Journal of Engineering and Applied Sciences *Volume 3, December 2011*

© 2011 Cenresin Publications
www.cenresinpub.org

A CONCURRENT PC – to – PC INTERACTIVE TELEPHONY

[1]*Japheth B. R. and* [2] *Bubou G. M.*
[1]*Department of Mathematics/Computer Science, Niger Delta University, Yenagoa, Nigeria*
[2] *National Centre for Technology Management, South-South Zonal Office Niger Delta University, Yenagoa*
jbunakiye@yahoo.com; gbubou@gmail.com

ABSTRACT

This paper presents software developed using the Visual Basic Integrated Development Environment capable of transmitting voice over an internet protocol. It is being implemented by running the software in two end computers connected over the internet. The Internet Telephony software project is a platform – independent application; meaning that it can run on different network environments, ranging from LAN, MAN, WAN, and the Internet. As long as the *ping.exe* utility gives a *ping* response from the remote host ID, there will be a voice communication anywhere in the world. The software utilizes the socket facility of the system's operating system to connect to the TCP/IP network. Computers using Windows, connected to these network environments can use it. This PC – to – PC telephony software works, however only if the two parties are using the software. It compresses the voice signal and translates it into an IP packet for transmission over the network.

Key words: Interactive, Telephony, Internet Protocol, PC- to - PC, Remote Host ID, Socket Facility.

NTRODUCTION

Internet Protocol (IP) telephony is the emerging communications technology arising from the convergence of the worldwide data infrastructure and the telecommunications network. It is about transmitting voice information across an Internet Protocol network, for example the Internet. Hence, it is also called Internet Telephony; a technology that allows one to make telephone calls using a broadband connection instead of a regular phone line. Until now, the voice world and the data world have been largely separate. Voice traffic moves on circuit switched networks where a fixed path is established at call setup and maintained for the duration of the call. Data traffic has moved on packet switched networks where messages or files are broken up into small packets, and each packet is sent out onto the network on its own, typically with header information containing the needed destination address and other descriptors. The series of packets moves through the network individually, and different packets may take different paths through the mesh of routers and switches. Packets do not necessarily arrive at the destination in the same order they left the sender, and some packets may be lost along the way. Now, the latest hype in the industry is the convergence of the voice and data transmission systems. The convergence states that voice will look like data, and the two can reside in packetized form on the same network.

BACKGROUND AND RELATED WORKS

The development of internet telephony system of this kind that establishes a reliable connection between the calling entities i.e. the computers can [1, 2] be likened to public switched telephone systems that were primarily used for voice traffic with a little bit of data traffic here and there. But the data traffic grew and grew, and by 1999, the number of data bits moved equaled the number of voice bits. By 2003, the volume of data traffic was an order of magnitude more than the volume of voice traffic and still growing exponentially, with voice traffic being almost flat (5% growth per year). As a result of this, many packet switched network operators suddenly became interested in carrying voice over their data networks. However, the average person's phone bill is probably larger than his internet bill, so the data network operators saw internet telephony as a way to earn a large amount of money without having to put any new fiber in the ground. Thus Internet telephony was born [3, 6, and 9].

THE LAYERED ARCHITECTURE REFERENCE MODEL

The Internet telephony software application presented in this paper is suited in a design paradigm that entails a communication system that can provide a call alert prompting telephone ringing or an audio alert and so giving the user a choice to either accept the call or not, thus tended towards identified reference models. A reference model is a conceptual blueprint of how communications should take place. It addresses all the processes required for effective communication and divides these processes into logical groupings called layers. When a communication model is designed in this manner, it is known as Layered Architecture [10, 13]. This reference model is used to understand the computer communication processes and see what type of functions need to be accomplished on any one layer. For example, in the process of 'binding' the specific layer's functionality of a protocol for a certain layer is considered in the development of the protocol and not all the layers. The communication process that is related to each other is bound at a particular layer [11, 5].

THE OSI REFERENCE MODEL

The OSI reference model [3] was created to help facilitate data transfer between network nodes. A set of guidelines that application developers can use to create and implement application that run on a network. It provides the framework for creating and implementing this networking standard and internetworking scheme. The main reason the ISO released the OSI model was to enable different vendor networks work with each other. The OSI model has 7 different layers divided into two groups. The upper four layers are used whenever a message passes from or to a user. The lower three layers are used when any message passes through the host computer. Messages intended for this computer pass to the upper layers. Messages destined for some other host are not passed up to the upper layers but are forwarded to another host. The seven layers [4, 3, and 11] are:

Layer 7 The application layer: This is the layer at which communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified.

Layer 6 The presentation layer: This is a layer, usually part of an operating system that converts incoming and outgoing data from one presentation format to another for example, from a text stream into a popup window with the newly arrived text.

