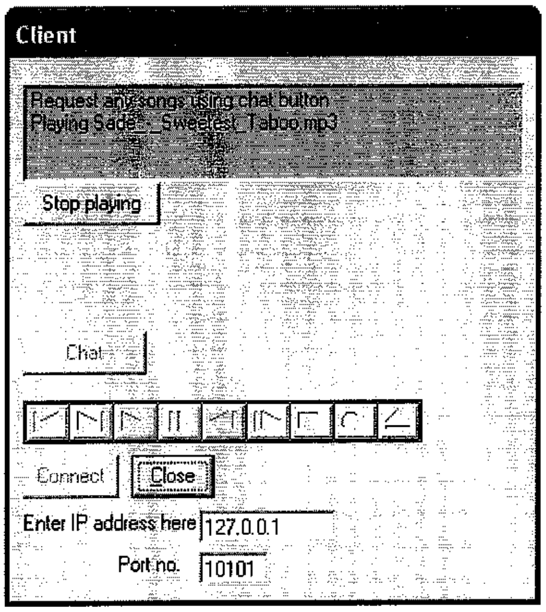
Code 3: SendFile button when clicked

In the client side, the Winsock_DataArrival will receive the incoming data from server. It will be whether BOF, middle file or the EOF. After receiving the BOF, client will request for another data until the EOF reached the client, and the process is completed.



```
Incoming connection request accepted
Select file to load
E:\MIBC\sounds\Sade_-_Sweetest_Taboo.mp3
Initializing transfer
Sending BOF: Sade_-_Sweetest_Taboo.mp3
Sending next chunk
Sent 102400 bytes
```

Figure 14 Sending of the BOF (server)



```
Request any songs using chat button
Playing Sade_-_Sweetest_Taboo.mp3
```

Enter IP address here: 127.0.0.1

Port no.: 10101

Figure 15 Sending of the BOF (client)

4.4 Chunking and streaming

The difference between the downloading and the streaming is on how the file is handled in the client side. For audio streaming the server will separate the data files first before sending it to the client, which is called chunk. The size of chunk (or packets) has to be defined first before transmitting. And then, each and every chunk will be sent one by one when until the client received the end-of-file (EOF). Whenever the client received the EOF, the server will stop sending the data and stop the transferring. The transfer is complete. The size of chunk for the file in this project is defined to be 100 kilo bytes each.

The chunking and streaming technique is the most important part of the streaming technique. In this program, each middle file will be separated into two different binary files namely temp1 and temp2. The temp file will be stored somewhere in the client folders for only a short period of time. This chunk of file will be deleted right after they are played.

For a real-time audio streaming, client will request for temp1 first, followed by the temp2. Each of this file will be played by the client before it will be deleted one by one. After both file have been deleted, the client will request for a new chunk and stored back to the temp1 or temp2 files. The process repeats. The code below explains the process.

```
hFile = FreeFile
Open App.Path & "\temp1.mp3" For Binary As #hFile
Close #hFile
i = FileLen(App.Path & "\temp1.mp3")
Debug.Print i

hFile = FreeFile
Open App.Path & "\temp2.mp3" For Binary As #hFile
Close #hFile
j = FileLen(App.Path & "\temp2.mp3")
Debug.Print j

If i < 102400 Then
hFile = FreeFile
Open App.Path & "\temp1.mp3" For Binary As #hFile
i = FileLen(App.Path & "\temp1.mp3")
Seek #hFile, LOF(hFile) + 1
Put #hFile, , strData
Close #hFile
i = FileLen(App.Path & "\temp1.mp3")
If i = 102400 Then Call deletetemp1
End If
End If



If i = 102400 Then
hFile = FreeFile
'lstEvents.AddItem ("arrived one")
```

```
Open App.Path & "\temp2.mp3" For Binary As #hFile
j = FileLen(App.Path & "\temp2.mp3")
Seek #hFile, LOF(hFile) + 1
Put #hFile, , strData
Close #hFile
j = FileLen(App.Path & "\temp2.mp3")
If j > 102400 Then Call deletetemp2
End If
End If
```

Code 4: Separating temp1 and temp2

The code above shows a process during separating the temp1 and temp2 file. First the temp1 and temp2 file is open, and then deleted right after the binary file is closed using the deletetemp1 or deletemp2 sub function. As we can see, each chunk is separated by 100 kilo bytes (102400 bytes each).

