中国计量大学
现代科技学院
本科毕业设计（论文）

基于 TCP/IP 协议的网络语音通信系统的设计

Implementation of Network voice communication system based on TCP/IP protocol

学生姓名 鲍克力 学号 1330334215

学生专业 通信工程 班级 通信 132

系 信息工程系 指导教师 肖英副教授

中国计量学院现代科技学院

2017 年 05 月

郑 重 声 明

本人呈交的毕业设计论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

学生签名：_____ 日期：_____

分类号：TP393.1　　密　　级：公开
UDC　：654　　学校代码：13292

中国计量大学
现代科技学院
本科毕业设计（论文）

基于 TCP/IP 协议的网络语音通信系统的设计

Implementation of Network voice communication system based on TCP/IP protocol

作　　者　鲍克力　　学　　号　1330334215
申请学位　工学学士　　指导教师　肖英 副教授
学科专业　通信工程　　培养单位　中国计量学院现代科技学院
答辩委员会主席　陈建国　评 阅 人　洪波

2017 年 05 月

致谢

时光荏苒，一转眼紧张而又充实的四年大学生活即将画上一个句号。在这四年里面，感谢各位老师对我的淳淳教诲，是你们教会了我们勤奋刻苦，诚实做人，踏实做事，以宽容之心面对生活，指引着我们沿着正确方向前进。

首先，我要感谢我的指导老师肖英老师对我的教导。无论是为人还是治学，她都是我学习的榜样，也是我能值得信赖的良师益友。她主动关心我的学习和科研，在她还有着繁重的教学工作任务的情况下。从论文的选题、构思、撰写到最终的定稿，肖老师都给了我热情的帮助和悉心的指导，使我的毕业论文能够顺顺利利的完成。

最后，我还要感谢我的同学和朋友在我撰写论文时对我提供的帮助，对我撰写论文启发很大。也要感谢参考文献的作者和提供者们，因为有他们这些前辈打下的基石，才使我现在研究的课题有个很好的出发点。

基于 TCP/IP 协议的网络语音通信系统的设计

摘要：本论文首先概述了基于 TCP/IP 协议的网络语音通信系统的研究背景和发展情况，介绍了几种比较典型的网络语音通信方式。相对着重的介绍了一下 TCP/IP 协议。在讨论了相应的网络技术后，对因特网上的语音传输以及语音信号的输入输出的关键问题也进行了探讨。

在之前的研究基础上，本文主要探讨基于 TCP/IP 协议的 Socket 网络编程接口，使用 C++和 Win32API 作为开发工具，开发出基于 TCP/IP 网络的语音通信系统。该语音通信系统采用 C/S 模型即客户端和服务端（Client/Server）组成，实现全双工语音通信。实验结果表面，在网络通畅的情况下，该语音通信系统语音交互清晰，杂音少，语音质量比较满意。经过多次的实验和比对发现该系统具有较大的实用性。

关键字：TCP/IP 协议；语音通信；Win32；
中图分类号：TP393.1

Implementation of Network Voice communication system based on TCP/IP protocol

Abstract: This paper first summarizes the research background and development of network voice communication system based on TCP / IP protocol, and introduces some typical network voice communication methods. Relative emphasis on the TCP / IP protocol. After discussing the corresponding network technology, the key issues of voice transmission and voice signal input and output on the Internet are also discussed.

On the basis of the above research, this paper based on TCP / IP protocol Socket network programming interface, using C + + and Win32API as a development tool to develop a voice over TCP / IP network communication system. The voice communication system uses C / S model that is the client and server (Client / Server) composition, to achieve full-duplex voice communications. Experimental results on the surface, in the case of smooth network, the voice communication system voice interaction clear, less noise, voice quality is more satisfactory. After several tests, it is found that the system has great practicability.

Keywords: DGS; microstrip filter; LC equivalent circuit; Win32;
Classification: TP393.1

目 次

1. 绪论

1.1 引言

在如今信息高速发展的时代里，信息的交流已经成为人们日常工作生活必不可少的活动。在过去电话是人们语音通信最常用的方式，但长时间的沟通就会带来高额的费用。网络通信已经发展了多年，如今的宽带接入越来越普及，速度也越来越快，支持网络语音已经不成问题。

互联网在近几年发展过程中，正在从一些大众门户转变为个人的客户端。在竞争激烈的互联网客户端中，最受欢迎的那就是即时通讯客户端。人们对即时通讯软件的强烈的需求加上商业刺激，促使互联网行业力推各式各样的即时通讯软件。再加上如今社会发展各行各业都离不开信息交流，即时语音通信系统已经成为互联网行业研究和发展的一个热门领域。

1.2 网络语音通信的研究背景

早在 1995 年 2 月，人们就通过互联网网络利用分组交换来传输语音信号。人们使用这种技术先用压缩算法对语音信号进行压缩，然后再按 TCP/IP 标准将这些语音数据进行打包，再通过 IP 网络将数据发送到目的地。接收地收到信号后把收集到的信号先串起来，通过解码解压等处理之后就恢复成原来的语音数据，从而达到了利用互联网实现语音通信的目的。

最初这种 IP 电话只能在两台计算机之间使用，双方都需要联入互联网并且安装相应的软件和多媒体设备才能通话。后来 IP 电话通过网关将互联网和电话网结合在了一起，实现了计算机和普通电话之间的通话。

后来市面上出现了许多的即时通信工具，一般采用 UDP 和 TCP 协议体系来实现。目前国内外做交谈系统的公司很多，产品也玲琅满目。例如国内比较知名的腾讯 QQ 和微信，国外的有 ICQ 以及微软的 MSN[1]。腾讯利用 QQ 就在互联网上积累了庞大的用户群，因为有这个庞大的用户群从而给他带来了巨大的利润。随着网络软件应用的迅速推广，对即时通信工具的新需求也不断增加，各种新技术、新思想不断涌现，而即时通信工具本身也具有自身的一些特殊性和难点。因此对基于 TCP/IP 的即时语音通信的研究和开发有很有实际应用价值和经济效益[2]。