Layer 5 The session layer: This layer sets up, coordinates, and terminates conversations, exchanges, and dialogs between the applications at each end. It deals with session and connection coordination.

Layer 4The transport layer: This layer manages the end-to-end control for example, determining whether all packets have arrived and error-checking. It ensures complete data transfer.

Layer 3The network layer: This layer handles the routing and forwarding of data by sending the data in the right direction to the right destination on outgoing transmissions and receiving incoming transmissions at the packet level.

Layer 2 The data-link layer: This layer provides synchronization for the physical level and furnishes transmission protocol knowledge and management.

Layer 1The physical layer: This layer conveys the bit stream through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier.

ANALYSIS OF INTERNET TELEPHONY PROTOCOLS

The internet telephone network [9] needs a number of protocols which define ways for devices to connect to each other and include specifications for audio codecs. A codec (coder – decoder) converts an audio signal into a compressed digital form for transmission and back into an uncompressed audio signal for replay. One thing that was clear to everyone from the start was if each vendor designed its own protocol stack, the system would never work. To avoid this problem, a number of interested parties got together under ITU auspices to workout standards [11, 12, and 13].

TCP/IP PROGRAMMING IN COMPUTER COMMUNICATIONS

With the acceptance of TCP/IP as a standard platform independent network protocol and the explosive growth of the internet, the Windows Sockets Application Programming Interface (WINSOCK API) has emerged as the standard for network programming in the Windows environment. TCP defines how data are transferred across the internet to their destinations. IP defines how data are divided into chunks called packets for transmission; it also determines the path each packet takes between computers [1, 4, and 7]. When two computers need to exchange information over a network, there are several components that must be put in place before the data can actually be sent or received, of course, the physical hardware must exist which is typically either a network interface card (NIC) or a serial communication port for dial up networking connections. Beyond this physical connection, however, computers also need to use a protocol, which defines the parameters of the communication between them. In short, protocol defines the set of rules that each computer must follow so that all the systems in the network can exchange data. By convention, TCP/IP is used to refer to a suite of protocol, all based on the internet protocol (IP). A reliable straightforward way to exchange data without been concern about data integrity was needed. TCP was developed to provide this need. Built on top of IP; TCP offer a reliable, full-duplex byte stream that may be read and written to in a fashion similar to reading and writing a file. TCP is known as a connection – oriented protocol. In other words, before two programs can begin to exchange data, they must establish a connection with each other. This is done with a three way handshake in which both sides exchange packets and establish the initial packet sequence number (sequence number is important because datagram can arrive out of order). When establishing a connection one program must assume the role of a client and the other a server. The client is responsible for initializing the connection while the server's responsibility is to wait, listen and respond to incoming connection, once the connection has been established; both sides may send and receive data until the connection is terminated [2, 8, and 5].

H.323 MULTIMEDIA COMMUNICATIONS SYSTEMS

The IP telephony architecture must emulate the PSTN to accommodate the various ways of interconnecting. H.323 is a mechanism for the manufacturers to produce the necessary devices and protocol compatibility for interoperability in the PSTN and the Internet convergence. It consists of a set of standards to provide support for real-time multimedia communications on LANs and other packet networks. ITU-T recommendation H.323 evolved out of the H.320 videoconferencing standards for ISDN. Its terminals and equipment can carry voice, video, data, or any combination of these. In particular, it provides a means for interconnecting telephone-based and IP-based conferencing. As shown in figure 1, an H.323 network involves several components; the terminals, the gateways, the gatekeepers, and multipoint control units [3, 5, and 8].

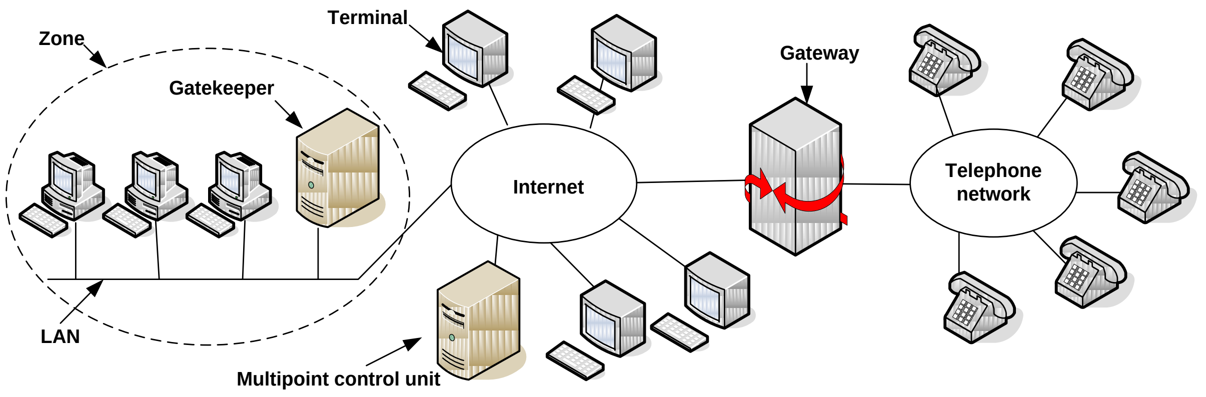


Figure 1: The H.323 architectural model for internet telephony

The Terminals: They are client end points on IP based network that provides real-time, two-way communication with other H.323 entities.

