INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE

Department of Computer Science

(2018-2020)

A REPORT

on

Application Software for Voice Over IP(VOIP) Protocol

Submitted To

Dr .P Sateesh Kumar

Associate Professor

IIT Roorkee

Submitted By

Abhishek Kumar Roy (18535001)

Arvind Kumar Gangwar(18535007)

Amar Sen(18535003)

Deepak Singh(18535010)

Deepak Singh(18535011)

ACKNOWLEDGEMENT

We are highly indebted to Dr. P Sateesh Kumar (Assosiate Professor) for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.
We would like to express our gratitude towards members of INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE for their kind co-operation and encouragement which helped in completion of this project.
Our thanks and appreciations also go to colleague in developing the project and people who have willingly helped us out with their abilities.

Contents

1 Problem Statement

To develop Application Software for Voice Over IP (VOIP) protocol.

2 Introduction

Voice over Internet Protocol (VoIP or IP telephony) is a of voice communications over Internet Protocol (IP) networks, such as the Internet. VoIP uses codecs to encode audio into data packets, transmit the packets across an IP network using sockets and decode the packets back into audio at the other end. Voice over IP (VoIP) technology has many advantages over the traditional Public Switched Telephone Network (PSTN). First, IP networks are more cost-efficient as it reduces operating cost from the standard equipment and utilizes the existing network of internet. So service subscribers are charged with lower price for voice services than PSTNs.
Second, IP networks can provide integrative data and voice services. VoIP can handle data packets as well as voice packets. So it can provide new interesting services. Third, IP networks are more bandwidth-efficient. Bandwidth of the communication channel is very limited. In some sense, it is a priceless source.
IP networks can use more advanced voice coding schemes and take up less bandwidth. Besides G.711, IP networks can also implement G.726, G.728, G.729 and G.723 coding schemes, which only take up 32kbps, 16kbps, 8kbps, and 6.3kbps bandwidth respectively whereas PSTNs mainly use International Telecommunication Union (ITU) Recommendation G.711 coding scheme which samples the analog voice signals and , encodes one sample with 8 bits and take up 64kbps bandwidth.

3 Discussions on protocol

3.1 Media Gateway Control Protocol(MGCP)

Media Gateway Control Protocol (MGCP) is a protocol used for controlling Voice over IP (VoIP) Gateways from external call control elements. MGCP is a protocol for controlling media gateways from call agents. In a VoIP system, MGCP can be used with SIP or H.323. SIP or H.323 will provide the call control functionality and MGCP can be used to manage media establishment in media gateways.
Characteristics of MGCP:

- A master/slave protocol

  - Assumes limited intelligence at the edge (endpoints) and intelligence at the core (call agent)
  - Used between call agents and media gateways
  - Differs from SIP and H.323 which are peer-to-peer protocols

- Interoperates with SIP and H.323.

3.2 RTP (Real-time Transport Protocol)

RTP supports the transfer of real-time media (audio and video) over packet switched networks. The transport protocol must allow the receiver to detect any losses in packets and also provide timing information so that the receiver can correctly compensate for delay jitter. The RTP header contains information that assist the receiver to reconstruct the media and also contains information specifying how the codec bits treams are broken up into packets. RTP does not reserve resources in the network but instead it provides information so that the receiver can recover in the presence of loss and jitter.
The functions provided by RTP include:

- Sequencing: The sequence number in the RTP packet is used for detecting lost packets

- Payload Identification: In the Internet, it is often required to change the encoding of the media dynamically to adjust to changing bandwidth availability. To provide this functionality, a payload identifier is included in each RTP packet to describe the encoding of the media

- Frame Indication: Video and audio are sent in logical units called frames. To indicate the beginning and end of the frame, a frame marker bit has been provided

- Source Identification: In a multicast session, we have many participants. So an identifier is required to determine the originator of the frame. For this Synchronization Source (SSRC) identifier has been provided.

- Intramedia Synchronization: To compensate for the different delay jitter for packets within the same stream, RTP provides timestamps which are needed by the play-out buffers.

