网络语音与视频技术实现

叶长青 张 嘉
(华东师范大学教育信息技术系,上海 200062)

摘 要 为了适应语音和视频在网上传输的需要,利用微软公司 DirectX 系统下的 DirectSound 和 DirectShow,研究实现了语音和点对点视频在网络上的传输,并探讨了开发中相关的注意点。

关键词 网络语音 网络视频 DirectX DirectSound DirectShow
中图法分类号: TN915.4 **文献标识码**:A **文章编号**: 1006-8961(2005)11-1398-04

Realization of Network Voice and Video Technology

YE Chang-qing, ZHANG Jia
(*Department of Education Information Technology, East China Normal University, Shanghai* 200062)

Abstract For transportation of voice and video over network, uses Microsoft's DirectSound and DirectShow subsystem of DirectX to study and realize network transmission of voice and point-to-point video. Discusses related issues in software development.

Keywords network voice, network video, DirectX, DirectSound, DirectShow

1 引言

随着计算机网络与音视频压缩技术的进步,以及网络带宽的持续拓展,网络音视频类产品呈现出蓬勃发展的态势。从 OICQ、MSN(microsoft network)等视频聊天工具的普及到基于 IP 的视频会议系统的应用,网络语音与视频技术的广泛市场前景带动了大批企业进入这一领域,同时音视频技术让网络基础运营商看到了宽带关键应用的方向。

微软推出的 DirectX 提供了与网络音视频应用相关的 SDK(system develop kit),而开发者以此为跳板则可以较方便地实现语音与视频的捕捉、网络传送,解压、回放等功能,并可以根据需要来定制和扩充应用。

2 基于 DirectSound 的网络语音实现

DirectSound 是 DirectX 系统的成员之一,利用它可以播放、处理及捕捉声音,并可使应用程序对硬件资源进行高级控制。其优点是速度快、延时短、可控制性强。

下面是一个捕捉音频数据、网络发送、接收方接收并播放的处理全过程及技术要点释义。

2.1 定义语音捕捉类

```
class CDSoundCapture
{…
public:
  BOOL InitDirectSound( HWND hDlg );
  //初始化 DirectSound
  BOOL
  SetCaptureFormat( WAVEFORMATEX* pwfx );
  //设置音频数据的格式
  BOOL
   CreateCaptureBuffer ( WAVEFORMATEX* pwfxInput );
//建立音频数据缓冲区
  BOOL StartOrStopRecord( BOOLbStartRecording );
  //开始或停止捕捉音频数据
```

基金项目:华东师范大学"十五""211"重点科研项目(BAA030012)
收稿日期:2005-08-17;**改回日期**:2005-09-23

第一作者简介:叶长青(1962~),男。1999 年获华东师范大学硕士学位,现为华东师范大学副教授。主要研究方向为远程教育、多媒体。E-mail:ycq@deit.ecnu.edu.cn

```
  BOOL RecordCapturedData( ); //捕捉数据
  …
}
```

其中,成员函数 InitDirectSound 对与声音捕捉相关的参数进行初始化设置。

```
BOOL CDSoundCapture::InitDirectSound( )
{…
  CoInitialize( Null ); //初始化 COM 库
  //获得语音捕捉对象指针
  DirectSoundCreate( Null,g_pDSCapture,Null );
  //为声音数据格式结构申请一块内存:
  CreateCaptureBuffer( &g_wfxInput );
  //设置采样频率
  SetCaptureFormat( &g_wfxInput );
}
```

考虑到网络带宽,将采样频率设置得低一些可以减少数据量。

```
BOOL
CDSoundCapture::SetCaptureFormat(WAVEFORM
ATEX* pwfx)
{
  pwfx -> nSamplesPerSec = 8000;
  //采样频率,可选 8000, 11025, 22050 或//44100
  pwfx -> wBitsPerSample = 16;
  //采样位数
  pwfx -> nChannels = 1;//声道数
  pwfx -> nBlockAlign = pwfx -> nChannels * (pwfx ->
wBitsPerSample / 8 );//块对齐
  pwfx -> nAvgBytesPerSec =
pwfx -> nBlockAlign * pwfx - > nSamplesPerSec;
  //每秒数据量
}
```

2.2 建立网络传输机制

可以定义一个专门用于发送,和一个专门用于接收的 SOCKET 类。其中 SockSend 负责将捕捉到内存缓冲区 DataBuffer 中的声音,发送给指定 IP 地址和端口的其他用户;SockRece 用于从网上接收声音,并放在内存缓冲区 DataBuffer2 中以备播放。