1.3 开发环境简介

我的开发是在 Windows 下进行的，系统使用的是 Win7 操作系统，开发语言为 C++。开发工具使用的是微软的 Visual Studio 2015 并使用 Win32 底层的 API 进行开发。

Win7 操作系统是目前市面上使用人数最多的主流图形化操作系统。该系统已在全球的家庭和办公环境中使用广泛，并且简单易用，效率高。因为普遍率非常高，

Windows 下的程序开发也因此变的非常热门。相应的 Windows 下开发的开源库和开源代码就比较全面，比较多，更方便开发者开发，提高开发的效率，降低开发成本。微软的成功加上它对标准化的承诺，使得有承诺的 Windows 开发人员利用他们掌握的技术在全球范围内得到越来越大的回报，由于计算机已越来越深入到每个人的生活中，因而对他们的技术需求与日俱增，从而使得他们的回报、经济收入和其他各方面相应地取得了满意的结果。

Win32 API 即为 Microsoft 32 位平台的应用程序编程接口。在所有的 Win32 平台（采用 32 位编码写出的操作系统）上运行的程序都可以调用这些接口。使用 Win32 的接口，编写出的应用程序能充分的发挥出 Windows 的 32 位操作系统的潜力。微软的所有 32 位的机器都支持 Win32API，包括宏、函数、接口、结构、以及消息。目前虽然市场上绝大多数使用的 Windows 操作系统,但是其中操作系统版本不一，有 XP，Vista，Win7，Win8，Win10 等等。想要程序兼容 Windows 下的各个操作系统，必须使用不依赖系统版本的 API 开发。为了兼容性提高系统的实用性，该语音系统选择了比较底层的 Win32 API 进行程序开发。

Visual Studio 2015 是微软公司 2015 年推出的一款 IDE 即集成开发环境。该开发软件包括了整个软件生命周期中所需要用到的各种工具，还支持多平台。Visual Studio 2015 中包含了 Win32 API，在开发过程中可直接使用。它的语法错误提示，以及代码自动补全功能，可大大提高开发者的效率，降低 BUG 的出现。

2. 网络通信

2.1 协议体系结构的简述

协议体系结构指的是由硬件和软件共同组成的一个分层结构，用来支持系统之间的数据交换以及分布式的应用，比如电子邮件以及文件传送的程序。在协议体系结构中的每一层，系统之间的通信必须执行至少是一个双方共同的协议。每个协议种族都会为系统之间的数据交换提供一套相应的规则。

TCP/IP 协议族是现阶段使用最广泛的协议体系结构，它包含了以下几层：物理层、网络接入层、网际层、运输层以及应用层。TCP/IP 是一个基于因特网的的概念，同时也是开发一套完整的计算机通信标准的框架。关于 TCP/IP 协议的具体内容，会在后文详细阐述。还有另一个重要的协议体系结构是开放系统互连（OSI）的七层模型。OSI 作为一个标准化的体系结构经常被用来描述通信功能，但是现阶段他已很少被真正是实现。

2.2 主要的通信协议

多年前，因特网一般就是用来传输文件（FTP）、电子邮件（Email）、网络新闻等，网民人数不多，数据的通信基本建立在 IP 上的两个传输层协议：传输控制协议（TCP）和用户数据报协议（UDP）。伴随着网民的数量不停的增长以及多媒体程序的推陈出新，现在的因特网协议系统已经逐步转向多媒体应用的需求。众多的多媒体需求，促成了 IP 版本 6（IPv6）、资源预留协议（RSVP）和实时传输协议（RTP）等协议的降生。如表 2.1 所示因特网协议体系相对应的多媒体应用。

表 2.1. 因特网协议体系相对应多媒体应用

<table>
<tr><td>标准的 Internet 应用（SNMP,NFS,FTP）</td><td>可靠的组应用（SRM）</td><td>实时（音/视频）应用（RTP）</td><td>WWW 应用（HTTP,RSTP）</td></tr>
<tr><td>UDP</td><td colspan="2">RSVP</td><td rowspan="2">TCP</td></tr>
<tr><td colspan="3">IP v4/v6</td></tr>
<tr><td colspan="4">底层（IEEE802,XLAN,ATM）</td></tr>
</table>

2.2.1 TCP/IP 协议

TCP/IP 协议体系结构是在是在实验性的分组交换网 ARPANET 上展开的协议研究开发工作的成果，ARPANET 由美国国防部研究计划署（Defense Advanced Research Project Agency,DARPA）资助。TCP/IP 协议体系结构一般被称为 TCP/IP 协议族。这个协议族里面集成了大量的协议，这些协议已经通过了因特网体系结构委员会（Internet Architecture Board,IAB）作为因特网标准发布。现阶段比较流行的是

四层 TCP/IP 模型，这个模型主要包括了应用层、传输层、网络层和数据链路层，每一层的功能都不一样。

（1)应用层,一般为用户提供对 TCP/IP 环境的接入,同时也提供分布式的服务。对于各种不同类型的应用程序，如传输文件程序，需要一个独立的专门负责该应用程序的模块。

（2）传输层，主要功能是在两台终端的应用程序之间传送数据。可能会提供差错控制、流量控制、拥塞控制、可靠的交付。

（3）网络层，也被叫做网际层，为高层屏蔽物理网络的配置细节。可以提供路由选择。可能提供 QoS 和拥塞控制。在 TCP/IP 协议族中，网络层协议包括 IP 协议、ICMP 协议（Internet Control Message Protpcol，因特网控制报文协议）以及 IGMP 协议（Internet Group Management Protpcol, 因特网组管理协议)。

（4）数据链路层，一般是到实际网络硬件的逻辑接口。他们一起处理电缆或者其他传输媒介的物理接口细节。可能是面向流的，也可能是面向分组的。

TCP 的全称为传输协议控制(Transmission Control Protocol)。因为他可以提供可靠的，面向连接的网络数据传递服务。传输控制协议主要包含下列几项功能：

