# 编译Emule with VS2008

## [qedit.h中dxtrans.h找不到错误解决方法](http://blog.csdn.net/huang_xw/article/details/8588629)

原因: 当前版本的direct sdk已经将DirectShow分离了，当前版本的sdk没有dxtrans.h文件  
**解决方法:**  
在include "dxtrans.h"前加上  
  
#define \_\_IDxtCompositor\_INTERFACE\_DEFINED\_\_   
#define \_\_IDxtAlphaSetter\_INTERFACE\_DEFINED\_\_   
#define \_\_IDxtJpeg\_INTERFACE\_DEFINED\_\_   
#define \_\_IDxtKey\_INTERFACE\_DEFINED\_\_   
并将dxtrans.h注释掉

## [Vs2008的bug：error C2471: cannot update program database vc90.pdb](http://blog.csdn.net/lly20000/article/details/7428041)

|  |
| --- |
| 编译找不到而且不能升级vc90.pdb文件的问题解决方法  问题描述：  1>e:\vcomh264net\vcomh\vcomh264net2.0 1.0.25 av\video\videoproc.cpp : error C2471: cannot update program database 'e:\vcomh264net\vcomh264net2.0 1.0.25 2008\vcomh264net2.0 1.0.25 av\debug\vc90.pdb'  1>e:\vcomh264net\vcomh\vcomh264net2.0 1.0.25 av\video\videoproc.cpp : fatal error C1083: Cannot open program database file: 'e:\vcomh264net\vcomh264net2.0 1.0.25 2008\vcomh264net2.0 1.0.25 av\debug\vc90.pdb': No such file or directory        为什么我把一个VC6的工程转换为VS2008的工程后，编译总是出现找不到而且不能升级vc90.pdb文件的问题。重新编译了也不行。  从VC6直接跳到VS2008  这个vs2008一个著名的bug。详情可以参见<https://connect.microsoft.com/VisualStudio/feedback/ViewFeedback.aspx?FeedbackID=309462>  官方现有解决方案如下：         I have found an alternate way for the time beging to avoid C2471 error but it works only in the case of successful release build.       for this click Build menu than Configuration manager than create a new setting from release settings. Change following things in your project settings as :  C\C++ | General | Debug Information format | C7 Compatible (/Z7)  C\C++ | Code Generation | Enable String Pooling | Yes (/GF)  Linker |Debuging |General Debug Info | Yes (/DEBUG)          after these settings. Rebuild your application. I'm sure you will be able to resolve the pdb updation issue if it is really not hampering your other settings. Hope it will be helpful to you. |

[**在vs2008中编译（调试）eMule0.50a以及veryCD版的easyMule**](http://blog.csdn.net/jo_say/article/details/8070756)

标签： [windows](http://www.csdn.net/tag/windows)[library](http://www.csdn.net/tag/library)[正则表达式](http://www.csdn.net/tag/%e6%ad%a3%e5%88%99%e8%a1%a8%e8%be%be%e5%bc%8f)[google](http://www.csdn.net/tag/google)[开发工具](http://www.csdn.net/tag/%e5%bc%80%e5%8f%91%e5%b7%a5%e5%85%b7)[.net](http://www.csdn.net/tag/.net)

2012-10-14 22:54 2217人阅读 [评论](http://blog.csdn.net/jo_say/article/details/8070756#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/jo_say/article/details/8070756#report)

本文章已收录于：

http://static.blog.csdn.net/images/category_icon.jpg分类：

P2P（13） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpghttp://static.blog.csdn.net/images/arrow_triangle_up.jpg

[作者同类文章](http://blog.csdn.net/jo_say/article/category/819578)*X*

Cpp & C（38） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpghttp://static.blog.csdn.net/images/arrow_triangle_up.jpg

