局域网中多方话音传输的实现

李 强 陆 捷 仰爱兰 李晓星（合肥电子工程学院计算机教研室）

摘要：本文主要讨论了在局域网中实现多方话音的传输和播放，以及实现过程中出现的问题及其解决方法。

关键词：话音 TCP/IP

一、引 言

在研制开发《RF－3200电台网络仿真综合训练系统》中，我们以单台计算机模拟单部RF－3200电台，利用计算机网络环境模拟多部电台组成的电台网络。RF－3200电台具有话音通信功能，因此就需要利用计算机网络实现话音数据的实时传输，来完成电台话音通话的功能。

二、系统功能及实现方法

在局域网中传输数据通常有两种模式，一种是通信双方直接建立连接，通信双方直接传输数据；另一种模式通过服务器转发，通信双方不直接连接，而是都与服务器建立连接，服务器收到数据后再转发给相应的接收方，以实现双方的通信，这种模式便于实现多方通信。

在我们的应用系统中，电台所组成的网络中通信方个数以及通信双方都是动态变化的，因此我们采用了后一种模式。

在网络的数据传输过程中我们使用了TCP/IP协议，通话方与服务器的连接采用了Socket的TCP连接，TCP连接是一种可靠的点对点的连接，它具有出错重发机制，防止数据在传输过程中丢失。话音数据的传输模型如图1所示。

图 1

其中：

话音录制模块主要通过声卡完成话音的录制，并将数据存放录音缓冲区中。

数据处理及压缩模块主要将录音缓冲区中已录制的数据进行打包和压缩处理，然后通过 Client Socket 由发送进程将数据包发往服务器。

数据转发模块位于服务器上，主要完成数据的转发功能。Server Socket收到数据包后先将数据放入接收缓

冲区中，并不直接发送出去，而由发送进程每次从接收缓冲区中顺序取出数据包，根据数据包的包头通过Server Socket发给相应的接收方。

数据处理及解压缩模块通过Client Socket收到服务器转发来的数据包后进行解包和解压缩处理，然后放入接收缓冲区中，以便播放模块播放话音数据。

话音播放模块主要将接收缓冲区中的话音数据通过声卡顺序播放。

三、确保话音播放连续的解决方法

话音数据的传输特点是数据量大，这样即使在千兆的局域网中传输有时也会因为数据流量的增大造成网络的阻塞，而造成话音的中断；另一个特点是实时性高。话音录制后，如果将数据存盘以文件的形式传输数据，这样增加了读、写盘的时间，而增加了两块话音数据块间的间隔时间，也造成了话音播放的不连续。

为了解决以上的问题，我们采取了一些技术处理。一是录制的话音数据不存盘，打包和压缩后直接发送，以消除因读、写而增加的数据块间的时间间隔；二是数据的收发方设置一定数量的缓冲区；三是采用了顺序延时的方法。

1. 发话方缓冲区大小和数据量的确定

发话方的缓冲区是指录音缓冲区。录音数据量的大小与录音质量密切相关，录音质量可以选择电话质量(PCM格式中电话质量的数据格式为8位单声道、采样频率为11025Hz)来减少数据量。另外每一数据缓冲区块不应选择太大，一方面太大的缓冲区会增加数据传输时间，当数据发送错误时，会明显增加重发次数，另一方面因录音缓冲区必须录满数据后才能发送，所以太大的缓冲区会使收方播放滞后的时间增加。一般每块缓冲区大小选择可存放0.5秒的话音数据(大约5K左右)，缓冲区块数可选择3至5块，块数太多会占用过多的内存资源。

2. 在服务器转发模块中设置缓冲区

我们在每个数据 包中增加了一个包头，以便服务器根据包头信息将收到的数据包转发给相应的受话方。当服务器进行转发数据时，就会有新的数据到来，如果不设置一定数量的缓冲区，将收到的数据包立即进行转发，此时若前一个数据包还没有转发完毕，就会造成前一个数据包丢失，数据就不能完整地到达受话方。设置缓冲区后，将新到的数据包放入缓冲区中，在确保前一个数据包发送完毕后，再发送缓冲区中下一个数据包。

受话方设置一定数量的接受缓冲区，以便存放话音播放过程接受到的后续话音数据包。

3. 话音播放不连续的现象

经过分析问题还是出在网络传输环节上。虽然发话方在不停地发送数据，由于网络传输的延时，受话方在播放完一块话音数据后，接受缓冲区中就有可能没有待播放的话音数据，便会出现话音播放中断。在整个过程中，我们看到话音数据是以数据流的方式进行传输的。话音进入计算机后，进行压缩处理，经过网络传输到达受话方，再解压缩，最终播放，话音数据都是连续的不间断的。如果受话方将收到的第一个数据包不马上播放，当数据包积累到一定数量后再开始播放，这样后续的数据都顺延一个时间段播放，只是稍稍增加发话方与受话方之间的滞后时间。当然这个滞后时间不能太长，应以能削除话音因网络传输造成的播放中断为宜，否则会造成明显的延时，在实际应用中这个滞后时间一般在1秒左右。