- 对程序发送的大数据进行分段和重组。
- 确保正常的排序以及按顺序传递分段的数据。
- 通过计算校验和，对传输后的数据完整性进行检查。
- 根据数据是否成功接收发送回应的信息。通过使用选择性确认，也对没有收到的数据发送进行否定确认。

为必须使用可靠的、基于会话的数据传输程序，如客户端和服务端数据库和网络邮件程序，提供了首选的传输方式。

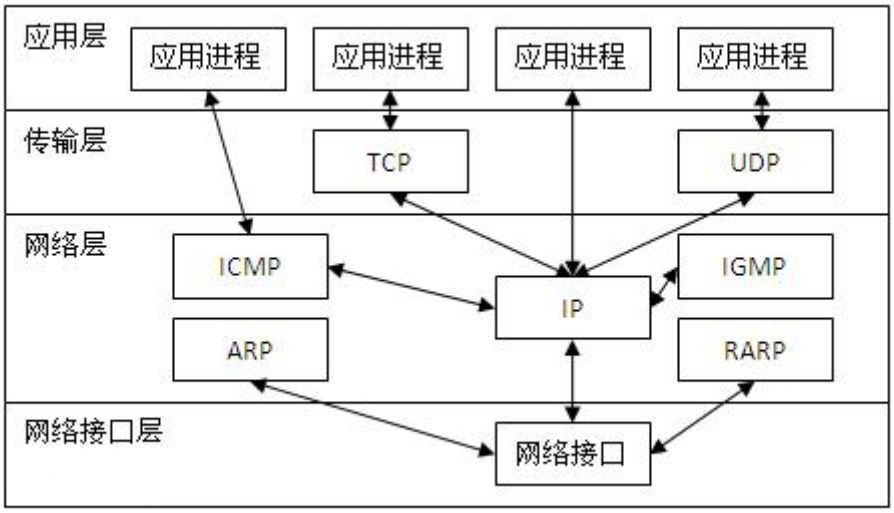

图 2.1. TCP/IP 协议模块关系

2.2.2 UDP 协议

除了 TCP 之外，TCP/IP 协议族中还有一个常用的运输层协：。用户数据报协议（User Datagram Protocol）简称 UDP。UDP 不保证交付的正确性，不维护到达时的顺序，也不管是否重复到达。某些面向事务的应用程序需要使用 UDP，其中的一个例子是 SNMP（简单网络管理协议），他是 TCP/IP 网络的标准网络管理协议。由于 UDP 是无连接的，所以它基本上不用做什么事情。尽管 TCP 中加入了多种安全保障功能，然而真正运行时中会使用大量的系统资源，因此使运行速率受到了巨大的影响。相反 UDP 因为摒弃了信息可靠传递机制，把安全和排序等功能交给上层的应用来管理，大大缩短了运行时间，保障了速率。

2.3 Winsock 的介绍

Winsock 即 Windows Sockets，是 Windows 下得到广泛应用的、开放的、支持多种协议的网络编程接口。如果打算在微软的 Windows 下使用 TCP 和 UDP 协议搭建网络应用程序，则需要使用 Winsock 套接口编程技术。

套接口（又叫套接字）其实本质就是个句柄，这个句柄指向一个传输的提供者。在 Win32 中，套接字和文件描述符不同，它拥有一个独立的参数类型 SOCKET。使用 Socket 的目的是方便用户在网络协议上不需要特别深入了解网络协议。而且使用 Socket 编写的程序还可以移植到其他任何支持 Socket 的网络系统中。建立网络通信连接至少需要一对端口号（Socket）。Socket 的本质是编程的接口（API），他是对 TCP/IP 的封装，TCP/IP 也要给程序员提供做网络开发的接口，这就是 Socket 编程接口。

Socket 的英文意思是“插座”或者“孔”。通常也被人们猜称作“套接字”，用于描述 IP 端口，是一种通信链的句柄，可以用来实现不同虚拟机或不同计算机之间的通信。在因特网上的主机一般运行了多个服务软件，同时提供几种服务。这每种服务都会打开一个 Socket，并绑定到一个端口上，不同的端口对应不同的服务。Socket 正如他的英文愿意一样，就像一个多孔的插座，可以插许多的插头，那些软件就相当于这些插头。

套接口有三种类型：流式套接字、数据报套接字和原始套接字。

流式套接字是一种可靠的面向连接的服务，实际上是基于 TCP 协议实现的。提供了面向连接、可靠的数据传输服务，数据无差错、无重复的发送，并且会按发送顺序接受。

数据报式套接字，实际上是基于 UDP 协议实现的，提供无连接服务。数据包以独立包形式发送，不采用 ACK 确认机制保证数据的到达，数据可能丢失或重复，并且接收顺序混乱。所以，不能保证数据的可靠性。

原始套接字，他允许对较低层次的协议的直接访问，比如 IP、ICMP 协议，他除

了可以处理普通的网络报文外，还可以处理一些特殊的报文以及操作 IP 层机以上的数据。虽然功能强大，但是操作和前两种套接字比较来说不太方便，开发中不考虑使用。

2.4 本章小结

本章主要叙述了现代通信网络以及网络编程的基本知识。介绍了目前比较流行的 TCP/IP 协议族，以及分别简述了 TCP/IP 中两种比较流行的传输协议 TCP 和 UDP。同时也介绍了将 TCP/IP 协议族封装过的网络编程接口 Socket 和在 Windows 下的 Winsock。下一章我们将对音频信号处理展开深入的探讨。

3. 语音信号的处理

在Windows中,API支援的多媒体功能主要分为两个集合。它们通常称为“低阶”和“高阶”介面。低阶的波形声音输入输出函数的开头一般是 waveIn 和 waveOut。我们将在本章详细介绍这些相关的函式。

在 Windows 上，常用的音频处理技术主要有：wave 系列的 API、DirectSound、Core Audio。其中 Core Audio 只能在 Vista 及以上的系统中使用，主要来取代 wave 系列的 API 和 DirectSound。其功能非常强大,能实现麦克风和声卡输出的声音采集，以及对控制声音的播放。Wave 系列的 API 主要实现对麦克风的输入采集，以及对控制声音的播放。为了实现采集模块对操作系统的兼容性更好，基本上对麦克风的输入采集使用 wave 系列的 API 比较多。

