 摘要 本文针对不同的局域网，提出一种通用的实时视频传输的解决方案。在使用Divx编解码的基础上，提出了从压缩、组帧、发送到接收、解压整个流程的思想，具体实施方案和VC++实现核心源代码以及传输控制策略，有效地保证了高质量的实时视频传输。  
  
　　关键词 客户/服务器;实时视频传输;Divx  
  
　　 引言  
  
　　 在局域网内部实时传输视频已经得到广泛应用。现在用以传输视频的局域网大多数是有线局域网，因为有线局域网技术成熟，传输速度快，稳定性好。但是视频数据量大，有线网络也会出现工作不稳定，引起数据堵塞，时间久了会导致严重的延迟现象;如果工作的环境不固定，要求移动性，那么就要采用无线网络，如今无线网卡的工作随环境的变化而变得不稳定，这样会导致视频传输的质量大幅度下降，容易引起画面的重影、抖动、花屏等现象。本文针对不同的局域网，提出一种通用的实时视频传输的解决方案，使用VC++自封装的Windows VFW SDK[软件开发](http://www.programbbs.com/doc/)包进行二次开发，通过Divx编解码，按照制定的传输策略，能够有效地解决由于网络的局部不稳定导致的视频图像重影、抖动、花屏等的问题。  
  
　　在局域网中实时视频传输存在的问题  
  
　　 为了在局域网上有效的、高质量的传输视频流，需要多种技术的支持，其中包括视频的压缩、编码技术，应用层质量控制技术等等。  
  
　　 网络的带宽是有限的，所以需要压缩传输视频图像，MPEG-4被广泛的应用于网络环境下的实时视频传输，因为MPEG-4具有：可以达到很高的压缩比;具有灵活的编码和解码复杂性;基于对象的编码方式，允许视频、音频对象的交互;具有很强的容错能力等优点。本文采用Divx编解码器对视频进行编码、压缩，实际上Divx=（视频）MPEG-4+（音频）MP3。  
  
　　 应用层质量控制技术现在采用的是RTP/RTCP协议，以确保视频流在网络中低时延、高质量地传输。RTP数据传输协议负责音视频数据的流化和负载，RTCP负责RTP数据报文的传输控制。此协议是通过客户端（接收方）反馈网络的状况，服务器端（发送方）来调整信息采集、发送的速度和压缩率。但是，对于图像采集速度固定，需要软件进行压缩、解压，调整采集的速度会引起采集的数据来不及压缩而直接丢弃，调整编码器的压缩率需要重新设置编码器的参数，重启编码器，相应的解码器也要调整，这个过程中需要很长的时间，达不到实时的要求。所以本文没有采用RTP/RTCP协议，而是从发送端出发，实时判断网络状况，采用“停等”策略进行实时传输。  
  
　　 网络通信有两种协议TCP和UDP，UDP更适合于网络环境下的视频传输，但是它不提供检错和纠错功能，一旦网络出现堵塞时，大量的数据报文会丢失。对于Divx编解码技术，是以帧为单位进行编解码的，分为关键帧和非关键帧。在传输过程中，由于压缩率比较高，只要一帧中错一比特位，将影响其它几百甚至几千的比特位，直接造成图像的模糊、花屏等现象。只有等到下一次关键帧的到来才有可能恢复图像的清晰。为了保证传输的正确性，自己需要在应用层制定协议。如此一来，UDP的优势荡然无存。所以本文选择使用TCP来进行网络通信。综合使用VFW技术、流媒体技术，辅助以“停等”控制策略，较好的解决局域网中实时视频传输容易引起的重影、抖动、花屏的问题。  
  
　　实时视频传输实现  
  
　　 为了达到视频传输的实时性，总的思想是最少的发送冗余信息，最大程度上发送最新的视频。  
  
　　 局域网实时视频传输采用服务器/客户机模式，利用VC++实现。其工作流程如图1所示。

图1 实时视频传输工作流程

　　 视频采集采用AVICap从视频采集卡捕获视频图像，得到的是位图型式的视频帧，然后用Divx编码器进行压缩，通过Winsock实现压缩后的视频数据在局域网中的实时传输，接收完的数据交给Divx解码器解压，最后实现视频显示。  
  
