# 基于 WINDOWS 95 局域网的语音通信

魏民　许刚　（中国科学院软件研究所　100080）

**摘要**：本文介绍了用 VC 的 MFC 和波形处理 API 在基于 WINDOWS 95 局域网上完成的语音通信，采用 Client / Server 模式和数据流套接字(Stream Socket)，并给出其实现机制和过程。

**关键词**：语音通信　套接字　波形数据

## 一、前 言

由于 WINDOWS 95 系统平台在 PC 用户中使用最为广泛，许多应用局域网的实现是以其为基础的，因此许多用户都需要在基于 WINDOWS 95 的局域网上信息交流和语音通信。利用 VC 的 MFC 和波形处理 API 在基于 WINDOWS 95 局域网上可以完成语音通信。

## 二、局域网的语音通信基本机制

VC 提供两种通信方式：数据报文和数据流。数据报文套接字(基于 TCP 协议)与数据流套接字(基于 UDP 协议)相比有两个限制：不能保证数据的完整送达(有些数据可能不能被所有用户所接受)和不能被 CArchive 类所支持(数据传送必须由程序自己来完成)。因此，采用 C/S 模式和数据流套接字，在 Server 端启动一个守护程序，负责用户管理和语音传送。在 Client 端启动一个谈话事例，设置服务 IP 地址和谈话频道，Server 端在收到 Client 端请求后建立一个新的连接套接字，由 Server 端新的连接套接字与 Client 端进行数据传送。

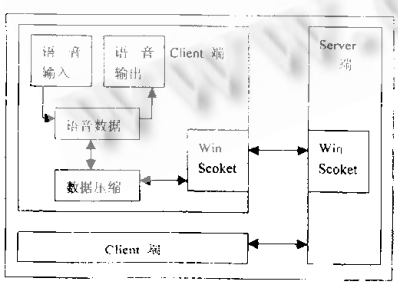


图 1　Windows 95 局域网的语音通信基本机制

语音的输入和输出使用波形处理 API，尽管 MFC 支持 MCI(Media Control Interface)，并且提供播放录制多媒体的标准命令，但是 MCI 只提供了高级接口，不符合语音通信实时性的要求。在使用波形处理 API 中，首先打开输入输出设备，设置数据缓冲区、语音采样频率等参数和回调函数(或回调窗口和事件)，当输入缓冲区数据充满时，系统通知回调函数，在回调函数中读入数据，释放其缓冲区，设置下一个缓冲地址和波形结构头，通知系统其数据准备好并接受下次数据。然后将读入的数据进行压缩，经过串行化，由套接字发送到 Server 端。Server 端把数据转发到谈话频道的所有用户。最后在收到数据的 Client 端，经过串行化，解压数据，放入波形输出数据缓冲区，播放语音。系统结构如图 1 所示：

## 三、数据流套接字的通信方法

由于数据流套接字是基于连接的协议，所以首先建立连接，然后才能从数据流中读数据，数据流套接字的一般用法如图 2 所示：

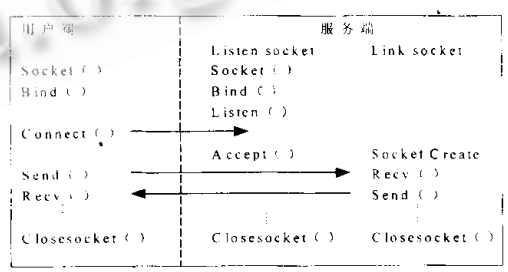


图 2　C/S 方式下的套接字使用方法

MFC 的 CScoket 类是由 CAsyneSocket 派生而来，为 WinSock API 提供了一个更高级别的接口，它使用 CArchive 类简化关于套接字的输入输出。而且在处理 Windows 消息时，会处理阻塞调用。首先创建 CSocket

对象,调用构造函数构造一个空的 CSocket,然后调用 BOOL Create( UINT nSocketPort = 0, int nSocketType = SOCK-STREAM, LPCTSTR lpszSocketAddress = NULL ); 对于 Client 端套接字,一般设置缺省参数就足够了,对于 Server 端套接字,通常需要设置端口号。然后建立连接,使用基类的 Connect(),Listen()和 Accept()函数来建立 Server 端和 Client 端套接字的连接。创建了 Csocket 对象,并连接后,可以用 Sendto()/Receivefrom()和 Send()/Receive()调用发送和接受数据。在 CSocket 类的两端可以连接 Carchive 对象进行数据传递。通常使用 CArchive 对象通信时,要创建一个 CsocketFile 与 Carchive 连接。其方法如下:

```
CSocket MySocket;
CSocketFile * pMyFile;
CArchive * pArchiveOut;
CArchive * pArchiveIn;
MySocket.Create();
MyScoket.Connect("SERVER",5150);
pMyFile = new CsocketFile (&MySocket, TRUE);
//create separate archives for input and output
pArchiveIn = new CArchive(pMyFile, CArchive::load);
pArchiveOut = new CArchive(pMyFile , CArchive::store);
```