2.3 接收和发送功能实现

接收功能设置:

```
//初始化 COM 组件库
CoInitialize( NULL );
//使用当前默认声音设备
DirectSoundCreate( NULL, &m_pDS, NULL );
CreatePlayBuffer( );//创建播放缓冲区
//创建套接口,用于接收声音
SockRece. Create(4344,SOCK_DGRAM);
将接收到的数据放在 DataBuffer2 中;
将 DataBuffer2 作为参数传给
FillPlayBufferWithSound;
//从接收数据缓冲区里取数据播放
StartOrStopPlay( );
//当缓冲区非空时播放
pDSBuffer -> Play( );
```

发送功能设置:

```
//创建套接口用于发送声音
CapturerSock. Create( );
循环检测录音设备,捕捉声音,声音数据送设定的缓冲区。
//循环捕捉声音,等待缓冲区准备好
g_pDSBCapture -> Start( DSCBSTART_LOOPING );
将缓冲区里的数据发送给指定的 IP 地址。
```

以上示例中,由于没有对数据进行压缩处理,也没有对回音进行控制,所以有一定延时,并有回音现象的发生。

3 点对点的网络视频传输

DirectShow 也是 DirectX 系统的成员之一。利用 DirectShow 实现网络视频传输可以有多种应用形式。从视频数据的流向看,首先是视频信号的采样,可以称作视频信号采集或捕获过程,然后是数据压缩、网络传输和接收方的数据解码和回放。为了保证数据采集速度与数据处理速度匹配,一般要开辟内存缓冲区,这样将缓冲区中的数据就地保存成指定格式的文件,就实现了录像功能。如果保存的视频数据只是一个特定帧,则可以看作是拍照功能的实现。由于以上两种功能常常工作在本地,因此技术上实现起来比较容易。但如果将内存缓冲区里的数据经由网络发送出去,则会带来以下一系列的研发课题:视频点播、直播、IP 电话、视频会议等等。

DirectShow 将一些与硬件或底层打交道的过程封装成一个个 COM 组件,在 DirectShow 中称为 filter,基于 DirectShow 的应用过程就是将合适的 filter 连接成从数据源到回放窗口的一个通道,称为 filtergraph。问题是,DirectShow 提供的现成的 filter 很可能不能满足人们的具体需求。因此,许多情况下我们需要按照一定的规范开发自己的 filter。例如,图 1 是视频头捕捉图象的预览的 filtergraph,图 2 为本地预览窗口。本文仿制一个 filter,用于每一帧

数据的反色处理,并在源 filter 之后插入一个 Smart_Tee filter 分流视频数据,其中一路仍走图 1 所示的线路形成视频预览,而另一路则将视频流的备份通过自定义的 filter 后进行播放(图 3),这时就可以看见经过处理的视频图象对比效果(图 4)。

图 1 视频预览时的 filter 连接

Fig. 1 Filter connection in preview

图 2 预览窗口

Fig. 2 Preview window

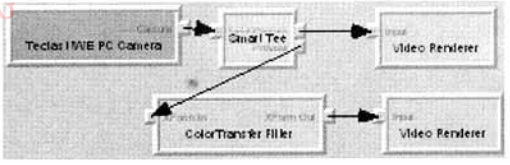

图 3 自定义 filter 加入 filtergraph 的情形

Fig. 3 User-defined filter adds in

为了完成简单的点对点的网络视频传输,需要两台电脑各自从摄像头捕捉视频图像发送给对方,就像 QQ 里的视频聊天那样。要实现这样的功能,可以仿照上面的示例,也设计一个专用 filter,这次是将分流的视频数据发送给指定的 IP,而接收方接收到数据后,在回放出来的同时将本地视频也发送回对方,就可以实现双方的视频交流。由此可以发现,自行设计特定的 filter 对于定制应用是个关键。

自制的 filter 是一个 COM 组件。每个 filter 中至少有一个称为 pin 的接口,负责与其他 filter 连接。向指定的 IP 地址和端口发送数据的专用 filter 类具体设计过程如下:

图 4 反色处理后的效果

Fig. 4 Color inverse effect

(1) 定义发送 filter 中 pin 类:

