基于JRTPLIB库的rtp包的传输和接收学习

2009-09-09 21:21

|  |
| --- |
| 转载：<http://xiyong8260.blog.163.com/blog/static/665146212008824112748499/> 基于JRTPLIB库的RTP数据传输设计文档  RTP即实时传输协议，用于Internet上针对多媒体数据流的传输。它通常使用UDP协议来传送数据，起初是为了“multicast”传输情况而设计的，目的是提供时间信息和保证流同步，不过现在也用于一对一的传输情况。RTP协议主要完成对数据包进行编号，加盖时戳，丢包检查，安全与内容认证等工作。通过这些工作，应用程序会利用RTP协议的数据信息保证流数据的同步和实时传输。 windows下vc6.0编译rtp库，过程如下：  压缩包可以从这里获得：<http://www.bairuitech.com/upimg/soft/jrtplib-3.7.1.rar>  下载jrtplib-3.7.1.rar后，首先将其解压到一个临时文件夹中，然后开始后续工作。  首先需要强调的是，jrtplib是一个库而不是应用程序，编译后我们获得的是.lib文件。这个文件是用来实现RTP协议的，意义和我们在写WIN32程序时用到的kernel.lib一样。  解压后的文件夹中包含两个目录，jrtplib-3.7.1和jthread-1.2.1，打开这两个目录后我们可以看到下面又有两个同名的目录，为了后面能顺利编译，我们把同名目录下的文件全部考到上一级目录中，就是说把f:\jrtplib-3.7.1\jrtplib-3.7.1\\*.\* 复制到f:\jrtplib-3.7.1\。同理，把f:\jthread-1.2.1\jthread-1.2.1\\*.\* 复制到f:\jthread-1.2.1\  完成上述步骤后我们就可以开始编译库文件了。  Windows平台下建议使用Visual C++6.0。  首先编译多线程库jthread，在vc6中直接打开工作区文件jthread.dsw，改变工程设置，选中source file下的文件，点右键选择setting，确保code generation下Use run-time library 为debug mulitithreaded DLL或debug mulitithreaded。  然后选build就可以了，和上面一样的方法完成jrtpthread的编译。编译时需要用到jthread中的头文件，需要将其包含到工程中，这个底下的文件比jthread多一些。  默认产生的文件是jthread.lib和jrtplib.Lib,这两个文件分别位于两个文件夹下的debug文件夹下，将它们复制到VC6的lib文件夹下。 完成上述工作后我们就可以开始尝试编译jrtplib附带的examples。  创建一个新的Win32 Console 应用程序项目，添加example文件到source files文件夹中，然后设置VC环境 1、 修改example.cpp文件，在文件开始添加  #pragma comment(lib, "jrtplib.lib")  #pragma comment(lib, "jthread.lib")  #pragma comment(lib, "WS2\_32.lib")  或者在VC中a) Project->Settings->Link中Object/library modules:添加jthread.lib jrtplib.lib， b) Link中添加ws2\_32.lib 2、 检查 工程-设置-C/C++-Use runtime library选择code generationdebug mulitithreaded DLL或debug mulitithreaded，方法同上文中检查库文件的方法。 3、 工具-选项-目录-include files 将两个库中头文件所在文件夹包含进来 工具-选项-目录-library files 将生成的两个静态库所在文件夹包含进来 工具-选项-目录-src files 将两个库中src文件夹包含进来  最后就可以编译、连接、生成可执行文件了。  rtp包传送和接收过程 1)        初始化  在使用JRTPLIB进行实时流媒体数据传输之前，首先应该生成RTPSession类的一个实例来表示此次RTP会话，然后调用Create()方法来对其进行初始化操作。RTPSession类的Create()方法只有一个参数，用来指明此次RTP会话所采用的端口号。  2)        数据发送  当RTP会话成功建立起来之后，接下来就可以开始进行流媒体数据的实时传输了。首先需要设置好数据发送的目标地址，RTP协议允许同一会话存在多个目标地址，这可以通过调用RTPSession类的AddDestination()、DeleteDestination()和ClearDestinations()方法来完成。目标地址全部指定之后，接着就可以调用RTPSession类的SendPacket()方法，向所有的目标地址发送流媒体数据。SendPacket()最典型的用法是类似于下面的语句，其中第一个参数是要被发送的数据，而第二个参数则指明将要发送数据的长度，再往后依次是RTP负载类型、标识和时戳增量。 sess.SendPacket(buffer, 5, 0, false, 10);  对于同一个RTP会话来讲，负载类型、标识和时戳增量通常来讲都是相同的，JRTPLIB允许将它们设置为会话的默认参数，这是通过调用 RTPSession类的SetDefaultPayloadType()、SetDefaultMark()和 SetDefaultTimeStampIncrement()方法来完成的。为RTP会话设置这些默认参数的好处是可以简化数据的发送，例如，如果为 RTP会话设置了默认参数： sess.SetDefaultPayloadType(0); sess.SetDefaultMark(false); sess.SetDefaultTimeStampIncrement(10); 之后在进行数据发送时只需指明要发送的数据及其长度就可以了： sess.SendPacket(buffer, 5);  3)        数据接收  对于流媒体数据的接收端，首先需要调用PollData()方法来接收发送过来的RTP或者RTCP数据报。由于同一个RTP会话中允许有多个参与者（源），因此既可以通过调用GotoFirstSource()和GotoNextSource()方法来遍历所有的源，也可以通过调用GotoFisstSourceWithDat()和GotoNextSourceWithData()方法来遍历那些携带有数据的源。在从RTP会话中检测出有效的数据源之后，接下去就可以调用RTPSession类的GetNextPacket()方法从中抽取RTP数据报，当接收到的RTP数据报处理完之后，要及时释放。下面的代码示范了该如何对接收到的RTP数据报进行处理： do{      sess.BeginDataAccess();             if (sess.GotoFirstSourceWithData())             {                      do                      {                               RTPPacket \*pack;                               uint8\_t \*data;                               size\_t length;                               if((outfile=open("/mnt/hgfs/download/temp1.YUV",O\_WRONLY | O\_CREAT | O\_APPEND))<0)                               {                                         perror("open outfile error");                                         return -1;                               }                               while ((pack = sess.GetNextPacket()) != NULL)                               {                                         data=pack->GetPayloadData();                                         length=pack->GetPayloadLength();                                         if((write(outfile,data,length))<0)//将接收到的图片信息写入文中。                                         {                                                  perror("write outfile error;");                                                  return -1;                                         }                                         sess.DeletePacket(pack);                               }                      } while (sess.GotoNextSourceWithData());             }             sess.EndDataAccess();             close(outfile); JRTPLIB为RTP数据报定义了3种接收模块，通过调用RTPSession类的SetReceiveMode()方法可以设置下列这些接收模式：  RECEIVEMODE\_ALL：缺省的接收模式，所有到达的RTP数据报都将被接受；  RECEIVEMODE\_IGNORESOME：除了某些特定的发送者之外，所有到达的RTP数据报都将被接受，而被拒绝的发送者列表可以通过调用AddToIgnoreList()，DeleteFromIgnoreList()和ClearIgnoreList()方法来进行设置；      　RECEIVEMODE\_ACCEPTSOME：除了某些特定的发送者之外，所有到达的RTP数据报都将被拒绝，而被接受的发送者列表可以通过调用AddToAcceptList()，DeleteFromAcceptList和ClearAcceptList()方法来进行设置。 |