The Gateways: The gateways provide the connection path between the terminals in the packet network and the circuit switched network. The gateway is responsible for the translation between audio and video codec formats, mapping of signaling messages from the packet side of the network to other networks. In telephone applications the gateway performs call setup between the packet network and the public telephone network. Part of the call setup involves establishing a path for the call across the gateway.

The Gatekeepers: Optional on the H.323 system functions. The gatekeepers are responsible for call control for calls within an H.323 network. Gatekeepers grant permission or deny requests for connections. They manage the bandwidth that can be used by calls and perform name-to-address translation. They also direct calls to appropriate gateways when necessary.

Multipoint Control Units: Terminals in a multipoint conference can send audio and video directly to other terminals using multicasting. They can also use multipoint control units that can combine incoming audio streams and video streams and transmit the resulting streams to all terminals [1, 8, and 9].

THE WINDOWS SOCKET API

An Application Programming Interface (API) allows application programs (such as this real time interactive internet telephony system presented in this paper) to access certain resources through a predefined and preferably consistent interface. One of the most popular of the APIs that provides access to network resources is the Berkeley socket interface, which was developed by a group at the University of California at Berkeley in the early 1980s. Another popular socket interface, which was derived from the Berkeley socket interface, is called the Windows sockets or Winsock and was designed to operate in a Microsoft Windows environment [6, 9, and 10]. The socket mechanism allows programmers to write application programs very easily without worrying about the networking details. Figure 2 shows how two applications talk to each other across the communication network through the socket interface.

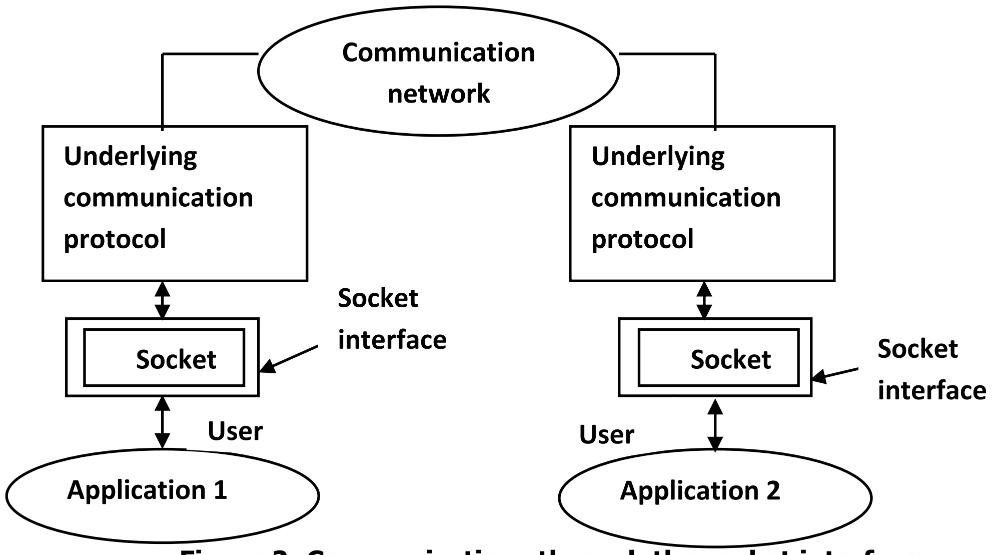


Figure 2: Communications through the socket interface

The Windows socket specification was created in an effort to standardize the TCP/IP suite of protocols under windows. Prior to Windows sockets, each vendor developed their own proprietary libraries and although they all have similar functionality, the differences were significant enough to cause problems for the software developers that use them. A program written against one vendor's product would not work with another. Windows socket was offered as a solution, leaving developers with their end-user free to choose any vendor's implementation with the assurance that the product will work. This standard was utilized in this application enabling its deployment in diverse vendor equipment. The two general approaches to Windows sockets programming were used i.e. coding directly against the API, and then use a component that provides a higher level interface to the library by setting properties and responding to events. This can be provided in a more natural programming interface, and it aids the avoidance of the error prone rigid work commonly associated with socket programming [1, 7, and 9].