波形声音是最常见的 Windows 多媒体特性。波形声音设备能根据麦克风收集到的声音，并把他转为量化数据，接着再把它们保存到储存器或者硬盘上的波形文件中，文件的后缀名是.WAV。这样，声音就可以播放了。

3.1 WaveAPI 的介绍

WaveAPI 是 Windows 提供的波形声音输入输出的 API。首字母带 waveIn 的函数和声音输入有关，带 waveOut 的函数和声音输出有关。要使用 wave 系列的 API 需要在程序中添加 winmm.lib，是否需要加入 mmsystem.h 看情况而定。在讨论声音的采集和播放之前先，须要首先定义音频的相关信息：使用 WAVEFORMATEX 结构体，配置需要的音频信息。以下是 MSDN 中的定义：

```
typedef struct tWAVEFORMATEX
{
WORD wFormatTag; /* format type */
WORD nChannels; /* number of channels (i.e. mono, stereo...) */
DWORD nSamplesPerSec; /* sample rate */
DWORD nAvgBytesPerSec; /* for buffer estimation */
WORD nBlockAlign; /* block size of data */
WORD wBitsPerSample; /* number of bits per sample of mono data */
WORD cbSize; /* the count in bytes of the size of */
/* extra information (after cbSize) */
} WAVEFORMATEX, *PWAVEFORMATEX, NEAR *NPWAVEFORMATEX, FAR *LPWAVEFORMATEX;
```

wFormatTag：设置波形声音的格式，通常情况下是配置成 WAVE_FORMAT_PCM。

nChannels：设置音频文件的通道数量，对于单声道的声音，设置为 1。对于立体

声的声音，设置为 2。

nSamplesPerSec：是配置各个声道播放和收集时的样本频率。如果 wFormatTag= WAVE_FORMAT_PCM，那么 nSamplesPerSec 通常为 8.0 kHz， 11.025 kHz， 22.05 kHz 和 44.1 kHz。例如对于采样率为 11.025 kHz 的音频,nSamplesPerSec 将被设为 11025。

nAvgBytesPerSec：设置请求的平均数据传输率，单位 byte/s。这个值对于创建缓冲大小是很有用的。

nBlockAlign：以字节为单位的块对齐的大小，一般为：

(nChannels*wBitsPerSample)/8

wBitsPerSample：根据 wFormatTag 的类型设置每个样本的位深。如果 wFormatTag= WAVE_FORMAT_PCM，此值应该设为 8 或 16。

cbSize：额外的空间，一般不需要，设置为 0。

按自己的需求创建一个 WAVEFORMATEX 对象来配置音频流的信息。

3.2 声音的采集

声音的采集大体分为三步：

（一）打开设备——waveInOpen（打开一个音频输入设备）

（二）开始录音——waveInStart 开始录音

关闭设备——waveInClose 关闭录音。（在关闭之前先调用 waveInRest，清理正在等待录音的数据缓冲区）

在第一步打开设备之前还需要做许多的准备，程序需要先使用 waveInGetNumDevs 检测程序所在的计算机是否有音频输入设备。如果存在音频输入设备 waveInGetNumDevs 会返回该计算机音频设备的数量。之后通过 waveInGetDevCaps 函数绑定一个音频输入设备进行采集声音。根据需要将音频信息结构体 WAVEFORMATEX 设置好，传入 waveInOpen 函数。再通过 waveInOpen 函数设置回调函数，在回调函数中获取音频数据。当缓冲区的数据被填满后可使用 waveInAddBuffer 函数将缓冲区重新放入采集队列中。下面是录音前准备初始化的代码：

```
bool GWaveIn::IfWaveIn()
{  MMRESULT mmresult = 0;
   mmresult = waveInGetNumDevs();
   if(mmresult == 0) return false;
   WAVEINCAPS waveincaps = {0};
   unsigned int i = 0;
   unsigned int num = mmresult;
   for( ; i<num; i++)
```

```
    mmresult = waveInGetDevCaps(i,&waveincaps,sizeof(WAVEINCAPS));
}
```

以上初始化的任务就完成了，下面是设置音频信息的结构体 WAVEFORMATEX 以及开始录音的代码：

```
 bool GWaveIn::PrepareWaveIn(HWND hwnd, BYTE* pbuf1, BYTE* pbuf2)
{  WAVEFORMATEX waveformatex = {0};
   waveformatex.wFormatTag = WAVE_FORMAT_PCM;
   waveformatex.nChannels = 2;
   waveformatex.nSamplesPerSec = 8000;
   waveformatex.nAvgBytesPerSec = 8000*4;
   waveformatex.nBlockAlign = 4;
   waveformatex.wBitsPerSample = 16;
   waveformatex.cbSize = 0;
   MMRESULT mmresult = 0;
   mmresult                                                    =
waveInOpen(&m_hWaveIn,m_iWaveInID,&waveformatex,(DWORD)hwnd,0,CALLBACK_WI
NDOW);
}
```

该段代码将音频的格式设置为 PCM 格式，声道为双声道。采样频率为 8000Hz，平均的数据传输率为 32000byte/s，最小对齐块为 4，不需要额外空间。设置完音频信息格式后，调用 waveInStart 开始音频采集。只要缓冲区一满，waveInStart 函数，就会主动使用 waveInOpen 函数中制定的函数/窗体/事件；使用该接口，用户能把缓冲区里的波形文件发送给其他的用户，也能将缓冲区的文件储存起来，也就是复制缓冲区。声卡会主动的把音频缓冲区从缓冲队列中撤出。复制完后，就把缓冲区以及相应的音频头文件进行初始化，并通过 waveInAddBuffer 函数从新插入录音缓冲队列里