3.3 PTCP(Real-time Control Protocol)

RTCP is a control protocol and works in conjunction with RTP. In a RTP session, participants periodically send RTCP packets to obtain useful information about QoS etc. The additional services that RTCP provides to the participants are:

- QoS(Quality of Service) feedback: RTCP is used to report the quality of service. The information provided includes number of lost packets, Round Trip Time, jitter and this information is used by the sources to adjust their data rate
- Session Control: By the use of the BYE packet, RTCP allows participants to indicate that they are leaving a session
- Identification: Information such as email address, name and phone number are included in the RTCP packets so that all the users can know the identities of the other users for that session
- Intermedia Synchronization: Even though video and audio are normally sent over different streams, we need to synchronize them at the receiver so that they play together. RTCP provides the information that is required for synchronizing the streams

3.4 Real-Time Streaming Protocol (RTSP)

RTSP, the Real Time Streaming Protocol, is a client-server protocol that provides control over the delivery of real-time media streams. It provides "VCR-style" remote control functionality for audio and video streams, like pause, fast forward, reverse, and absolute positioning. It provides the means for choosing delivery channels (such as UDP, multicast UDP and TCP), and delivery mechanisms based upon RTP. RTSP establishes and controls streams of continuous audio and video media between the media servers and the clients. A media server provides playback or recording services for the media streams while a client requests continuous media data from the media server. RTSP acts as the "network remote control" between the server and the client. It supports the following operations:

- Retrieval of media from media server: The client can request a presentation description, and ask the server to setup a session to send the requested data. The server can either multicast the presentation or send it to the client using unicast.
- Invitation of a media server to a conference: The media server can be invited to the conference to play back media or to record a presentation

- Addition of media to an existing presentation: The server or the client can notify each other about any additional media that has become available

3.5 Resource Reservation Protocol (RSVP)

The network delay and Quality of Service are the most hindering factors in the voice-data convergence. The most promising solution to this problem has been developed by IETF viz., RSVP. RSVP can prioritize and guarantee latency to specific IP traffic streams. RSVP enables a packet-switched network to emulate a more deterministic circuit switched voice network. With the advent of RSVP, VOIP has become a reality today. With RSVP enabled, we can accomplish voice communication with tolerable delay on a data network. RSVP requests will generally result in resources being reserved in each node along the data path. RSVP requests resources in only one direction, therefore it treats a sender as logically distinct from a receiver, although the same application process may act as both a sender and a receiver at the same time. RSVP is not itself a routing protocol, it is designed to operate with current and future unicast and multicast routing protocols. In order to efficiently accommodate large groups, dynamic group membership, and heterogeneous receiver requirements, RSVP makes receivers responsible for requesting a specific QoS.

3.6 Session Description Protocol (SDP)

SDP is intended for describing multimedia sessions for the purpose of session announcement, session invitation etc. The purpose of SDP is to convey information about media streams in multimedia sessions to allow the recipients of a session description to participate in the session. SDP includes the following information:

- Session name and purpose
- Address and port number
- Start and stop times
- Information to receive those media
- Information about the bandwidth to be used by the conference
- Contact information for the person responsible for the session

3.7 Session Announcement Protocol (SAP)

This protocol is used for advertising the multicast conferences and other multicast sessions. A SAP announcer periodically multicasts an announcement packet to a well known multicast address and port. SAP requires announcements to occur periodically, generally with an IP time-to-live (TTL) of 255. Announcements must be sent on Port 9875. To announce a multicast session, the session creator sends a multicast packet to a well-known multicast group using the Session Description Protocol (SDP) to describe the packets.

4 Current research areas in VOIP

VoIP protocols and products have been repeatedly found to contain numerous vulnerabilities that have been exploited. As a result, a fair amount of research has been directed towards addressing some of these security issues. More efficient codec s is being implemented. QoS of service is the measure issues in the VOIP.

5 Implementation details

Here a simple application has been developed which essentially demonstrates the working feature of VOIP excluding the signalling protocol and QoS protocols such as SIP,RVSP etc..In a simple VoIP system (as shown in Figure 1), there are three main modules, speech collection/playing module, CODEC (coder/decoder) module and socket communications module. At the transmitter side, the speech signal is collected and encoded before transmitted to IP networks. At the receiver side, we do the reverse processes. The received data stream is decoded and recovered into speech signal and played back.