SOCKETS

The communication structure that we need in socket programming is a socket. A socket [11] acts as a communication end point that can be likened to a telephone. However, creating a socket by itself does not let you exchange information, just like having a telephone in your house does not mean that you can talk to someone by simply taking it off the hook. You need to establish a communication with the other program, to do this, you need a socket address of the application that you want to connect to. This address consists of three parts: the protocol family, IP address and the service port number. The protocol family is a number which is used to typically designate the group that a given protocol belongs to. Since the socket interface is generally enough to be used with several protocols, the protocol family tells the underlying network software which protocol is being used by the socket. In our case, the internet protocol family will always be used when creating sockets. With the protocol family, IP address of the system and the service port number for the program that you want to exchange data with, are ready to establish a connection. Figure 3 shows a simplified version of a socket structure with five fields [12].

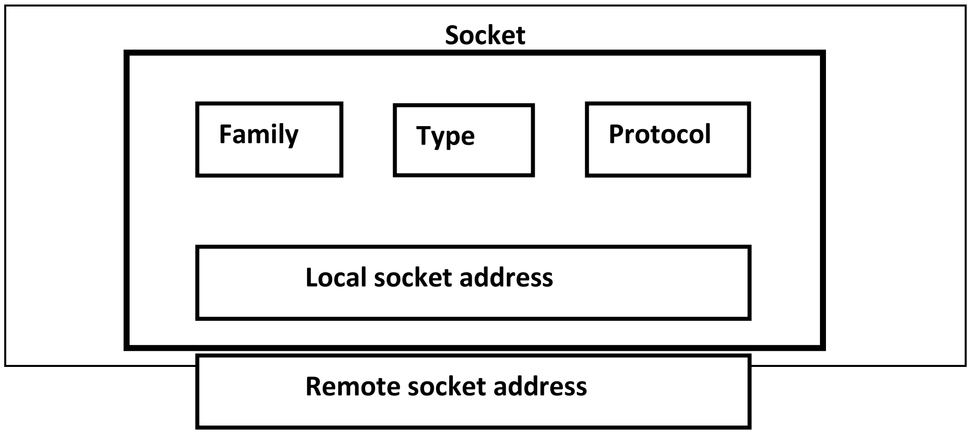


Figure 3 Socket Structure

The Family field defines the protocol group: IPv4, IPv6, UNIX domain protocols, and so on. The Type field defines the type of socket, which includes stream socket, packet socket or raw socket. All three types can be used in a TCP/IP environment. The stream socket is designed to be used with a connection-oriented protocol such as TCP. TCP uses a pair of stream sockets to connect one application program to another across the internet. The packet socket is designed to be used with a connectionless protocol such as UDP. UDP uses a pair of datagram sockets to send a message from one application program to another across the internet. *The Raw socket* is designed for use by such protocols that directly use the services of IP. The Protocol field is usually set to zero for TCP and UDP. Local socket address is a combination of the local IP address and the port address of the local application program. Remote socket address; is a combination of the remote IP address and the port address of the remote application program[5, 8, and 4].

ANALYSIS OF THE CONCURRENT INTERACTIVE TELEPHONY SYSTEM

This Internet Telephony software project is a Windows based application. It links with the TCP/IP network using the socket facility of the system's operating system. Windows Socket (Winsock) application programming interface provide communication between PCs connected over an IP network – the internet. The PCs are equipped with sound card, network interface card, integrated microphone and speakers. The software packetizes the analog voice into digital form, compresses the digital audio stream, and translates the wave-audio stream into an IP packet. The Winsock control provides access to the TCP network which is used for packet transmission over the network in real-time. The process is reversed at the other end for playback. This PC- to-PC telephony software works, however only if both parties are using the software. The Internet Protocol (IP) controls the access to and transfer of data over the internet in a client – server model, which means that the IP telephony programs are actually the entities that communicate with each other and not, the PCs or users.

DESCRIPTION OF THE DEPLOYMENT SCENARIO

Figure 4 shows the deployment scenario of the application as operated over an IP network in a PC – to – PC dialogue. The mode of deployment including the skeletal description employs the WinSock approach. A WinSock control allows you to connect to a remote machine and exchange data using the Transmission Control Protocol (TCP/IP). The protocol can be used to create client and server applications. Like the Timer control, the WinSock control doesn't have a visible interface at run time. When using the WinSock control, the first consideration is whether to use the TCP or the UDP protocol. The major difference between the two lies in their connection state: In this case TCP protocol control is used. It is a connection-based protocol, and is analogous to a telephone and certainly the user must establish a connection before proceeding. The Winsock control, invisible to the user, provides easy access to network services. It can be used by Microsoft Access, Visual Basic, or Visual C++ developers. To write client or server applications you do not need to understand the details of TCP or to call low level Winsock APIs. By setting properties and invoking methods of the control, you can easily connect to a remote machine and exchange data in both directions. The Transfer Control Protocol allows you to create and maintain a connection to a remote computer. Using the connection, both computers can stream data between themselves. If you are creating a client application, you must know the server computer's name or IP address (**RemoteHost** property), as well as the port (**RemotePort** property) on which it will be "listening." Then invoke the **Connect** method. If you are creating a server application, set a port (**LocalPort** property) on which to listen, and invoke the **Listen** method. When the client computer requests a connection, the ConnectionRequest event will occur. To complete the connection, invoke the **Accept** method within the ConnectionRequest event. Once a connection has been made, either computer can send and receive data. To send data, invoke the **SendData** method. Whenever data is received, the DataArrival event occurs. Invoke the **GetData** method within the DataArrival event to retrieve the data.