3.3 声音的播放

当音频数据收集满整个缓冲区后，会发送一个 MM_WIM_DATA 消息给消息队列。声音的播放的初始化有点类似声音的采集的初始化，也是先用 waveOutGetNumDevs 获取音频输出设备的数量，然后使用 waveOutGetDevCaps 绑定一个音频输出设备。最后调用 waveOutWrite，按照设置的 WAVEFORMATEX 音频信息进行播放缓冲区内的音频数据。具体的代码实现如下：

```
bool GWaveOut::PrepareWaveOut(HWND hwnd, BYTE* pbuf1, BYTE* pbuf2)
{    WAVEFORMATEX waveformatex = {0};
     waveformatex.wFormatTag = WAVE_FORMAT_PCM;
```

```
    waveformatex.nChannels = 2;
    waveformatex.cbSize = 0;
    waveformatex.nAvgBytesPerSec = 8000*4;
    waveformatex.nBlockAlign = 4;
    waveformatex.nSamplesPerSec = 8000;
    waveformatex.wBitsPerSample = 16;
    MMRESULT mmresult = 0;
    mmresult =
waveOutOpen(&m_hWaveOut,m_iWaveOutID,&waveformatex,(DWORD)hwnd,0,CALL
BACK_WINDOW);
}
```

以下WaveOutWrite函数负责将语音信号写入播放队列当中，当程序发现播放队列当中有数据了，就将他取出并播放出来。

```
bool GWaveOut::WaveOutWrite(PWAVEHDR pwavehdr)
{   MMRESULT mmresult = 0;
    waveOutWrite(m_hWaveOut,pwavehdr,sizeof(WAVEHDR));
    if(mmresult != MMSYSERR_NOERROR) return false;
    return true;
}
```

3.4 本章小结

本章内容主要先简单的阐述了一下 Windows 下音频处理技术，稍微着重的介绍了一下 wave 系列的 API。之后介绍了使用 wave 系列 API 进行对声音的采集以及播放的过程，及其具体功能的 C++实现代码。

4. 语音传输系统的设计方案

本章主要讨论网络通信的具体实现以及结合语音模块实现网语音通信的功能。本系统会将两个独立的应用分别作为服务端和客户端。所以该系统中将使用的是现阶段应用非常普遍的 C/S 结构，下面就简单的介绍一下 C/S 结构的基本功能和特点。

4.1 C/S 结构的介绍

目前，客户/服务器（Client/Sever，C/S）体系结构在分布式系统上得到普遍的使用。在 C/S 结构下，一个或者多个客户端可以和一个或者多个服务端进行通信。客户端发送请求，通过网络通信传输到服务端，服务端统一处理后把结果再反馈给客户端。这样的好处就是所有重要数据统一由服务端处理，客户端只要负责将收到的结果执行就可以了。

因为客户端和服务端是直接相连的，所以点对点的模式似的它更加安全。C/S 结构还有很多有点，比如它可以直接操作本地文件，减少获取文本的时间和精力。由于直接相连，减少了通信的流量，这样对于客户来说可以减少一大笔通信费用。因为是直接相连，中间也没有什么岔路或者阻隔，所以响应的速度非常快，特别是当通信的数据非常庞大的时候。

不过 C/S 架构也有缺点，由于客户要安装客户端，安装部署困难，所以不容易扩展等。一旦程序有修改，就要重新编写，然后再重新安装部署。当用户数过多时，服务端会出现通信拥堵、响应速度迟缓等现象。

4.2 使用 C/S 结构的语音传输系统的设计方案

该语音传输系统由客户端和服务端两部分组成，双方通过 2 个 Winsock 连接通信。由于 Socket 的接收采用阻塞 recv，当没有接收到数据时线程会处于阻塞状态，等待数据收到为止才会执行下面的操作。因此需要将发送和接受两个功能分开两个线程来运行，否则没接收到数据就无法发送本地数据给对方。两个线程分别拥有一个独立的 Winsock，一个负责接收，一个负责发送。

在 Windows 上使用 Socket 之前需要先加载 Winsock，加载的接口是 WSAStartup（），成功加载后即可使用 Socket。服务端和客户端使用 Socket 前都要构造一个 Socket 并对其初始化，设置为流式 Socket。初始化之后，服务端需要配置好监听地址和端口，这些设置可以通过 SOCKADDR_IN 结构体来进行设置。配置完监听地址和端口之后需要使用 bind（）函数将它和之前创建的 Socket 进行绑定，这样创建出的 Socket 会根据配置的内容进行监听。准备就绪后，服务端则使用 accept（）开始等待客户端的接入。客户端的步骤相比服务端则较为简便，在初始化完之后则可以直接调用 connect（）函数连入服务端。连接上之后双方可以用 recv（）和 send（）函数

进行数据的收发。使用 Socket 结束后，需要调用 closesockt（）将 Socket 关闭释放连接。图 4.1 画出了服务端 socket 的详细流程。

以下是本系统使用 Winsock 的具体代码：

```
bool SocketServer::Init()
{    if (WSAStartup(MAKEWORD(1, 1), &wsa) != 0) return false;
     m_sSocket = ::socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
     if (m_sSocket == INVALID_SOCKET) return false;
     return true;
}
```