[作者同类文章](http://blog.csdn.net/jo_say/article/category/901293)*X*

版权声明：本文为博主原创文章，未经博主允许不得转载。

--------------------------------------------------------

关键字：

* Vs2008
* eMule0.50a
* easyMule
* 调试
* atlrx.h
* windows SDK 7.0
* 增量链接
* ITaskbarList3
* LNK2005
* 无法解析的外部符号
* LIBCMT and LIBCMTD
* MTD
* /FORCE:MULTIPLE

----------------------------------------------------------

Background：

目的是看看kAD的具体运算，因此下了eMule的最新版0.5a, 源代码的结构如果之前不了解，还是不太容易弄懂，因为开源软件一般都会有较多的依赖库。如果你在linux下debug过开源软件，你就知道库依赖的重要性了。以前在windows下太顺利，所以遇到一点挫折，如果google不出来就急的不行。eMule的源代码要想编译，是必须依赖几个lib的，他们一般也都是通过项目编译而来，试图从网上直接找lib和头文件的，反而太累人，因为你不知道后面需要多少个头文件，nnd，太多了。

【看看这个帖子：<http://blog.csdn.net/colinchan/article/details/4648844> 会增加一些了解，但是版本不同，需要解决的问题可能也就不同了】

按照网上大多数文章的说法，具体如crypto,libpng,resizablelib,zlib等都需要google，然后到官方下，不过这些一般都没有vc9（vc2008）的项目版本，直接用08打开转换即可，很少出问题。但是：事情总是没有那么顺利，不是找不全依赖lib的源项目，就是找到了编译出一堆错。本着前人种树，后人乘凉的想法，我直接g最好的解决方案，试了试，并没有发现完全适合我的环境的（win7+vs08+emula0.50a). 不过还是要感谢CSDN，资源里给了一个：【电驴最新版源码VS2008直接编译】【这个也是我最后编译eMule的重要帮手】

<http://download.csdn.net/download/WS355362/3378757>

* 直接编译（包含了lib解决方案）

但是这里的电驴不是国外eMule-project的那个，而是国内改造的easyMule。发布者给了很给力的说明，按其走着就可以了。

      ---------------

      编译说明（推荐使用Visual Studio .NET 2008）  
     1.将rcdll.dll复制到Visual Studio .NET 2008安装目录的Vc\bin目录中。（这是为了使用能在vista下显示的图标）  
     2.打开easyMule\_vs2008\_Libs.sln，执行“生成解决方案”。（easyMule\_vs2008\_Libs.sln里所包含的是easyMule所依赖的库文件。）  
     3.打开easyMule\_vs2008.sln编译即可。

      ---------------

* Atlrx问题：

上面编译easyMule时出现了atlrx.h找不到的问题，如果你g下，会发现这个问题很普遍，是其中正则表达式用到的一个库，下面的这篇帖子告诉你怎么解决，无非是下载库，然后链接而已。

<http://blog.csdn.net/wu_huiwen/article/details/5523128>。

* 增量链接问题

运行时，vs2008又出现了增量链接出错的问题，到属性->链接器->增量链接中关闭其，ok了

* ITaskbarList3问题

这些问题也都是普遍问题，g下就会有解决方案，其实这里是windows7任务栏问题，需要安装对应的windows SDK, 看下面帖子吧：

<http://blog.csdn.net/x313695373/article/details/6068603>。 还有一个无法打开数据库文件的问题，这个帖子也有回答。不瞒您说，这些能遇到的问题我几乎都遇到了，不过还有万能的search engine，要不咱怎么办呢。。。

---------------------------------------------------------------------------------------------------------------------------------------

解决了以上问题，如果正常的话，easyMule是可以编译运行的，至少我成功了。

----------------------

但是我知道easyMule的改造还是蛮大的，虽然提高了eDonkey的一些默认服务连接点，增加了很多资源选择界面。但是实在是不喜欢那么大红大紫的界面，还有那来来往往的广告。决定还是要编译好原始的eMule0.50a。