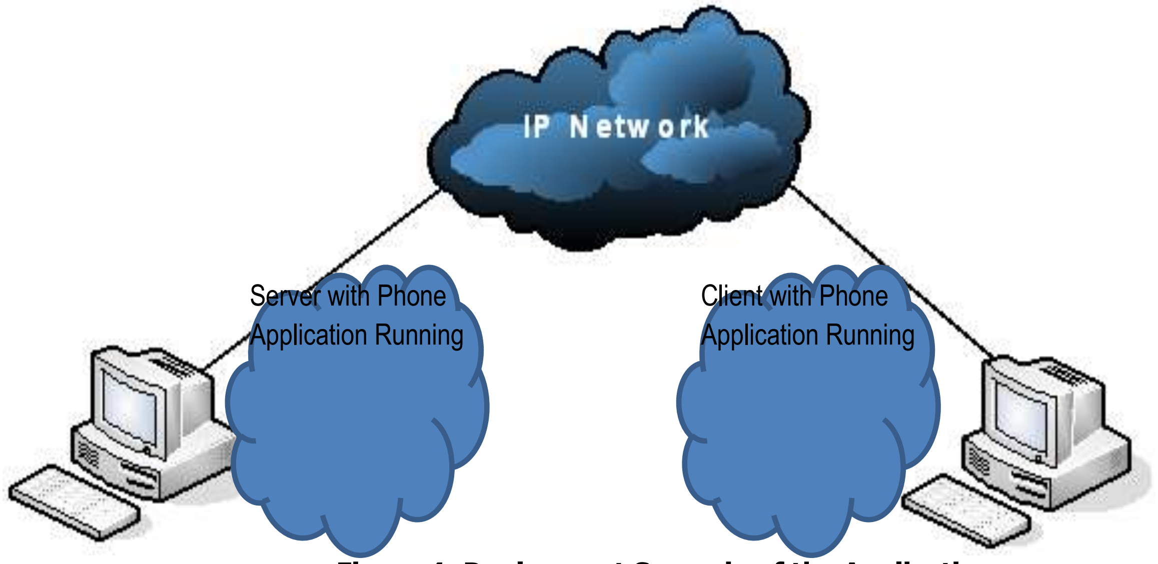

IP Network
Server with Phone Application Running
Client with Phone Application Running

Figure 4: Deployment Scenario of the Application

BANDWIDTH UTILIZATION

It has been determined that during a conversation, only about 25 percent of the circuit time is actually utilized to carry the voice. The rest of the time, are in idle condition either by listening to the other end, thinking of a response to a question, or by breathing between our words. In this idle capacity, the compression of voice stream can facilitate less circuit usage and encourage the use of a packetized form of voice. The use of a packet switching transmission system enables users to interleave voice, video and data packets where there is idle space. As long as a mechanism exists to recoup the information and reassemble it on the receiving end, it can be a more efficient use of bandwidth. It is just this bandwidth utilization and effective saving expectations that have driven the world into frenzy over packetizing voice and interleaving it on a data network, especially the Internet. The concept is to use the Internet for voice communications in a PC to PC dialogue [3,6]. The IP packet networks were developed primarily for the transport of data information that did not have stringent timing requirements. Advances in compression algorithms and computer processing power combined with improvements in transmission bandwidth and packet switching capability are making it possible to support real-time voice communications over packet networks. Advances in computer processing in particular make it possible for a wide range of media types to coexist in multimedia applications. Real-time packet communications make it possible to exploit the ubiquity of the internet. The basic processes involved in creating IP packets [10] are as follows:

1. An analog voice signal is converted to a linear pulse code modulation (PCM) digital audio stream
2. Compression of the resulting PCM digital audio stream
3. Translation of the digital audio stream into an IP packet for transmission
4. When the destination receives the packets, the process is reversed at the other end for playback. The receiving end must reassemble the packets in their correct order when they arrive out of order to create voice