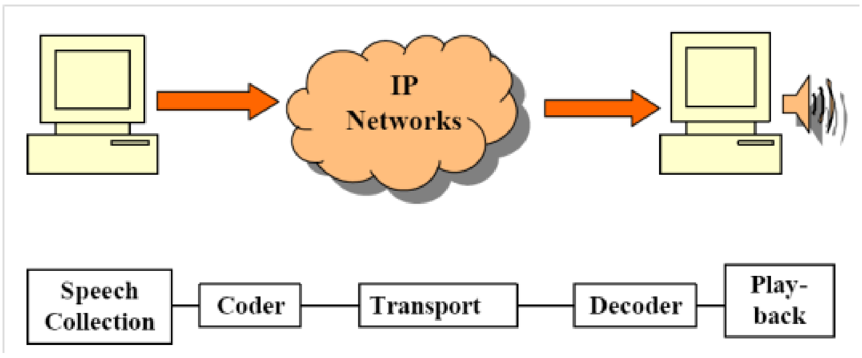


Fig1.Block diagram of VOIP application

CODEC

A codec, which stands for coder-decoder, converts an audio signal (your voice) into compressed digital form for transmission (VoIP) and then back into an uncompressed audio signal for replay. It's the essence of VoIP. Codecs vary in the sound quality, the bandwidth required, the computational requirements, etc. To be able to transmit voice signal, we need to convert it from analog format to binary data first. This step is called voice coding. Voice coder is used to encode speech samples into a small number of bits so that the speech is robust in the presence of link errors, jittery networks, and burst transmission. At the receiver, the bits are decoded back to the PCM speech samples and then converted to analog waveform. Coders are mainly classified into three types, waveform coders, vocoders and hybrid coders.

Waveform coders reproduce the analog waveform as accurately as possible, including background noise. They operate at high bit rate. G.711 is the waveform coder to represent 8 bit compressed pulse code modulation (PCM) samples with the sampling rate of 8000Hz.

Vocoders do not reproduce the original waveform. Linear Prediction Coding (LPC), for example, is used to derive parameters of a time-varying digital filter. The quality of vocoder is not good enough for use in telephony system.

VoIP use the hybrid coder, which uses the attractive features of waveform coder and vocoder. It is also attractive because it operates at a low bit rate as low as 4-16 kbps. G.728 is a hybrid between the lower bit rate linear predictive analysis-by-synthesis coder (G.729 and G.723.1) and the backward ADPCM coders. G.728 is a LD-CELP coder and operates on five samples at a time.

SOCKET COMMUNICATIONS

Socket is used to communicate between two communication endpoint. From the connection view, there are 2 types of socket communications, connection oriented and connectionless socket communications. For the first type, a connection must be set up before transmission. It is more reliable than connectionless socket communication. TCP is the transport protocol for the connection-oriented socket communication. So, we choose TCP as our transport protocol to communicate between two end points. We have implemented window socket for this. Visual C++ , library has been used for sampling the voice.

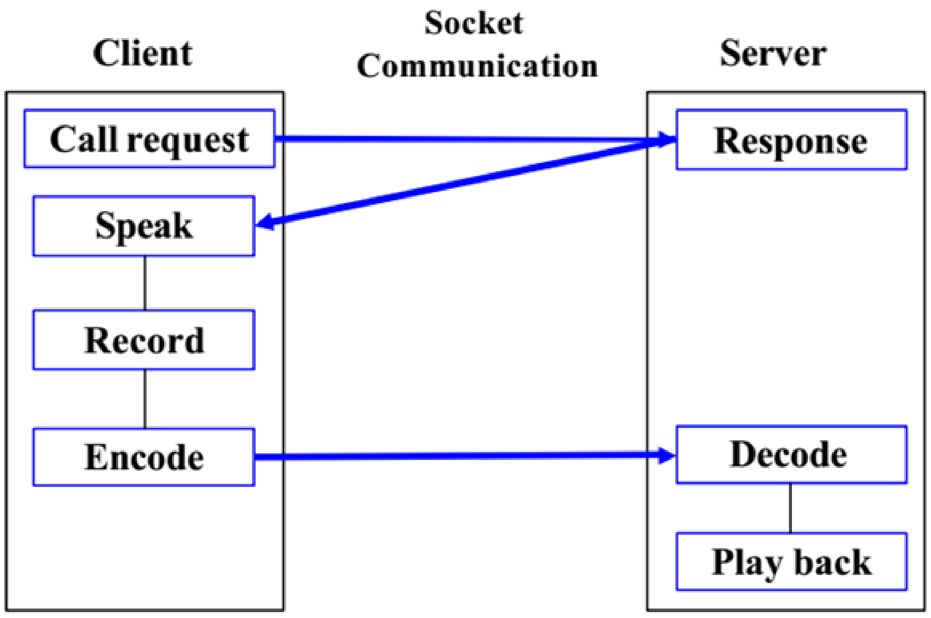


Fig2. Application Flow chart

Figure 2 is a simplex mode of transmission, the client makes a Call request, the server responses this request. When, the client begins to speak, the voice gets recorded and encoded, then get transmitted to the decoder on the server side. Then it get decoded and played back in the server side. To realize our duplex function, we will use the coder and decoder in the both ends of the communication links. Of course, we will need 2 communications sockets, one for each direction.