创建两个 CArchive 对象,一个用于接受数据用 CArchive::load 创建,另一个用于发送数据用 CArchive::store 创建。当数据写入到档案文件后,必须调用 CArchive::Fluse()来刷新缓冲区数据,否则连接端点会一直等待数据接收。使用完 CSocket 对象后,调用 Close()释放套接字有关资源,包括套接字的缓冲区。

## 四、语音的输入和输出

SoundOut 类提供输出信号的接口,SoundIn 类提供输入信号的接口。两者在实现上十分相似,在这里主要说明 SoundOut 类。CSoundOut::OpenOutput() 是主程序,初始化声卡的所有参数,建立一个 EVENT 体和开始声音的线程。为了叙述方便只保留了主要的函数的调用,省略了设备检查,错误检查和线程通信,方法如下:

```
MMRESULT CSoundOut::OpenOutput()
{
    MMRESULT result;
    WaveInitFormat ( 1/ *  mono * /, m-WaveOutSam-
pleRate / * khz * /,16 / * bits * /); // Open Output
    result = waveOutOpen ( &m-WaveOut, 0, &m-Wave-
Format, ( DWORD ) m-WaveOutEvent, NULL, CALL-
BACK-EVENT); //设置线程 EVENT
    result = waveOutPrepareHeader( m-WaveOut, &m-
WaveHeader, sizeof(WAVEHDR) );
    result = waveOutWrite ( m-WaveOut, &m-Wave-
Header, sizeof(WAVEHDR) );
    result = waveOutRestart( m-WaveOut );
    return result;
}
```

另外还有两个重要的程序是 AddBuffer()和 CloseOutput(),方法如下:

```
void CSoundOut::AddBuffer()
{
    MMRESULT result;
    result = waveOutUnprepareHeader ( m-WaveOut,
&m-WaveHeader, sizeof(WAVEHDR) );
    //设置新的数据缓冲区
    result = waveOutPrepareHeader( m-WaveOut, &m-
WaveHeader, sizeof(WAVEHDR) );
    result = waveOutWrite ( m-WaveOut, &m-Wave-
Header, sizeof(WAVEHDR) );
    result = waveOutRestart( m-WaveOut );
}

void CSoundOut::CloseOutput()
{
    waveOutReset(m-WaveOut);
    waveOutClose(m-WaveOut);
}
```

其中最重要的结构是 WAVEHDR,定义如下:

```
typedef struct {
    LPSTR lpData;
    DWORD dwBufferLength;
    DWORD dwBytesRecorded;
    DWORD dwUser;
    DWORD dwFlags;
    DWORD dwLoops;
    struct wavehdr-tag * lpNext;
    DWORD reserved;
} WAVEHDR;
```

在 Client 端首先打开输出设备后，播放声音数据，当缓冲区数据播放完成时系统发出一个 EVENT 事件，声音线程得到时，调用 AddBuffer( ) 函数，其中 waveOutUnPrepareHeader( ) 完成释放原数据当缓冲区，在 WAVEHDR 结构中由 lpData 指针指向新数据当缓冲区，由 waveOutPrepareHeader( ) 通知系统数据缓冲区准备好，waveOutRestart( ) 通知系统开始播放数据。一般播放时需要三个缓冲区，当一个缓冲区用于播放时，另外的缓冲区准备数据，这样才能保证数据的读写不发生冲突以及数据读写的连续性。WaveOutOpen( ) 支持多种回调方式，包括事件，线程，窗口和过程。可以通过设置参数的最后一项的标志来实现不同的回调方式。

SoundIn 类的实现与 SoundOut 类基本一致，主要的函数变成 waveInAddBuffer ( )，waveInClose ( )，waveInOpen( )，waveInPrepareHeader( )，waveInReset( )，waveInStart( )，waveInStop( ) 即可。

## 五、结 论

在基于 WINDOWS 95 的局域网上采用 Client / Server 模式，数据流套接字 (Stream Socket) 和波形处理 API 可以很好的完成语音通信。但是，由于系统在写满语音输入缓冲区后，在新缓冲区和旧缓冲区交换时，有一定的间隙从而产生输入数据的泄漏。因此，当缓冲区设置较小时，语音质量有所下降。

## 参 考 文 献


[1] David Bennett，徐军，《Visual C + + 5 开发人员指南》机械工业出版社 1998.9

[2] Peter Norton, Rob McGregor，《MFC 开发 Windows 95/NT 4 应用程序》清华大学出版社 1998.4

[3] 袁松青，苏建泉，《数字通信原理》人民邮电出版社 1996.5


（来稿时间：1999 年 2 月）