ALGORITHM FOR THE INTERACTIVE TELEPHONY SYSTEM

An algorithm is a finite sequence of effective statements that when applied appropriately will realize the objectives of the application. Each algorithm developed should have a specific beginning; at the completion of one step have the next step uniquely determined and have an ending that is reached in a specific amount of time. The algorithm is stated below.

```
Start
Create a socket
Invoke a Listen method
Establish a connectionRequest event
Invoke the Accept method
Call the wavestreaming DLL
Activate audio recording/playback device
Invoke the SendData method
Establish the DataArrival event
Invoke the GetData method
Establish playback procedure
Terminate connection and destroy the socket
End
```

SYSTEM TESTING AND IMPLEMENTATION

This involves running the software in two computers connected in the internet after it has been packaged and deployed. This is to confirm the interaction between the software running in the two end computers and other designed objectives of the software program. The main consideration in the implementation is to ensure the reliability and scalability of the software application. This Internet telephony software is a window based application and was implemented using the Microsoft Visual Basic Integrated Development Environment. The software runs in the Windows operating system for computers connected in the internet in a PC – to – PC dialogue. One computer running the software is the client that requests a service; the other is the server that provides a service. The software utilizes the TCP/IP network as a medium to ensure that there was interaction and communication between the software running on the two end computers and activates the device drivers for the network interface card and the sound card which provides the capability for data and voice transmission. Each host is assigned a static IP address to identify it on the network and a service port number to identify the individual software running on each host.

Creating a Socket

To create a Socket the following module was designed for that purpose.

```
Private SubForm_Load ( )
‘Set the LocalPort property to an integer.
‘Then invoke the Listen Method.
tcpServer.LocalPort = 701
tcpServer.Listen
frmClient.show ‘Show the client form.
End Sub.
```

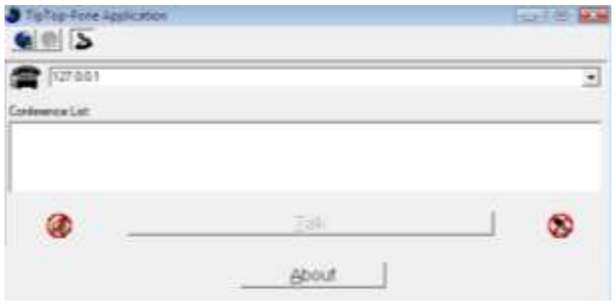
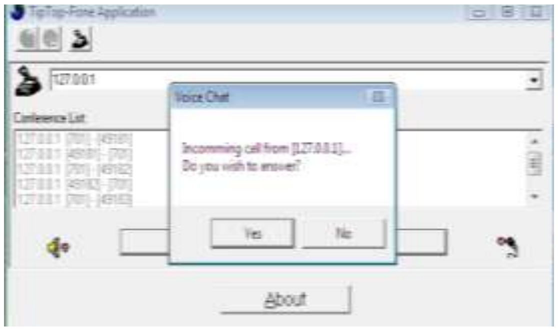


Figure 5: Client form

Requesting a connection through the TCPSocket by the LocalHost

For the client to make a connection request to the server, the following module was designed for that purpose.

```
Private Sub TCPSocket_ConnectionRequest(Index As Integer, ByVal requestID As Long)
    Dim rc As Long
    Dim RemHost As String
    If (TCPSocket(Index).RemoteHost <> "") Then
        RemHost = UCase(TCPSocket(Index).RemoteHost)
    Else
        RemHost = UCase(TCPSocket(Index).RemoteHostIP)
    End If
     If (Tools.Buttons(tbAUTOANSWER).Value = tbrUnpressed) Then
        rc = MsgBox("Incomming call from [" & RemHost & "]..." & vbCrLf & _
                    "Do you wish to answer?", vbYesNo)
    Else
    rc = vbYes
    End If
   End Sub
```



Incomming call from [127.0.0.1]...
Do you wish to answer?