刚才编译easyMule之所以简单，是因为它的配套库都有了，但是它虽然基于eMule，但是还是有较大变换，甚至源代码的组织都几乎完全不同。

然后事情总没有那么坏，至少现在我可以告诉你，他们的依赖库几乎是可以通用的（我说几乎，表明还是有例外，如后面我要说的CxImage等）

--------

于是我做了一下事情：

1. 直接把easyMule项目里面的生成lib的解决方案：crypto51，crypto55, resizableLib，zlib, CxImage等全部改拷到了eMule源代码下（按照结构，它们是和srchybrid并列的，当然CxImage是默认放在srchybrid下的。
2. 根据这些项目的位置以及产生的lib位置，调整eMule项目文件属性中链接器->输入->附加依赖项中的具体内容（它默认是有内容的，你要根据当前情况来调整）[注意，两个点开头表示上两级目录，一个点表示上一级目录]

|  |
| --- |
| version.lib winmm.lib ws2\_32.lib gdiplus.lib ADSIId.lib crypt32.lib wmvcore.lib ..\zlib\debug\zlib.lib ..\crypto51\crypto\debug\cryptlib.lib ..\id3lib\libprj\id3libD.lib .\CxImage\debug\cximage.lib ..\libpng\projects\visualc71\Win32\_LIB\_Debug\libpngd.lib ..\ResizableLib\DebugUnicode\ResizableLib.lib .\miniupnpc\debug\_vc9\miniupnpc.lib |

1. 然后仔细确认各个依赖库生成顺利，边编译eMule，毫无疑问，错误是一大堆的。主要有：

-----------------------

* 直接移过来的CxImage项目的输出函数MakeBitmap是单参数的，但在eMule0.50a中有一处是两个参数调用。这说明了eMule0.50a对应的那个CxImage版本的MakeBitmap函数（xImage.h定义，ximawnd.cpp中实现）是两个参数的。但这里为了简单，我就把eMule0.50a中的代码中两个参数的第二个给去掉了，这个参数是反应透明与否的，是关于界面，并不影响我对eMule网路的了解，所以果断cancel之。如果反之，即修改CxImage项目为两个参数，会出现【无法解析的外部符号】这样的错误，事实上，这个问题恼了我一个小时，我最终也没有解决之，所以我才不得已采用了上面的方法。
* error LNK2005: \_\_initp\_misc\_invarg 已经在libcmtd.lib(invarg.obj) 中定义。这是因为项目链接了两个库LIBCMT and LIBCMTD，以至于出现already defined。解决办法是：   
  1.确保当前工程的Runtime Library和引用lib的Runtime Library相同便可以解决问题   
  比如：lib库的Runtime Library是MD，当前工程也得设置为MD。 (我这里全是MTD）  
  2.In Debug mode，在Debug命令行中加上/FORCE:MULTIPLE。（我在属性->配置->链接的命令行和调试的命令中都添加了，应该只需要一处吧，我懒得比较了）.方法来自：<http://weihe6666.iteye.com/blog/1405657>
* 至此，基本上ok了。编译，成功1个，警告n个，哎，总算可以歇会了。

-------------------------

还出现的其他问题是，我试图用release模式时，系统编译时提示我zlib.lib库文件找不到（因为链接到了一个没有的路径）。

我这里其实是投机取巧，把用在easyMule上的库直接用到eMule0.5上，的确简短了我去网上找对应windows版本，然后编译的过程。

其可实现原因和注意点有：

* 主要是这些库都很成熟，多数已经出现很久，切已很少更新，所有这里easyMule和eMule用到的可以一致。
* 还有就是easyMule和eMule这里我都用vs08编译，直接挪过来，不冲突。
* 需要注意的libpng在easyMule中不是这样的，它是pnglib，我也懒得转换，这个是从官网下的，解压后为lpng1513文件夹，从里面深入多层后，可以找到vs7项目，转换，生成即可。
* 另外easyMule用的是crypto55，但应该eMule还是用crypto51.
* 还有CxImage项目编译是需要png文件夹的，由于都是从easyMule项目中copy过来，所以不妨把png folder给copy过来到CxImage同级目录，方便编译。
* 安装windows SDK时，有点慢，不要急。还有只需要安装开发工具和头文件即可。
* 我的系统是windows7。
* 很多地方是debug\_v9，在填写是注意，其实都无所谓，但是要一致。