The playchunk and playchunk2 private sub function is actually played the media file using a multimedia control declares in VB. It is the same process control, but played a different file, temp1 and temp2 respectively. In this case, the multimedia control will play the mp3 files created during the chunking process.

```
Private Sub playchunk()
    MMC.FileName = App.Path & "\temp1.mp3"
    MMC.Command = "open"
    MMC.Command = "play"
    Delay 7
    MMC.Command = "close"
    DoEvents
    Call Winsock.SendData("NEXT")
End Sub

Private Sub playchunk2()
    MMC.FileName = App.Path & "\temp2.mp3"
    MMC.Command = "open"
    MMC.Command = "play"
    Delay 7
    MMC.Command = "close"
    DoEvents
    Call Winsock.SendData("NEXT")
End Sub
```

Code 5: The process of the playchunk and playchunk2

Full source code can be view in appendix A.

4.5 GUI

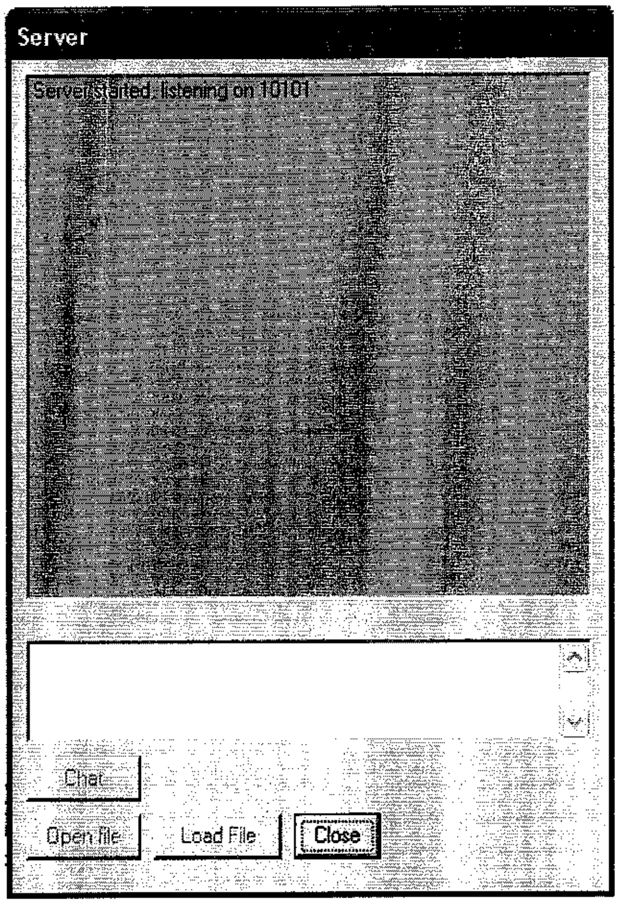
They are two main windows defined in the program:

1. Server
2. Client

4.5.1 Server

The main operation performed by the server is:

1. Open file for transfer
2. Send file or start loading button
3. Close button
4. List of events to show user what is happening during the transfer



Server Started, listening on 10101

Figure 16 GUI of server

4.5.2 Client

The main operation performed by the client is:

1. Connect button for retrieving connection.
2. Close button
3. List of events to show user what is happening during the transfer.

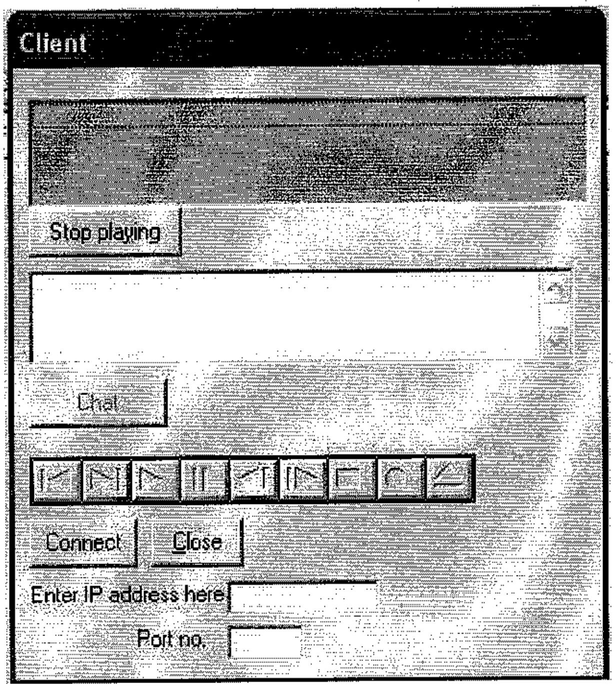


Figure 17 GUI of the client

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In a nutshell, the project undertaken is a success as it corresponds to the initial objectives despite some pinch faced along the way. It is duly noted at this stage that time has proven to be a major factor in completing all the tasks and troubleshooting phases. This project showcases the integration of network programming and the graphical user interface in implementing software to produce a real-time digital audio streaming using the network connection. Visual Basic plays an important role in applying both of the procedure. It supports the GUI capabilities, network programming via Winsock. It also supports the capability of playing an mp3 files using the multimedia component.