6 Pros and Cons of using VoIP

8.1. Advantages of VoIP:

VoIP technology can offer many advantages over the traditional Public Switched Telephone Networks to the residential and small office/home users. If you have a high speed internet connection then choosing a VoIP phone service might be very useful.

Low Cost :

This technology may be vary cheaper because there exists only one network carrying the voice and data provided by only one supplier. If you have a broadband Internet connection (DSL or cable), you can make PC-to-PC phone calls anywhere in the world for free. If you wish to make a PC-to-phone connection, there's usually a charge for this but probably much cheaper than your regular phone service.

Low Taxes :

Since the calls are being carried over the Internet, governments have not heavily taxed VoIP phone services. Compare that to your local telephone bill (go ahead and take a close look) and you will see you are spending quite a bit on taxes each month. Therefore, choosing a VoIP provider could add up to significant savings for you and your family.

Portability :

One important concept to understand about VoIP is that unlike it's forefathers (let's call them PSTN for now), it is not distance or location dependent. As far as VoIP is concerned, you could be calling your supplier 1,000 miles away in Indonesia or calling your business partner on the other end of town, and it doesn't make any difference at all, in terms of connectivity and cost.

You can make and receive phone calls wherever there is a broadband connection simply by signing in to your VoIP account. This makes VoIP as convenient as e-mail – if you are traveling, simply pack a headset or Internet phone and you can talk to your family or business associates for almost nothing.

No extra cables, no extra cost :

A VoIP phone number, unlike your regular phone number, is completely portable. Most commonly referred to as a virtual number, you can take it with you anywhere you go. Even if you change your office address to another state, you phone number can go with you. Heck, you can even take your whole business with you wherever you travel.

Additional Features :

Unlike regular phone service which usually charges more for extra features, VOIP comes with a host of advanced communication features. For example, call forwarding, call waiting, voicemail, caller ID and three-way calling are some of the many services included with VOIP telephone service at no extra charge. You can also send data such as pictures and documents at the same time you are talking on the phone.

VoIP phones can integrate with other services available over the Internet, including video conversation, message or data file exchange in parallel with the conversation, audio conferencing, managing address books and passing information about whether others (e.g. friends or colleagues) are available online to interested parties.

Flexibility :

When you choose a VoIP phone service provider, you will be sent a converter to allow a regular phone to use the VoIP phone service. Your phone number is programmed into the converter. This means that you can take your phone converter and phone number and use them wherever you travel in the world, just as long as you have access to a high-speed Internet connection. Because your telephone number is based in your converter (and not your home/office), you have the option of choosing any area code for your phone number. Some carriers will allow you to have more than 1 phone number in different area codes for a small additional fee (called a virtual phone number).

8.2 Disadvantages of VoIP:

If VOIP is starting to sound really good to you, make sure you understand the following downsides as well.

No service during a power outage :

During a blackout a regular phone is kept in service by the current supplied through the phone line. This is not possible with IP phones, so when the power goes out, there is no VOIP phone service. In order to use VoIP during a power outage, an uninterruptible power supply or a generator must be installed on the premises. It should be noted that many early adopters of VoIP are also users of other phone equipment such as PBX and cordless phone bases that also rely on power not provided by the telephone company.

Emergency calls :

Another major concern with VOIP involves emergency 911 calls. Traditional phone equipment can trace your location. Emergency calls are diverted to the nearest call center where the operator can see your location in case you can't talk. However, because a voice-over-IP call is essentially a transfer of data between two IP addresses, not physical addresses, with VOIP there is currently no way to determine where your VOIP phone call is originating from.

Although many companies are making an effort to provide for emergency calls in their service, this issue remains an important deterrent against VoIP.

Reliability :

Because VOIP relies on an Internet connection, your VOIP service will be affected by the quality and reliability of your broadband Internet service and sometimes by the limitations of your PC. Poor Internet connections and congestion can result in garbled or distorted voice quality. If you are using your computer at the same time as making a computer VOIP call, you may find that voice quality deteriorates dramatically.