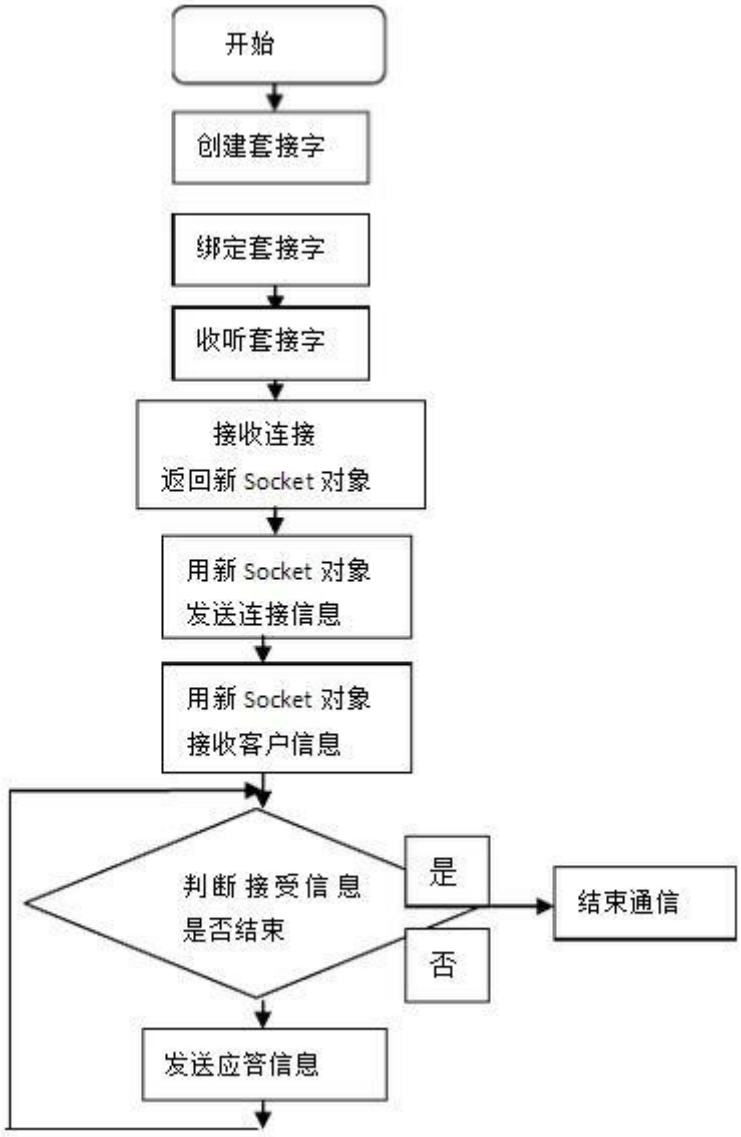

图4.1 服务端通信流程图

```
bool SocketServer::Listen(short port)
{    m_Addr.sin_family = AF_INET;
```

```
    m_Addr.sin_port = htons(port);
    m_Addr.sin_addr.S_un.S_addr = INADDR_ANY;
    if (::bind(m_sSocket, (LPSOCKADDR)&m_Addr, sizeof(m_Addr)) ==
SOCKET_ERROR)
        {closesocket(m_sSocket);return false;}
    if (::listen(m_sSocket, 5) == SOCKET_ERROR)
    {closesocket(m_sSocket); return false; }
    return true;
}
```

服务端的 Socket 初始化完成之后即可开始运行等待新的客户端 Socket 连接。代码中通过一个死循环 while（1）来不停的检测是否有新的客户端 Socket 连入。一旦有新的 Socket 连入，它就会和连入的的 Socket 建立一个稳定的连接。为了防止死循环占用过多的 CPU 资源，在每一次的循环当中都增加了一次 1 毫秒的休眠。下面是服务端 Socket 循环监听新 Socket 连接的代码：

```
bool SocketServer::Run(short port)
{   if (!Listen(port)) return false;
    SOCKET resultSocket;
    int nAddrLen = sizeof(m_Addr);
    while (1)
    {     resultSocket = accept(m_sSocket, (sockaddr*)&m_Addr, &nAddrLen);
        if (resultSocket != INVALID_SOCKET)
        {   SaveSocket(resultSocket);
            m_nLinkNum++;
        }
        Sleep(1);
    }
    return true;
}
```

服务端运行后就是等待客户端Socket连接。客户端Socket连接需要知道目标服务端Socket的具体IP地址以及端口，并设定好自己的连接方式和传输方式。下面是客户端Socket连接服务端的具体代码实现：

```
bool SocketClient::Connect(const char *ip, short port)
{   m_cSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (m_cSocket == 0) return false;
    m_Addr.sin_family = AF_INET;
```

```
    m_Addr.sin_port = htons(port);
    inet_pton(AF_INET, ip, (void*)&m_Addr.sin_addr.S_un.S_addr);
    if (::connect(m_cSocket, (struct sockaddr*)&m_Addr, sizeof(m_Addr)) == -1)
    {
        closesocket(m_cSocket);
        return false;
    }
    return true;
}
```

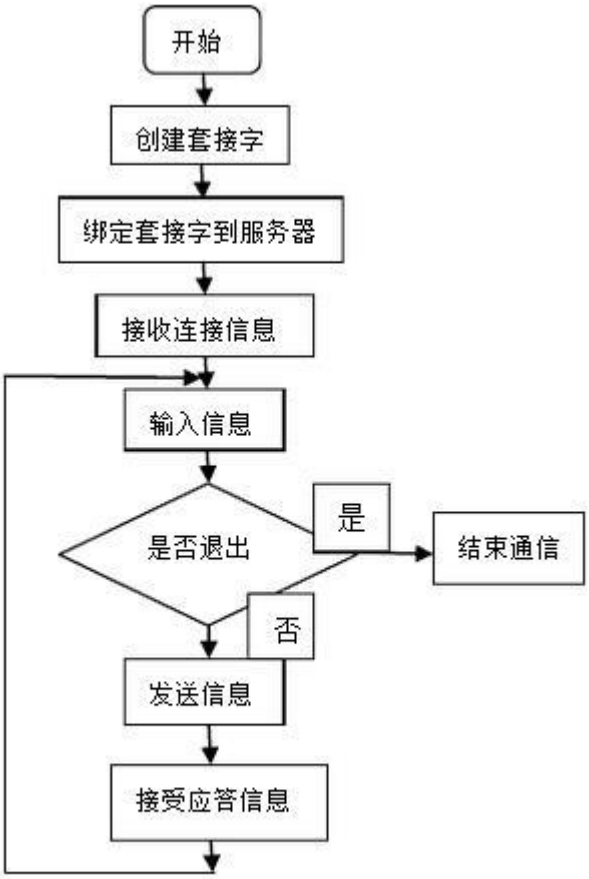

图4.2 客户端通信流程图

在断开Socket之后必须要手动关闭Socket连接，以免程序占用不必要的资源。图4.2列出了客户端通信的具体流程。下面几行简单的代码就是关闭Socket连接的代码：

```
void SocketClient::Close()
{   closesocket(m_cSocket);
    m_cSocket = NULL;
}
```