Figure 6: Connection Request

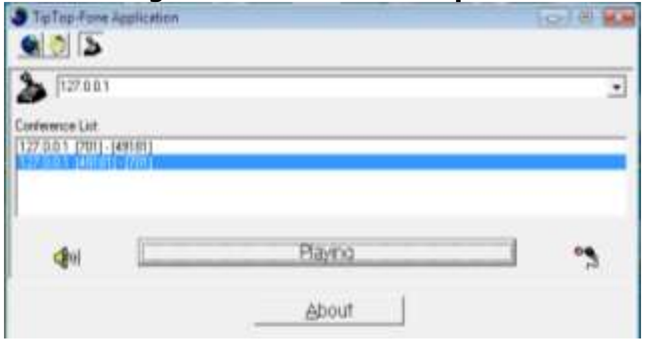


Figure 7: Client Computer Connected

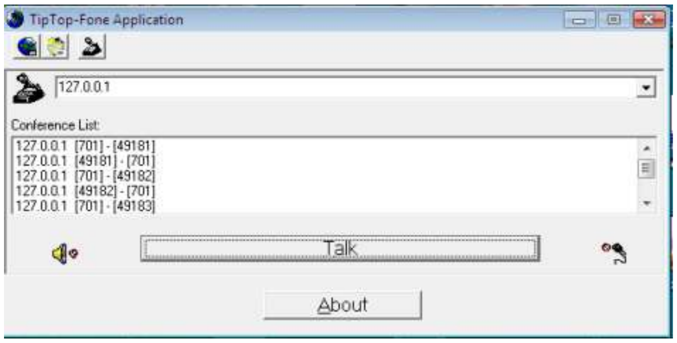


Figure 8: Telephony Activated

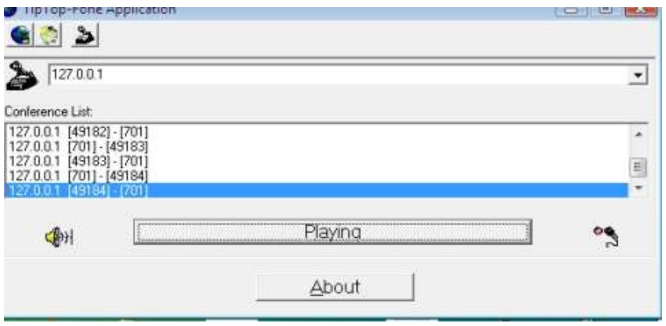


Figure 9: Playback for Telephony Activity Completing

PROGRAM FLOWCHART

The flowchart is a procedural representation of the steps which are taken to write a code to achieve a specific task. The codes for the application were preceded by a detailed flowchart which illustrates the working logic of the system. The program flowchart for both the client end and the server end of the application is as follows:

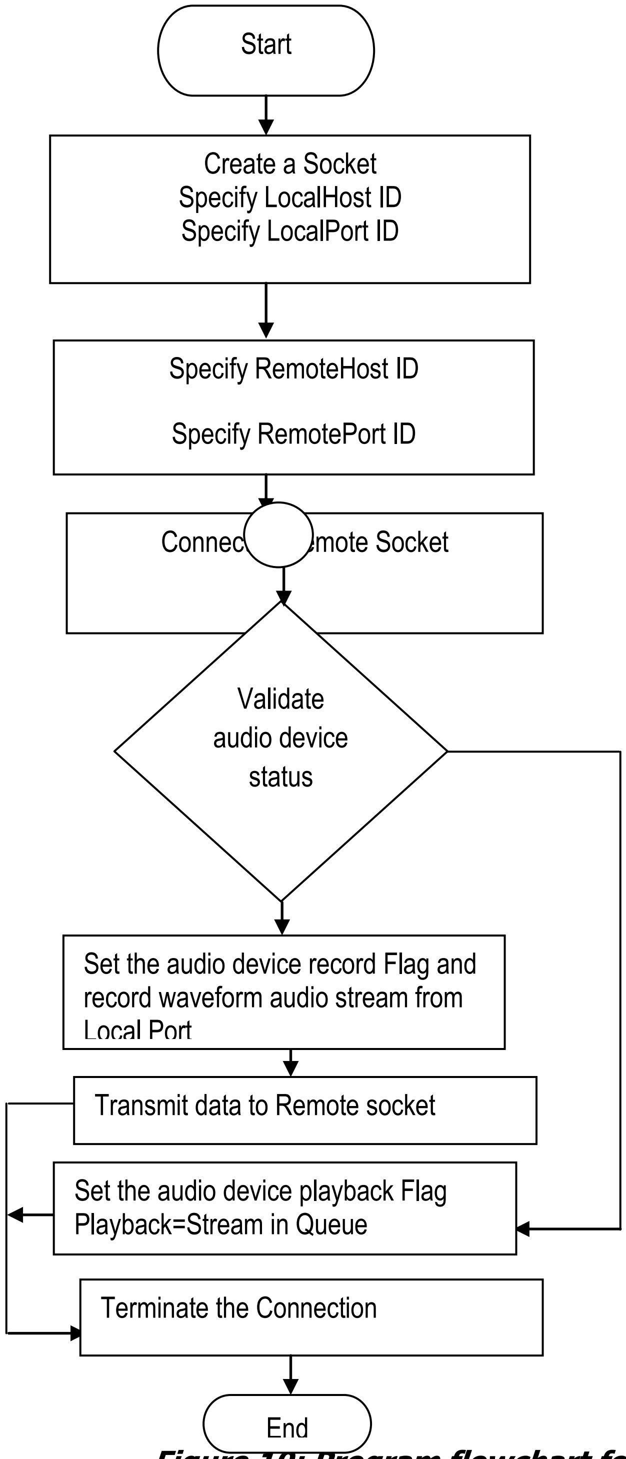

Start
Create a Socket
Specify LocalHost ID
Specify LocalPort ID
Specify RemoteHost ID
Specify RemotePort ID
Connec mote Socket
Validate audio device status
Set the audio device record Flag and record waveform audio stream from Local Port
Transmit data to Remote socket
Set the audio device playback Flag Playback=Stream in Queue
Terminate the Connection
End