This is more noticeable in highly congested networks and/or where there are long distances and/or internetworking between end points.

VoIP Voice Quality :

VoIP has a bit to improve on Voice Quality, but not in all cases. VoIP QoS(Quality of service) depends on so many factors: your broadband connection, your hardware, the service provided by your provider, the destination of your call etc. More and more people are enjoying high quality of phone calls using VoIP, but still many users complain of hearing Martian, having to wait a lot before hearing an answer etc.

Security :

This one is the last in this list, but it is not the least! Security is a main concern with VoIP, as it is with other Internet technologies. The most prominent security issues over VoIP are identity and service theft, viruses and malware, denial of service, spamming, call tampering and phishing attacks.

7 Conclusion

In our project, we have implemented a simple VoIP application software, which is able to transmit the voice from one internet host to another. The project has given us a good understanding of the concept of networking using sockets in general and the concept of VoIP in particular. The implementation of this simple VoIP application has helped us in understanding, how we can establish communication between two internet hosts using sockets and how we can use an internet connection to transfer the audio signal using internet. Voice over IP (VoIP) technology has many advantages over the traditional Public Switched Telephone Networks. However we need to tackle many challenges to deploy a fully functional VoIP system in place. However over a period of time these challenges has been successfully conquered by developing many protocols to handle the issues of quality, reliability and security.

8 Implementation (Source Code

VoIP implementation has been carried in C-C++ language using Microsoft Visual Studio IDE. The software implements functionality for –

- Getting PCM samples of audio recorded through microphone at a rate of 8000 samples per second
- Windows AF_INET datagram socket for transmit and receive of data over IP
- Plays back received data at PC's speaker/headphone at 8000 samples per second