4.3 加密传输

为了数据的安全性，防止被别人截走数据，该系统对通信模块添加了加密功能。如今加密的算法五花八门，一般分为对称加密算法、非对称加密算法和散列算法。对称加密中比较常见的有 DES、3DES、AES，非对称中有 RSA、DSA、ECC 等，散列算法有比较有名的 MD5 和 SHA 算法。这些算法各有各的优点，考虑到本系统中数据传输比较大，对延时有较高要求。所以选择了和 DES 一样原理的加密算法。该算法采用了 DES 中最精华的部分，自定义了 key 值作为密匙，通过异或位运算，将数据进行处理。使用相同的密匙再次处理该数据则会还原回原来的数据。下面是本系统中加密算法的具体实现：

```
const static char key[16] = "PassWord:#1s$5f";
void SocketClient::SendtoServer(SOCKET socket, int xyid, char* buff)
{
    char str[4096 +3];
    int len = sizeof(str)-3;
    str[0] = xyid;
    str[1] =1 ;
    str[2] =1 ;
    memcpy(str+3,buff,len);
    for (size_t i = 0; i < 4096 + 3; i++)
    {
        str[i] ^= key[i % sizeof(key)];
    }
    ::send(socket, str, len+3, 0);
}
```

4.4 本章小结

本章主要讲述了 C/S 结构的优缺点以及其结构的大概内容。并讨论了该结构的具体实现。讲解了 Socket 通信的具体流程，还介绍 Socket 通信中的关键部分的代码。最后简单的介绍了一下本系统使用的加密方式及其原理。

5. 语音传输系统的实现与结果

客户端和服务端应用程序生成之后，首先将服务端开启，接着打开客户端程序输入服务端的 IP 地址并点击连接即可连上服务器。连上之后一旦有声音数据传来则会立即播放。

5.1 占用过高的 CPU 的解决

由于系统需要不停的进行接收和发送语音数据，所以使用了 While（1）进行了一个死循环，但是如果没有数据接收和发送时，该程序还是在执行循环，导致占用了大量的 CPU 进行做无用功。一个客户端能达到 25%的 CPU 占用率。解决该问题变得非常必要。最后查阅了一些资料发现可以在循环中添加 sleep（1）使程序执行完一次循环内的内容后休息 1 毫秒，该方案大大降低了程序占用 CPU 的使用率。添加 1 毫秒的休眠代码后，程序只占用了 1%的 CPU 占用率。

5.2 语音传输实现结果

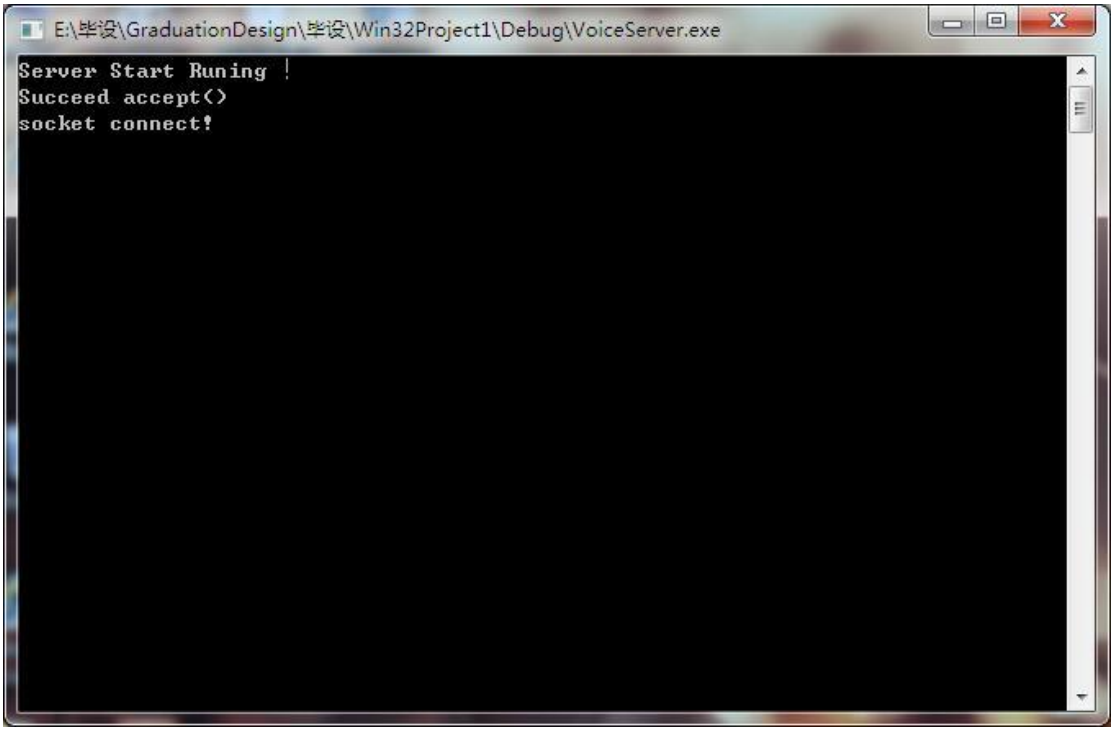

图 5.1 服务端开启后成功连接上客户端

上图 5.1 是服务端成功连接上客户端后的界面显示。通过两台计算机设备多次测试。该系统传输来的语音音质清晰，音频过大时会产生啸叫。延迟较小，使用计算机内存较小，占用 CPU 使用率只有 1%，不会产生卡顿等现象。图 5.2 为客户端连接成功后进行通话时的界面。