Looking from the economic perspective, this project is definitely cost effective as it comprises of only one programming language that are generally low in terms of cost. Furthermore, they are also widely available in the market. This certainly proves to be a boost as this system is not only practical, but also competitive in performance and cost. Though this system is small scale at this stage, with some improvisations, it can adapt and be implemented in various application of the global internet world. The world of entertainment seems to be no red light in applying the audio streaming program.

5.2 Recommendation

As to end the project presentation, it is therefore important to suggest some improvements and recommendations for the benefit of the project. In any cases at all, engineers are known to improvise and modify to better the system. As mentioned earlier in the discussion, this project can be adapted to suit different applications.

Therefore, the recommendations suggested are as follows:

1. Enhanced the capability of the program to be used for a multiple computer connection.
2. Applying the tag for the play list file for easy reference.
3. Increasing the graphical interface used such as mini web browser, and changing color scheme.
4. Adding the equalizer to enhance the output sound.
5. Real-time lyrics browser, so listener can sing along with the songs played.
6. Upgrading the design of the Visual Basic GUI interface such as introducing new features which may include a better visual of the secured parameter and more user-friendly functions.

All these recommendations will definitely improve the current system into a more powerful real-time digital audio streaming.

REFERENCES

1. Steven Holzner; *Visual Basic 6 – Black Book Comprehensive Problem Solver;* Coriolis; United States of America;1998
2. Jose Alvear ; *Guide To Streaming Multimedia,* 1998
3. David I. Schneider; *An Introduction to Programming Using Visual Basic 6.0.;* Fourth edition; Prentice-Hall; 1995
4. Viktor Toth; *Visual C++ 5;* Second Edition; Sams Publishing ;United States of America; 1997
5. Wallace Wang,; *Visual Basic 6 for Dummies*; IDG Books Worldwide; 1998
6. Evangelos Petroutsos; *Database Programming with Visual Basic 6*; SYBEX Inc.; United States of America; 2000
7. Evangelos Petroutsos; *Visual Basic 6 Developer's Handbook;* Sybex Inc.; 1998
8. Rod Stephens; *Ready-to-Run Visual Basic Code Library*; 1999
9. www.juicystudio.com
10. www.aivosto.com
11. www.experts-exchange.com
12. www.tutorilaized.com
13. www.officecomputertraining.com
14. www.winsockvb.com
15. www.howstuffworks.com

APPENDICES

APPENDIX A: FULL SOURCE CODE OF AUDIO STREAMING PROGRAM

FOR SERVER

```
Option Explicit
Private Const CHUNK_SIZE As Long = 102400 ' 100KB.
Private blnTransferring  As Boolean
Private lngFilePos       As Long
Private strFilename      As String
Private Sub cmdOpen_Click()
   comdlg.ShowOpen
   Call lstEvents.AddItem(comdlg.FileName)
   cmdSendFile.Enabled = True
   Call lstEvents.RemoveItem(0)
End Sub
```

```
Private Sub Form_Load()
  Call StartServer
  cmdOpen.Enabled = False
End Sub
Private Sub cmdClose_Click()
  Call Unload(Me)
End Sub
Private Sub cmdSendFile_Click()
  Call lstEvents.AddItem("Initializing transfer...")
  Call SendFile(comdlg.FileName)
  cmdOpen.Enabled = False
  cmdSendFile.Enabled = False
End Sub
  Call lstEvents.AddItem("Server started, listening on 10101")
  With Winsock
    Call .Close
    .LocalPort = 10101
    Call .Listen
  End With
End Sub
Private Sub SendFile(ByVal strFile As String)
  strFilename = strFile
  Call lstEvents.AddItem("Sending BOF: " & strFilename)
  blnTransferring = True
  Call Winsock.SendData("BOF" & strFilename)
  DoEvents
End Sub
```

```
Private Sub SendNextChunk()
Dim hFile        As Long
Dim lngchunksize  As Long
Dim strdata       As String
  If (Not blnTransferring) Then Exit Sub
  hFile = FreeFile
  Open strFilename For Binary As #hFile
  If (lngFilePos = 0) Then lngFilePos = 1
  Seek hFile, lngFilePos
```

```
 lngchunksize = LOF(hFile) + 1 - lngFilePos
   If (lngchunksize > CHUNK_SIZE) Then lngchunksize = CHUNK_SIZE
   If (lngchunksize = 0) Then
     strdata = "EOF"
     blnTransferring = False
     Call lstEvents.AddItem("0 bytes, transfer completed. Sending EOF.")
     Call Winsock.SendData(strdata)
    DoEvents
   Else
     strdata = String$(lngchunksize, 0)
     Get #hFile, , strdata
     lngFilePos = lngFilePos + lngchunksize
     Call Winsock.SendData(strdata)
     Call lstEvents.AddItem("Sent " & lngchunksize & " bytes")
     DoEvents
   End If
Close #hFile
    End Sub
```