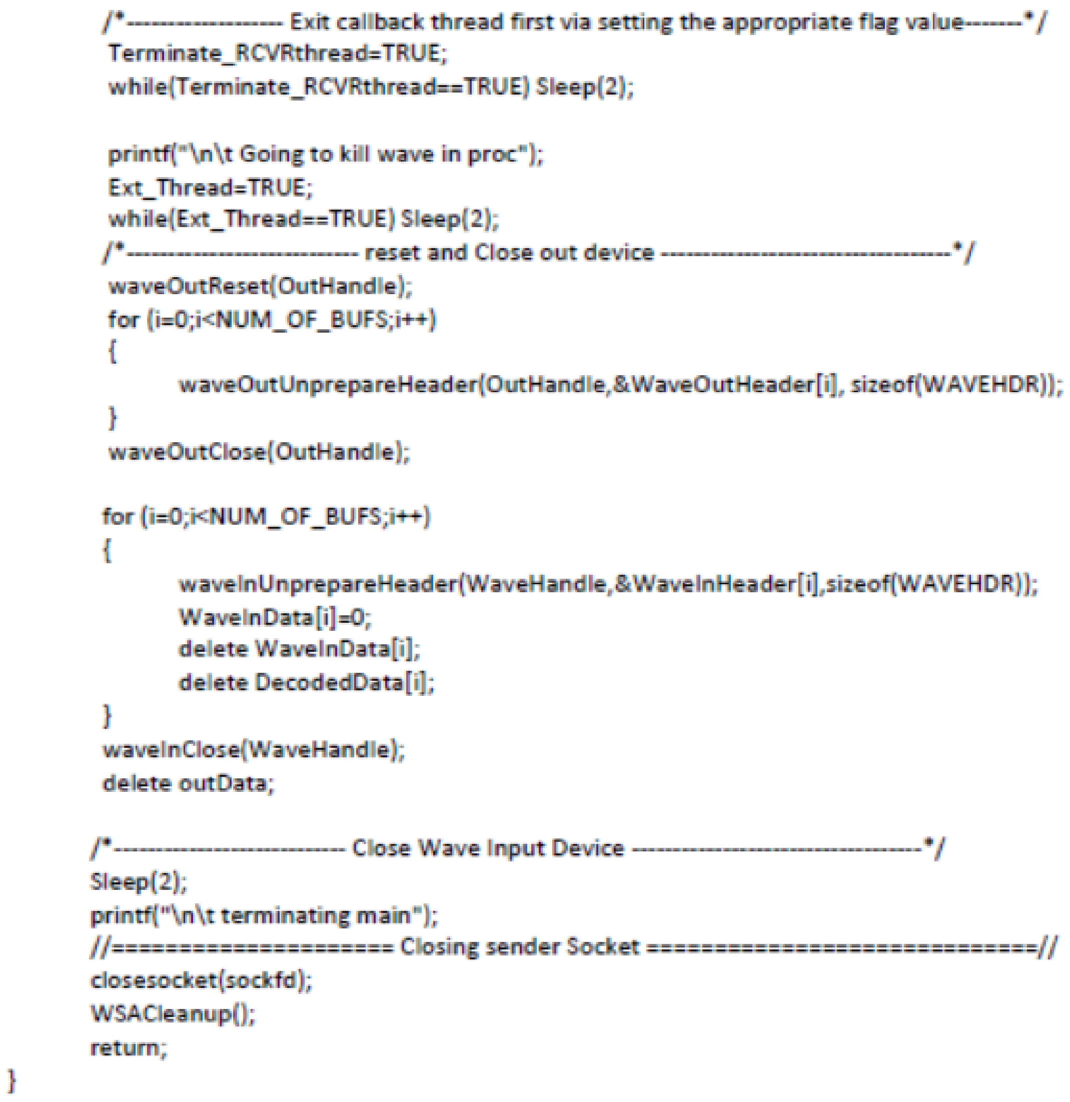
Code-

```
#define NUM_OF_BUFS 20
int sockfd;
struct sockaddr_in their_addr;              // connector's address information
struct sockaddr_in my_addr;                 // my address information
struct hostent *he;
char *dest_IP="192.168.7.10";              //destination IP address

#define MYPORT 4950            // the port users connect to
#define MAXBUFLEN 1024
#define ReadDataSize 1024
#define DATAPACKET 0x01
#define REQPACKET  0x00
#define ENDPACKET  0x02

/*---------------------------- for wave in open global ------------------------------------*/
WAVEFORMATEX WaveFormat;
HWAVEIN WaveHandle;
HWAVEOUT OutHandle;

char *outData
BOOL Ext_Thread=FALSE;
BOOL Buffer_Played=TRUE;
BOOL Terminate_RCVRthread=FALSE;
BOOL Terminate_TxThread=FALSE;

char* WaveInData[NUM_OF_BUFS];
WAVEHDR WaveInHeader[NUM_OF_BUFS];
WAVEHDR WaveOutHeader[NUM_OF_BUFS];
int recordingCounter=0, playingCounter=0;
int BufferSize;
int size;
int playCounter,recordCounter;
char *EncodedData, *DecodedData[NUM_OF_BUFS];
int EncodedLength=0, DecodedLength[NUM_OF_BUFS];

struct packet{
int pktType;
char data[ReadDataSize];
};


void Handle_Buffer()
{
        int Res,k;
        int numbytes;
        struct packet myPkt;
        /*---------------------------- Extract Size of Recorded Data -------------------------------------*/
        size=WaveInHeader[recordingCounter].dwBytesRecorded;
        /*---------------------------- Buffer to Hold Data to Play ------------------------------------*/
        for(k=0;k<size;k++) // 2048
        {
                        outData[k]=WaveInData[recordingCounter][k];
                        myPkt.data[k]=WaveInData[recordingCounter][k];
        }
        /*-------------------------------- Enqueue This buffer for next recording-----------------------------*/
Res = waveInAddBuffer(WaveHandle, &WaveInHeader[recordingCounter], sizeof(WAVEHDR));
if(Res!=MMSYSERR_NOERROR)
{
    LPSTR errtext;
   errtext= new char[150];
   waveInGetErrorText(Res,errtext,150);
   printf("wave in add buffer error:: handle buffer : %d error = %s\n", Res,errtext);
   return;
}
myPkt.pktType=DATAPACKET;
if((numbytes=sendto(sockfd, (char *) &myPkt, sizeof(struct packet), 0,(struct sockaddr
*)&their_addr, sizeof(struct sockaddr))) == -1)
{
        cout<<"sendto error="<<WSAGetLastError()<<endl;
}
//==============================================================//
recordingCounter=recordingCounter+1;
if(recordingCounter==NUM_OF_BUFS)
{
        recordingCounter=0;
}
}

static void CALLBACK waveInProc(LPVOID arg)
{
        HWND hWnd = (HWND)arg;
        MSG    msg;

        while (GetMessage(&msg, (HWND)-1, 0, 0) == 1)
         {
                if(Ext_Thread==TRUE)
                        break;
                switch (msg.message)
                 {
                        case MM_WIM_DATA:
                                {
                                        Handle_Buffer();
                                        break;
                                }
                }
        }
        Ext_Thread=FALSE;
        printf("\n\t waveInProc thread exit");
        ExitThread(0);
}
```

```
void Receiver(void)
{
  int k;
  int numbytesRcvd,addr_len=sizeof(their_addr);
  struct packet rcvdPkt;


        for(; ;)
         {
          if(Terminate_RCVRthread==TRUE)
                break;

          if ((numbytesRcvd=recvfrom(sockfd,(char*)&rcvdPkt, MAXBUFLEN, 0,(struct sockaddr
          *)&their_addr, &addr_len)) == -1)
         {