图 5.2 客户端成功连接服务端后的界面

5.3 本章小结

本周主要讲述系统完成后测试的结果和亲自体验后的感觉。还讲述了在开发当中遇到的占用 CPU 过高的问题，以及如何解决它。

6. 总结

本文的主要内容就是探讨基于 TCP/IP 协议的网络语音通信系统的原理以及它的具体实现代码。对 Windows 下的语音采集和播放以及 Socket 网络通信接口进行了详细的介绍，并贴出了关键部分的代码实现。接着探讨了本系统采用的加密算法，介绍了它的原理和代码实现。最后的部分讲述了对该系统进行的一些优化处理。生成的出的系统，通过测试交互性良好，音质清晰，延迟非常小，且体积小占用很小的计算机资源。相信在网络语音通信领域有一定的应用价值。

通过这次课题的研究，我除了巩固了从课本上学习到的知识外，还学到了很多课本上没有的知识。学习了网络编程相关的只是，深入了解了其原理，并将其编写出来。本论文的创新点在于，通过提取别的算法精华的部分，自己编写了一套独特的加密算法。不过该系统还有很多需要改进的地方。比如音量较大时会产生啸叫，将自己电脑内的声音也录制进去，很影响用户体验。因此如何抑制啸叫，分离麦克风和电脑声音等问题还是要继续研究。

参考文献

[1] 光文华. 基于局域网的即时通信软件设计与实现[D]. 昆明理工大学, 2014.

[2] 郭锋. 浅析 TCP/IP 网络协议[J]. 商情, 2014(7):345-345.

[3] 马莉. 浅谈网络协议中的 TCP/IP 协议[J]. 数字通信世界, 2016(2).

[4] 杨群. 基于 WI-FI 热点技术的即时 P2P 语音通信系统的研究与设计[D]. 华中师范大学, 2014.

[5] 张晓, 郭弘, 王永全. 即时通讯语音特征研究[J]. 计算机科学, 2014(B10):55-61.

[6] 胡修林, 陈富贵. TCP/IP 网际语音/数据综合传输方法分析[J]. 系统工程与电子技术, 1998(8).

[7] 蒋金弟. IP 网络语音技术及其应用研究[D]. 浙江工业大学, 2001.

[8] 刘光蓉, 周红. 运用 Visual C++构建基于 Socket 的 C/S 模式通信[J]. 武汉轻工大学学报, 2004, 23(3):29-31.

[9] 李文娟. 基于模拟语音的加密技术研究[D]. 西安电子科技大学, 2014.

[10] 刘文琦. 基于 Wi-Fi Direct 语音传输的语音压缩算法的研究与实现[D]. 西安电子科技大学, 2014.

[11] Emerson P. Real Time Voice Communication in Mobile AdHoc Network[J]. D.M.I College of Engineering, 2016.

[12] Yin Q, Zhang J. Development of remote video monitoring system based on TCP/IP[C]// International Conference on Computer Science & Education. IEEE, 2015:596-600.

作者简介

鲍克力，男，汉族，共青团员，浙江省温州市人

教育经历：

2001—2007	平阳县中心小学
2007—2010	平阳县实验中学
2010—2013	鳌江中学
2012—2016	中国计量学院现代科技学院

在校期间：

曾担任：通信 132 班班长。

个人荣誉 ：

2013-2014 学年第二学期浙江省电子设计竞赛成功参赛奖；

2014-2015 学年第二学期全国电子设计竞赛省三等奖；

2015-2016 学年第二学期浙江省挑战杯铜奖；

学位论文数据集

<table>
<tr><td colspan="3">关键词*</td><td>密级*</td><td>中图分类号*</td><td>UDC</td></tr>
<tr><td colspan="3">TCP/IP 协议；语音通信；Win32；</td><td>公开</td><td>TP393.1</td><td>654</td></tr>
<tr><td>论文赞助</td><td colspan="5">无</td></tr>
<tr><td colspan="2">学位授予单位*</td><td>学位授予单位代码*</td><td>学位类别*</td><td colspan="2">学位级别*</td></tr>
<tr><td colspan="2">中国计量大学现代科技学院</td><td>13292</td><td>工学</td><td colspan="2">学士</td></tr>
<tr><td>论文题名*</td><td colspan="3">基于 TCP/IP 协议的网络语音通信系统的实现</td><td colspan="2">论文语种*</td></tr>
<tr><td>并列题名*</td><td colspan="3">Implementation of Network voice communication system based on TCP/IP protocol</td><td colspan="2">简体中文</td></tr>
<tr><td>作者姓名*</td><td>鲍克力</td><td>学号*</td><td colspan="3">1330334215</td></tr>
<tr><td>培养单位名称*</td><td>培养单位代码*</td><td colspan="3">培养单位地址</td><td>邮编</td></tr>
<tr><td>中国计量大学现代科技学院</td><td>13292</td><td colspan="3">浙江省杭州下沙高教园区学源街</td><td>310018</td></tr>
<tr><td colspan="2">专业*</td><td colspan="2">研究方向*</td><td>学制*</td><td>学位授予年*</td></tr>
<tr><td colspan="2">通信工程</td><td colspan="2">语音通信</td><td>4</td><td>2017</td></tr>
<tr><td>论文提交日期*</td><td colspan="5">2017.5.18</td></tr>
<tr><td>导师姓名*</td><td>肖英</td><td>职称*</td><td colspan="3">副教授</td></tr>
<tr><td>评阅人</td><td>洪波</td><td>答辩委员会主席*</td><td colspan="3">陈建国</td></tr>
<tr><td>答辩委员会成员</td><td colspan="5">陈建国、周小微、徐明彪、章乐、裘国华</td></tr>
<tr><td colspan="6">电子版论文提交格式　　文本（ √ ）图像（ ）视频（ ）音频（ ）多媒体（ ）其他（ ）
推荐格式：application/msword；application/pdf</td></tr>
<tr><td colspan="2">电子版论文出版（发布者）</td><td colspan="2">电子版论文出版（发布）地</td><td colspan="2">权限声明</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td colspan="2">论文总页数*</td><td colspan="4">19</td></tr>
<tr><td colspan="6">注：共 33 项，其中带“*”为必填数据。</td></tr>
</table>