一个输入 pin 是一个类(COM 组件),其中主要成员函数是 Receive(COM 接口),它负责接收从前面传来的数据包(在 DirectShow 中称为 Sample)。

```
class CInputPin : public CbaseInputPin
{ public :
  CInputPin(...);
  ~CInputPin ();
  HRESULT GetMediaType( IN int iPos, OUT CMediaType* pmt );//pin 上的媒体类型
  HRESULT CheckMediaType(IN const CMediaType* pmt);
//pin连接时的媒体类型检查
  STDMETHODIMP Receive( IN IMediaSample* );
  ...
};
```

相应的还有输出 pin,作为下一级 filter 的接口与输入 pin 连接。filter 之间通过 pin 相互连接,构成一条顺序的链路,即 filtergraph。

(2) 定义专用 filter:

```
class MyFilter : public CbaseFilter,
    public ImulticastConfig, //配置多播接口
    public IspecifyPropertyPages, //属性页
    public CpersistStream //持久信息
{ ...
  CInputPin * m_pInput; //输入 pin
  MyFilter * m_pNetSender; //发送 filter
  ULONG m_ulIP; //IP 地址
  USHORT m_usPort; //端口号
  public :
```

```
  HRESULT Send (IN IMediaSample*);
  ...
}
```

(3) 实现 filter 中的发送成员函数:

```
HRESULT MyFilter:: Send ( IN IMediaSample* pIMediaSample )
{
 BYTE* pbBuffer;
 HRESULT hr;
 //获得 buffer 当前指针然后发送 samples
 hr = pIMediaSample -> GetPointer(&pbBuffer);
 if( SUCCEEDED(hr) )
 {
  hr = m_pNetSender -> Send( pbBuffer, pIMediaSample -> GetActualDataLength () );
 }
 return hr;
}
```

(4) 发送数据操作过程:

```
HRESULT CNetSender:: Send ( IN BYTE* pbBuffer, IN DWORD dwLength )
{  HRESULT hr;
   DWORD   dw;
   DWORD   dwSnarf;
   int     i;
   if( m_hSocket != INVALID_SOCKET )
   {
      while( dwLength > 0 )
      {
         dwSnarf = Min < DWORD >
         (m_dwPTTransmitLength, dwLength );
         //然后发送
         i = sendto( m_hSocket, pbBuffer,dwSnarf,0,
            &m_saddrDest, sizeof(m_saddrDest));
         if(i != (int)dwSnarf )
         {
            dw = GetLastError();
            return HRESULT_FROM_WIN32(dw);
         }
         pbBuffer += dwSnarf;
         //缓冲区指针后移
         dwLength -= dwSnarf;
         //缓冲区长度缩短
      }
   hr = S_OK;
   }
   else{
      hr = E_UNEXPECTED;
   }
   return hr;
}
```

4　开发注意事项

在网络传输过程中,语音对延时、抖动比较敏感,而对误码则相对不敏感;视频要求高带宽,并对实时性要求较严,但允许有误码。

因此,为了符合音视频业务的要求,开发时应考虑如下事项:

(1) 服务质量 (quality of service, QoS)。当网络过载或拥塞时,QoS 能确保重要业务量不受延迟或丢弃,同时能保证网络的高效运行。一个典型的例子就是 H.323 视频会议协议利用 RTCP(real time control protocol)监测 QoS;

(2)实时性,这是音视频通信与传统数据传输的本质区别,应根据上述音视频的不同特征和网络带宽,正确处理好延时和误码量。

5　结　论

(1)使用 DirectSound 和 DirectShow 不仅可以大大简化网络音视频传输软件的开发过程,而且两者可以完成数据的主要采集处理过程,这样开发人员可以集中精力提高网络传输的质量,以提高程序质量。

(2)用 DirectSound 和 DirectShow 开发的程序扩展性较强。例如在 DirectShow 应用开发中,可以通过在 filtergraph 中加入自定义的 filter 来实现许多附加功能,如附加字幕、音效处理、视频特效等。

参考文献(References)

1 LIU Wei-wei. Visual C++ Audio/Video develop utility case choiceness[M]. Beijing: Posts&Telecom Press, 2004. [刘炜玮. Visual C++视频/音频开发实用工程案例精选[M]. 北京:人民邮电出版社,2004]

2 LU Qi-ming. DirectShow develop guide[M]. Beijing: Tsinghua University Press, 2003. [陆其明. DirectShow 开发指南[M]. 北京:清华大学出版社, 2003.]

3 DirectX for C++ [EB/OL]. http://msdn.microsoft.com/archive/enus/directx9_c/directx/directx9cpp.asp 2005.