　　 在VC++中，采用VFW技术，客户端通过capSetCallbackOnFrame()注册回调函数，当采集卡采集到一幅图像后，系统就会自动调用回调函数，然后再回调函数中使用ICSeqCompressFrame()函数进行压缩。然后再通过Winsock将压缩后的数据发送到服务器端。服务器端接收完一帧以后，交给ICDecompress()解压，最后用SetDIBitsToDevice()将图像显示出来。  
  
　　 1、视频帧的组建  
  
　　 视频采集的数据是位图型式的视频帧，Divx编码器压缩以后形成以帧为格式的Mpeg4流。Divx解码器也是以帧的格式解压。所以提出以帧为单位发送视频数据流。为了在接收端能够方便地提取出一帧，提出如图2所示的格式组建帧。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 帧开始标志 | 帧大小 | 帧编号 | 帧类型 | 帧数据 |

图2 视频帧格式  
  
　　 完整的一帧由5个字段组成，各个字段的意义如下：帧开始标志，标志着一帧地开始，占用4个字节的空间。不妨设为0xffffffff。帧大小，表示整个帧的大小，包括5个字段的大小，占用4个字节的空间。帧编号，表示帧的顺序编号，占用4个字节的空间。帧类型，标志此帧是否是关键帧，占用1个字节的空间。帧数据，存放压缩后一帧的完整数据。  
  
　　 2、视频帧的发送  
  
　　 实时视频传输为了达到实时，要不断地将压缩好的数据发送到接受端。所以在发送端创建一个线程，专门用来发送数据。同时主线程仍然不停的采集数据并进行压缩。发送线程的工作流程如图3所示。

图3 发送线程工作流程

　　 不妨假设创建的线程名为sendThread，其核心代码实现如下：

|  |
| --- |
| while(1) { 　 isOK=true; //准备就绪  　 SuspendThread(sendThread); //挂起线程  　 isOK=false; //线程正在发送数据  　 int length=frameLength; //待发数据长度  　 if(length <50000) {//判断数据是否正常  　　 int n=0;  　　 int sendCount=0;  　　 while(length >0) { 　　　 n=send(sock,(char\*)imageBuf+sendCount,length,0); //发送数据，  　　　 //imageBuf是指针，指向待发数据帧  　　　 if(n==SOCKET\_ERROR) //网络出现异常，则退出线程  　　　　 break;  　　　 length-=n;  　　　 sendCount+=n;  　　 }  　 }  } |

　　 线程中发送的数据帧是按照上一节中的方法组建好的数据帧。这种方法能够保证正在发送的当前帧能够完整地到达接收端。  
  
　　 注意此线程中刚开始或者每当发送完一帧以后，线程就转到挂起状态，等待外界唤醒。这个任务由回调函数完成，在回调函数中，判定如果发送线程准备就绪（处于挂起状态），则进行图像压缩，然后唤醒线程发送压缩完的数据，否则直接跳出，等待下一次调用回调函数，这种策略称之为“停等”策略，在后面有详细介绍。  
  
　　 3、视频帧的接收  
  
　　 接收端最重要的是从接受的数据流中提取出完整的一帧。方法的思想是：首先从数据流中寻找帧开始标志，再从紧挨后面的数据中提取出帧的大小，然后再从接收缓冲区中读入该帧剩余的数据。再寻找下一帧的开始标志，如此往复。图4是接收端的工作流程。