--------------------

以上只是我回头看记起来的，其实我在编译过程中，痛苦的读了大量的国外论坛帖子和国内论坛帖子，对我的理解还是很有帮助的。

部分帖子如下：

http://forum.emule-project.net/index.php?showtopic=149213  
http://www.cnblogs.com/jianu/archive/2010/12/25/how\_to\_compile\_eMule\_in\_vs2008.html  
http://blog.csdn.net/cecilulysess/article/details/4631458  
http://jiangliwei.blogbus.com/logs/28509863.html

<http://blog.csdn.net/wu_huiwen/article/details/5523128>

## How To Build Emule 0.49C With Vs 2008

I built eMule 0.49c under Visual Studio 2008.  
You can use the archive I provide instead of getting the single source packages one-by-one.  
  
ed2k://|file|eMule.0.49c.VS2008.rar|5513085|7813FC47A0BED0FD5406C9B80D5A8331|h=ESXJ7CXOCFW3YIBOUKRDHQN5EZGBTCTD|/  
  
  
The system I used was equipped as follows:  
  
- Windows XP Professional SP3  
- Visual Studio 2008 SP1  
- Windows SDK v6.1 (Windows SDK for Windows Server 2008 and .NET Framework 3.5, build name: 6.0.6001.18000.367)  
- DirectX 9.0 SDK (October 2005)  
- whole system MSUpdated to 2010-02-08  
  
Under this configuration, neither Microsoft Speech SDK 5.1 nor WMFormat SDK9 was needed.  
  
  
Solution doesn't build with these SDKs:  
  
WIN SDK  
- 6.1.7600.16385; Microsoft Windows SDK for Windows 7 and .NET Framework 3.5 Service Pack 1  
- 6.1.6000.16384.10; Windows Vista Update & .NET 3.0 SDK  
  
DirectX SDK  
- 9.27.1734 Aug 2009  
  
NOTES:  
- many releases of DirectX SDK can be installed, if you don't want to mess-up your machine, just avoid to install run-time components if they're obsolete.  
- check DX SDK and Win SDK paths in Visual Studio options!!!!  
  
SOURCES  
I used the latest releases, just google the name to find its home page.  
  
eMule v0.49c  
  
ATL Server v9.0.70425 Alpha  
Crypto++ v5.60  
CxImage v6.0.0  
id3lib v3.8.3  
libpng v1.4.0  
ResizableLib v1.3  
zlib v1.2.3  
  
(reported only for thoroughness reason, you don't need it to rebuild eMule)  
Flex v2.5.35 - it generates scanner.cpp/scanner.h  
  
NOTE:  
ATL Server is needed because VS2008 doesn't support it anymore,  
I preferred to place it in the project, rather than under VS components.  
  
NOTE ON STEPS:  
It's useful to verify differences between subsequent steps by a comparison tool (i.e. examDiff).  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
STEP1 - merging eMule and libs source in project root  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
The aim of this step is to obtain all the sources merged and organized under a root folder.  
  