                 perror("recvfrom error");
         }
        for(k=sizeof(int);k<size;k++) // 2048
        {
          DecodedData[playingCounter][k-sizeof(int)]=WaveInData[recordingCounter][k];
        }

        waveOutWrite(OutHandle, &WaveOutHeader[playingCounter], sizeof(WAVEHDR));
        playingCounter=playingCounter+1;
        if(playingCounter==NUM_OF_BUFS)
        {
                playingCounter=0;
        }
        }
        Terminate_RCVRthread=FALSE;
        printf("\n\t Rx thread exitted");

        ExitThread(0);
        return;
}

void main()
{
        int j=0,i;
        WSADATA wsa;
        if(WSAStartup(MAKEWORD(2,2), &wsa)!=0)
        {
                cout<<"WSA startup Error"<<endl;
                getchar();
                exit(0);
        }
  //===================== Initialising Sender Socket ================================//
        if ((he=gethostbyname(dest_IP)) == NULL)
        {
                cout<<"gethostbyname error"<<endl;
                exit(0);
```

```
    }
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
    {
            cout<<"socket Open Error"<<endl;
            exit(0);
    }
    their_addr.sin_family = AF_INET;
    their_addr.sin_port = htons(MYPORT);
    their_addr.sin_addr = *((struct in_addr *)he->h_addr);
    memset(&(their_addr.sin_zero), '\0', 8);
    //===================== Initialising Receive Socket ============================//
    my_addr.sin_family = AF_INET;
    my_addr.sin_port = htons(MYPORT);
    my_addr.sin_addr.s_addr = INADDR_ANY;
    memset(&(my_addr.sin_zero), '\0', 8);
    if (bind(sockfd, (struct sockaddr *)&my_addr,sizeof(struct sockaddr)) == -1)
    {
            perror("bind");
            getchar();
            exit(1);
    }
    DWORD dwTid,dwTid1,dwTid2;
    HANDLE wavThread,wavThread1,wavThread2;
  wavThread=CreateThread(NULL,0,(LPTHREAD_START_ROUTINE)waveInProc,NULL,0,&dwTid);
  CloseHandle(wavThread);
  wavThread1=CreateThread(NULL,0,(LPTHREAD_START_ROUTINE)Receiver,NULL,0,&dwTid1);
  CloseHandle(wavThread1);

    /*-------------------------- Initialize format parameters ---------------------------------*/
    WaveFormat.wFormatTag         = WAVE_FORMAT_PCM;
    WaveFormat.nChannels          = 1;
    WaveFormat.nSamplesPerSec     = 8000;
    WaveFormat.wBitsPerSample     = 8;
    WaveFormat.nAvgBytesPerSec    = 8000; // nsamplespersec *nblockalign
    WaveFormat.nBlockAlign        = 1; // nchannels * wbitspersample/8;
    WaveFormat.cbSize             = 0;

    /*-------------------------- Alocate memory to buffers ---------------------------------*/
    BufferSize =(int)(0.016 * 8000);  // recording till size for 16 millisecond
    for(i=0;i<NUM_OF_BUFS;i++)
    {
            WaveInData[i] = new char[BufferSize];
            DecodedData[i] = new char[BufferSize];
            DecodedLength[i]=0;
    }
    outData= new char[BufferSize];

    /*-------------------------- Five Wave Headers to hold each Buffers ----------------------------------*/
    for(i=0;i<NUM_OF_BUFS;i++)
    {
            WaveInHeader[i].dwBufferLength          =         BufferSize;
```

```
        WaveInHeader[i].dwFlags                 =       0;
        WaveInHeader[i].lpData                  =       WaveInData[i];

        WaveOutHeader[i].dwBufferLength         =       BufferSize;
        WaveOutHeader[i].dwFlags                =       0 ;
        WaveOutHeader[i].lpData                 =       DecodedData[i];
}

/*--------------- Query and Then Open Wave Audio Device using waveinopen---------------------*/