FOR CLIENT

```
Option Explicit
Private m_blnTransferring  As Boolean
Private m_strFilename      As String
Private m_blntemp2 As Boolean
Private Sub cmdConnect_Click()
  Call lstEvents.AddItem("Connecting ...")
  With Winsock
    Call .Close
    .RemoteHost = "" & txtIP.Text
    .RemotePort = "" & txtPort.Text
    Call .Connect
  End With
  Delay 1
  cmdConnect.Enabled = False
  Call lstEvents.RemoveItem(0)
End Sub
Private Sub cmdClose_Click()
  Call Unload(Me)
End Sub
```

```
Private Sub Winsock_DataArrival(ByVal bytesTotal As Long)
Dim strData    As String
Dim hFile      As Long
Dim i As Long
Dim j As Long
Dim hchunk As Long
  Call Winsock.GetData(strData)
  If (Not m_blnTransferring) Then
    If (Mid$(strData, 1, 3) = "BOF") Then
      Call lstEvents.AddItem("Received BOF - incoming file:" & Mid$(strData, 4))
      m_blnTransferring = True
      m_strFilename = Mid$(strData, 4)
      Call lstEvents.AddItem("Requesting first piece")
      Call Winsock.SendData("NEXT")
      DoEvents
    End If
  Else
      If (Mid$(strData, 1, 3) = "EOF") Then
       'The transfer is complete.
      m_blnTransferring = False
      Call lstEvents.AddItem("Received EOF, transfer complete")
      DoEvents
```

```
Else
   hFile = FreeFile
   Open App.Path & "\temp1.mp3" For Binary As #hFile
   Close #hFile
   i = FileLen(App.Path & "\temp1.mp3")



   hFile = FreeFile
   Open App.Path & "\temp2.mp3" For Binary As #hFile
   Close #hFile
   j = FileLen(App.Path & "\temp2.mp3")
   Debug.Print j

   If i < 102400 Then
   hFile = FreeFile
  Open App.Path & "\temp1.mp3" For Binary As #hFile
   i = FileLen(App.Path & "\temp1.mp3")
   Seek #hFile, LOF(hFile) + 1
   Put #hFile, , strData
   Close #hFile
   i = FileLen(App.Path & "\temp1.mp3")
   If i = 102400 Then Call deletefirst
   End If
   End If
```

```
If i = 102400 Then
hFile = FreeFile
Open App.Path & "\temp2.mp3" For Binary As #hFile
j = FileLen(App.Path & "\temp2.mp3")
Seek #hFile, LOF(hFile) + 1
Put #hFile, , strData
Close #hFile
j = FileLen(App.Path & "\temp2.mp3")
If j > 102400 Then Call deletetemp
End If
End If
End Sub
```

```
Private Sub Winsock_Error(ByVal Number As Integer, Description As String, ByVal Scode .
Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Lo
CancelDisplay As Boolean)


 Call lstEvents.AddItem(Description)
 Call lstEvents.AddItem("Shutting down")
 Call Winsock.Close
 m_blnTransferring = False


End Sub


Private Sub playchunk()
  Dim count As Integer
  MMC.FileName = App.Path & "\temp1.mp3"
  MMC.Command = "open"
  MMC.Command = "play"
  Delay 7
  MMC.Command = "close"
  DoEvents
  Call Winsock.SendData("NEXT")
End Sub
Private Sub deletetemp()
    Dim FileSystemObject As Object
  Set FileSystemObject = CreateObject("scripting.Filesystemobject")
  FileSystemObject.deletefile App.Path & "\temp1.mp3"
  DoEvents
End Sub
```

```
Sub Delay(ByVal Seconds As Double)
 Dim StartTime As Double
 StartTime = Timer
  Do
  Debug.Print Timer
     DoEvents
     Debug.Print Seconds
  Loop Until Timer > StartTime + Seconds

End Sub
Private Sub playchunk2()
  MMC.FileName = App.Path & "\temp2.mp3"
  MMC.Command = "open"
  MMC.Command = "play"
  Delay 7
  MMC.Command = "close"
  DoEvents
  Call Winsock.SendData("NEXT")
End Sub
Private Sub deletefirst()
  Dim FileSystemObject As Object
  Set FileSystemObject = CreateObject("scripting.Filesystemobject")
  FileSystemObject.deletefile App.Path & "\temp2.mp3"
  DoEvents
  Call playchunk
 End Sub
```