另外在话音的录制和播放过程中没有使用文件的方式，就是为了减少读写盘的时间，也是为保证话音播放连续性而采取的一个措施。因此就必须使用录音和放音的API底层函数。下面给出相关函数（用C++ Builder 实现）。

```
//录音及放音格式设置
void WaveDevSet()
{
    PCMWaveFmtRecord->wf.nChannels = 1;
//单声道
    PCMWaveFmtRecord->wf.wFormatTag = 1;
//PCM格式
    PCMWaveFmtRecord->wf.nAvgBytesPerSec =
11025;//平均数据传输速率
    PCMWaveFmtRecord->wf.nSamplesPerSec =
11025;//采样速率
    PCMWaveFmtRecord->wf.nBlockAlign = 1;
//数据的对齐方式
    PCMWaveFmtRecord->wBitsPerSample = 8;
//采样的位数
}
//打开录音设备
void WaveInDevOpen(DWORD hwnd)
{
    ErrorCode = waveInOpen(&hWaveIn, 0,
PCMWaveFmtRecord, (DWORD)Form1->Handle,
0, CALLBACK_WINDOW);
```

```
    if (ErrorCode!=0) ShowMessage("Record Device is error!");
}
```

以回调方式打开录音设备，当一个录音缓冲区满时向指定的窗口发送一个 MM_WIM_DATA 消息，以便对话音数据进行处理。

```
//录音
void WaveInRecord()
{
    //分配录音缓冲区
     hMem = GlobalAlloc(GMEM_ZEROINIT|GMEM_MOVEABLE,MemSize);
     WaveInHeader.lpData = (LPSTR)GlobalLock(hMem);
    WaveInHeader.dwBufferLength = MemSize; //设定录音缓冲区的大小
    WaveInHeader.dwFlags = 0;
    WaveInHeader.dwLoops = 0;
    //将已分配的缓冲区加入到录音队列中
     ErrorCode = waveInPrepareHeader(hWaveIn, &WaveInHeader, sizeof(WAVEHDR));
      ErrorCode = waveInAddBuffer(hWaveIn, &WaveInHeader, sizeof(WAVEHDR));
    //开始录音
    ErrorCode = waveInStart(hWaveIn);
}
```

当一个录音缓冲区满后，向指定的窗口发送 MM_WIM_DATA 消息，表明一个录音缓冲区已满，我们对数据进行如下处理：

```
case MM_WIM_DATA:
    DataPro(); //数据打包及压缩处理过程
        GlobalUnlock(hMem); //释放录音缓冲区
       //准备新的录音缓冲区
        WaveInHeader.lpData = (LPSTR)GlobalLock(hMem);
        WaveInHeader.dwBufferLength = MemSize;
        WaveInHeader.dwFlags = 0;
        WaveInHeader.dwLoops = 0;
       ErrorCode = waveInPrepareHeader(hWaveIn, &WaveInHeader, sizeof(WAVEHDR));
        ErrorCode = waveInAddBuffer(hWaveIn, &WaveInHeader, sizeof(WAVEHDR));
//打开播放设备
void WaveOutDevOpen()
{
    ErrorCode = waveOutOpen(&hWaveOut, 0, PCMWaveFmtRecord, (DWORD)Form1->Handle, 0, CALLBACK_WINDOW);
    if (ErrorCode !=0 ) ShowMessage("Wave_output device is error.");
}
```

以回调方式打开播放设备，当一个话音数据播放后向指定的窗口发送一个 MM_WOM_DONE 消息。

```
//话音的播放
void WaveOutPlay(int Index)
{
     WaveOutHeader.lpData = (LPSTR)DataBuf[Index]; //指定播放缓冲区
     WaveOutHeader.dwBufferLength = MemSize; //指定缓冲区的大小
    WaveOutHeader.dwFlags = 0;
    WaveOutHeader.dwLoops = 0;
    //准备播放队列
    ErrorCode = waveOutPrepareHeader(hWaveOut, &WaveOutHeader, sizeof(WAVEHDR));
    //播放数据
    if (ErrorCode == 0)
        ErrorCode = waveOutWrite(hWaveOut, &WaveOutHeader, sizeof(WAVEHDR));
}
```

四、结束语

经过上面的技术处理后，话音录制完后再经过网络的传输到达受话方播放比较平滑，达到比较满意的效果。■

参考文献

1 贾罗尔著 Visual Basic 多媒体开发指南 科学出版社龙门书局 1996 年