int Res = waveInOpen(&WaveHandle,WAVE_MAPPER, &WaveFormat,(DWORD)dwTid, 0,
WAVE_FORMAT_QUERY);
if (Res == WAVERR_BADFORMAT){printf("\n\t Wave in Query error"); return;}
Res = waveInOpen(&WaveHandle, WAVE_MAPPER, &WaveFormat,(DWORD)dwTid, NULL,
CALLBACK_THREAD);
if (Res !=MMSYSERR_NOERROR ) {printf("\n\t Wave in open error"); return;}

/*---------------------------- Query and Open OutPut Device -----------------------------------*/
Res = waveOutOpen(&OutHandle,WAVE_MAPPER, &WaveFormat, 0, 0,
WAVE_FORMAT_QUERY);   //(DWORD)dwTid2
if (Res == WAVERR_BADFORMAT){printf("\n\t Wave out Query error\n"); return;}
Res = waveOutOpen(&OutHandle, WAVE_MAPPER, &WaveFormat,0, 0, CALLBACK_NULL);
if (Res !=MMSYSERR_NOERROR ) {printf("\n\t Wave out open error\n"); return;}
/*---------- Prepare Headers and Then Add all five buffers to Audio device-----------------*/
for(i=0;i<NUM_OF_BUFS;i++)
{
        Res = waveInPrepareHeader(WaveHandle, &WaveInHeader[i], sizeof(WAVEHDR));
        if (Res !=MMSYSERR_NOERROR ) {printf("\n\t Wave in prepare header error");
        return;
}

Res = waveInAddBuffer(WaveHandle, &WaveInHeader[i], sizeof(WAVEHDR));
if (Res !=MMSYSERR_NOERROR ) {printf("\n\t Wave in Add header error"); return;}
Res = waveOutPrepareHeader(OutHandle, &WaveOutHeader[i], sizeof(WAVEHDR));
if (Res !=MMSYSERR_NOERROR ) {printf("\n\t Wave out prepare header error\n"); return;}
}
recordCounter=0;
printf ("Recording now\n");
/*---------------------------- Start Recording -----------------------------------*/
Res = waveInStart(WaveHandle);
if (Res !=MMSYSERR_NOERROR ) {printf("\n\t Wave in start error"); return;}
printf("\nPress Any Key To Stop\n");
/*---------------------------- Record Until a key is pressed by user -----------------------------------*/
while(!kbhit())
{
        Sleep(0.1);
}

/*---------------------------- Free Buffered Memory -----------------------------------*/
 waveInStop(WaveHandle);
 waveInReset(WaveHandle);
```



```
        /*------------------ Exit callback thread first via setting the appropriate flag value-------*/
         Terminate_RCVRthread=TRUE;
         while(Terminate_RCVRthread==TRUE) Sleep(2);

         printf("\n\t Going to kill wave in proc");
         Ext_Thread=TRUE;
         while(Ext_Thread==TRUE) Sleep(2);
        /*--------------------------- reset and Close out device -----------------------------------*/
         waveOutReset(OutHandle);
         for (i=0;i<NUM_OF_BUFS;i++)
         {
                waveOutUnprepareHeader(OutHandle,&WaveOutHeader[i], sizeof(WAVEHDR));
         }
         waveOutClose(OutHandle);

        for (i=0;i<NUM_OF_BUFS;i++)
        {
                waveInUnprepareHeader(WaveHandle,&WaveInHeader[i],sizeof(WAVEHDR));
                WaveInData[i]=0;
                delete WaveInData[i];
                delete DecodedData[i];
        }
        waveInClose(WaveHandle);
        delete outData;

       /*--------------------------- Close Wave Input Device ----------------------------------*/
       Sleep(2);
       printf("\n\t terminating main");
       //===================== Closing sender Socket ==============================//
       closesocket(sockfd);
       WSACleanup();
       return;
}
```

9 Referencess

1. Walter J. Goralski and Matthew C. Kolon, IP telephony. ISBN 007135221X, McGraw-Hill, 1st edition, 2000.

2. Simple Voice over IP (VoIP) Implementation Hong Xiong, Yuan Guo, Fang Zhu ECE Department, University of Florida

3. http://programmerworld.net