　　 同样接收端创建一个线程专门用来执行数据接收。不妨假设线程名为recThread，核心代码实现如下：

|  |
| --- |
| while(temp!=SOCKET\_ERROR) { 　 if(!isStart) {//帧数据是否开始，true表示开始  　　 if(endNum >3) //endNum纪录当前接收未处理的数据 　　　 endNum=0;  　　 temp=recv(clisock,(char\*)(recBuf+endNum),1000,0);//从缓冲区读取数据  　　 startPos=serchStr(temp+endNum); //查找帧开始标志  　　 if(startPos!=-1) {  　　　 isStart=true;  　　　 endNum=temp+endNum-startPos-4;  　　　 memcpy(imageBuf,recBuf+startPos+4,endNum); //保存帧数据  　　 }  　　 else{  　　　 memcpy(recBuf,recBuf+temp+endNum-3,3);//保存最后三个字节的数据  　　　 endNum=3;  　　 }  　 }  　 else{  　　 if(endNum <4) {//判定紧跟开始标志的数据，如果小于4表示不能获得帧大小 　　　 temp=recv(clisock,(char\*)(recBuf),1000,0); //读入数据  　　　 memcpy(imageBuf+endNum,recBuf,temp);//保存数据  　　　 endNum+=temp;  　　　 if(endNum <4) 　　　　 continue;  　　　 frameSize= \*((int\*)imageBuf);//获得帧大小  　　　 if(frameSize <500 || frameSize>50000) {//异常处理（帧大小非法） 　　　　 isStart = false; //丢弃数据重新查找帧开始标志  　　　　 endNum = 0;  　　　　 continue;  　　　 }  　　　 frameSize-=endNum+4;  　　 }  　　 else{  　　　 while(frameSize >0&&temp!=SOCKET\_ERROR) {//获得完整帧的剩余数据 　　　　 temp=recv(clisock,(char\*)(imageBuf+endNum),frameSize,0);  　　　　 endNum+=temp;  　　　　 frameSize-=temp;  　　　 }  　　　 if(frameSize <=0) {//帧结束置位，解压  　　　　 isStart=false;  　　　　 endNum=0;  　　　　 deCompress();//判断数据的有效性，调用ICDecompress进行解压  　　　 }  　　 }  　 }  } |

　　 以上[程序](http://www.programbbs.com/doc/)执行的结果是将完整的一帧（除帧开始标志）保存在imageBuf中。  
  
　　 4、“停等”控制策略  
  
　　 如果局域网通信速率很高，而且工作稳定，则按照以上说的方法进行实时视频传输，不需要任何控制策略，就可以达到非常好的效果。但是在很多情况下，网络会出现异常，这样会导致数据传输率明显下降，造成发送端数据积压，等待发送的数据不能正常发出去。此时就要采取一定的策略来控制发送端，以达到实时性的要求。  
  
　　 上文发送[程序](http://www.programbbs.com/doc/)中，变量isOK是用来表示发送端当前帧有没有发完，如果发完则置为true，同时也表示发送端准备就绪，可以继续发送数据，否则为false。那么可以用isOK来通知视频采集和压缩线程，如果isOK为true，则可以采集视频并且压缩，然后唤醒发送线程继续发送新来的帧数据，否则一直等待，直到网络可以继续发送数据（isOK为true）。当然，视频采集一直不停的进行，那么当网络发生数据堵塞时，只要不让编码器进行压缩则可解决;当网络恢复正常时，继续进行压缩传输，换句话说，当网络发生堵塞时，直接抛弃等待发送的帧，保证一旦网络恢复时，发送最新的压缩帧。当然要保证一旦有一帧开始发送，就要将其完全发出。  
  
　　 按照这样的“停等”策略进行实时视频传输，只会带来一个问题：当网络质量差时，接收端画面中的移动目标会出现瞬间移动的现象。但是这种策略会保证不会出现重影，抖动，花屏等现象。  
  
　　结论  
  
　　 本文提出的实时视频传输方案在100M的局域网、10M局域网和11M无线局域网中进行了测试。测试时让一个目标在镜头前（发送端）移动，观察接收端视频的显示。在不同的局域网中进行了多次测试，每次测试时间从10分钟到30分钟不等，并且改变目标的运动速度进行实验。最后将数据汇总，得出统计结果。测试结果如表1所示。  
  
　　 表1 不同局域网下的测试结果

|  |  |  |  |
| --- | --- | --- | --- |
|  | 剧烈运动 | 正常运动 | 缓慢运动 |
| 100M局域网 | 图像清晰，很流畅 | 图像清晰，很流畅 | 图像清晰，很流畅 |
| 10M局域网 | 偶尔出现停顿，丢帧率1%左右 | 图像清晰，人眼感觉流畅 | 图像清晰，很流畅 |
| 11M无线局域网 | 经常出现停顿，丢帧率5%-6% | 经常出现停顿，丢帧率2%-3% | 偶尔出现停顿，丢帧率1%左右 |

　　 其中，   
  
　　 注：11M无线网卡是通过USB1.0接口和PC机连接的，如果采用USB2.0接口效果会更好。  
  
　　 从实际测试的结果看，效果是良好的，除了出现瞬间移动外，图像能够保持清晰，消除了由于网络质量差而导致的重影、抖动等现象，对于不同的局域网都能满足实时传输的要求。