Figure 10: Program flowchart for the client end

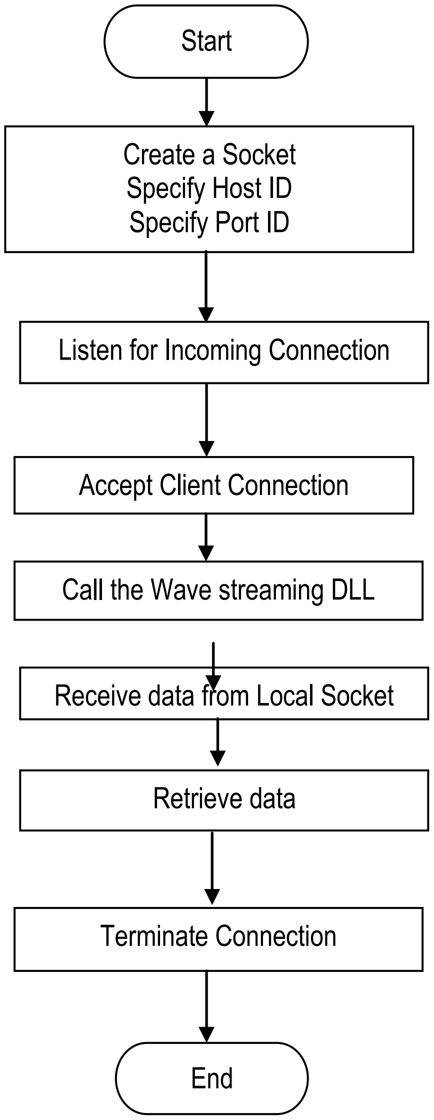


Figure 11: The Program flowchart for the server end.

CONCLUSION

Internet telephony is a technology that is attracting more and more users because it offers tremendous cost savings relative to the PSTN. It could be applied to almost any voice communications requirement; this is made possible by the advances in compression algorithms and computer processing power which actually support real-time voice communications over the internet.

REFERENCES

[1] Bellamy, J.: (2000). Digital Telephony, New York, NY: Wiley.
[2] Comer, D.E.: (2000). Internetworking with TCP/IP, vol. 1, 4th ed., Eaglewood Cliffs, NJ: Prentice Hall,

[3] Chukwudebe, G.A.: (2007). An Evaluation of VoIP and the future of Telecommunications, White Paper, EEE Dept., FUTO Nigeria.
[4] Henning Schulzrinne and J. Rosenberg, (1999). "Internet telephony: Architecture and protocols – an IETF perspective,"*Computer Networks and ISDN Systems*, vol. 31, no. 3, pp. 237–255,
[5] Kundan Singh and Henning Schulzrinne, (2004). "Failover and load sharing in SIP telephony," Tech. Rep. CUCS-011-04, Columbia University, Computer Science Department, New York,
[6] Haakon Bryhni, Espen Klovning, and Øivind Kure, (2000). "A comparison of load balancing techniques for scalable web servers," *IEEE Network*, vol. 14, no. 4,
[7] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, (2001). "A scalable contentaddressable network," in *SIGCOMM Symposium on Communications Architectures and Protocols*, San Diego,bCA, USA, ACM.
[8] Antony Rowstron and Peter Druschel, (2001). "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, pp. 329–350.
[9] Weibin Zhao, Henning Schulzrinne, and Erik Guttman, (2003)."Mesh-enhanced service location protocol (mslp)," RFC 3528, Internet Engineering Task Force.
[10] Miguel Castro, Anne-Marie Kermarrec, Antony Rowstron, and Alec Wolman, (2003). "An evaluation of scalable application-level multicast built using peer-to-peer overlays," in
Proceedings of the Conference on Computer Communications (IEEE Infocom).
[11] Jonathan Lennox, (2004). "Services for internet telephony," PhD. thesis, Department of Computer Science, Columbia University, New York, New York,
http://www.cs.columbia.edu/˜lennox/thesis.pdf.
[12] Anjali Gupta, Barbara Liskov, and Rodrigo Rodrigues, (2004). "One hop lookups for peer-to-peer overlays," in *Hot OS IX: The 9th workshop on hot topics in operating systems*, Lihue, Hawaii, USA, USENIX.
[13] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica, (2002). "Routing algorithms for DHTs: some open questions," in *International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, USA, IEEE.