to obtain STEP1:  
  
extract eMule sources in the project root folder:  
  
root  
- EditBinX  
- id3lib  
- ReplaceVistaIcon  
- srchybrid  
- zlib  
  
remove id3lib and zlib folders:  
  
root  
- EditBinX  
- ReplaceVistaIcon  
- srchybrid  
  
extract ORIGINAL libs in the project root folder:  
  
root  
- ATL  
- cryptopp  
- EditBinX  
- id3lib  
- libpng  
- ReplaceVistaIcon  
- ResizableLib  
- srchybrid  
- zlib  
  
remove root\id3lib\zlib folder  
  
NOTES:  
- id3lib will reference to zlib in the project root  
- in the STEP1 I provide, id3lib and CxImage source files are the originals, then replaced by PATCH1 with the ones in eMule 0.49c archive.  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
STEP2 - patching sources  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
The aim of this step is to patch the source files which need to be adjusted.  
  
to obtain STEP2, copy PATCH1 over STEP1, overwriting files with the same name.  
  
sources modified in STEP2 by PATCH1 are:  
  
- some id3lib modified in the original eMule 0.49c package  
- some CxImage modified in the original eMule 0.49c package  
- some adjust for include directive paths ("crypto51" to generic "cryptopp")  
- some adjust to comply a couple of fields renamed in the struct png\_info (libpng 1.4.0 - png.h)  
- one adjust in PeerCacheFinder.cpp to comply the change of returned value of Encode(), from crypto++ 5.60  
- file root\id3lib\config.h (it's root\id3lib\config.h.win32 renamed, see root\id3lib\libprj\win32.readme.first.txt)  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
STEP3 - converting project/solution/workspace files to VS2008 format  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
The aim of this step is to obtain each sub-project buildable in VS2008, not the whole eMule solution.  
  
to obtain STEP3, copy PATCH2 over STEP2, overwriting files with the same name.  
  
here we get:  
- the emule\_vc9-all.sln (the whole eMule solution)  
- each sub-project (libs and the single eMule prj) in VS2008 format  
  
note:  
- emule\_vc9-all.sln is already adjusted under "configuration manager" / "build order" facet  
- all include/dependence paths are adjusted in each sub-project file  
  
For those prj files that needed conversion to VS2008, I used the newest format available in their own original package.  
For each sub-project, I took care of only the 2 config (release-win32/debug-win32) used in eMule-all solution.  
Since some libray are solutions themselves, I took care of only their .vcproj used in eMule-all solution.  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
STEP4 - building eMule solution  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
The aim of this step is to build emule\_vc9-all.sln in both configs: release-win32/debug-win32  
  
to obtain STEP4, copy PATCH3 over STEP3, overwriting files with the same name.  
  
PATCH3 modifies these options:  
- libpng change RuntimeLibrary to /MT - /MTd  
- ResizableLib change CharacterSet to unicode (to comply original 0.49c for VC7.1)  
- zlib remove preprocessor define ZLIB\_WINAPI  
- emule remove AdditionalOptions /safeseh  
  
Release builds with 757 warnings,  
Debug builds with 757 warnings.

# 跟踪CAsyncSocketEx类族

1、CeMuleAPP::InitInstance()🡪CServerConnect::CServerConnect()🡪InLocalIP(),m\_nLocalIP=192.168.1.115

2、CServerListCtrl::OnNmCustomDraw()根据服务器列表反复调用🡪CServerConnect::GetCurrentServer()，

3、eMuleDlg::StartupTimer()🡪CListenSocket::StartListen()

3.1🡪CAsyncSocketEx::Create (UINT nSocketPort /\*=0\*/, int nSocketType /\*=SOCK\_STREAM\*/,long lEvent /\*=FD\_READ | FD\_WRITE | FD\_OOB | FD\_ACCEPT | FD\_CONNECT | FD\_CLOSE\*/, LPCSTR lpszSocketAddress /\*=NULL\*/, BOOL bReuseAddr /\*=FALSE\*/ )

3.1 .1🡪CAsyncSocketEx::InitAsyncSocketExInstance(){ m\_spAsyncSocketExThreadDataList==NULL,执行else分支},

3.1.2🡪CAsyncSocketEx::AttachHandle(SOCKET /\*hSocket\*/)🡪 CAsyncSocketExHelperWindow:: AddSocket(CAsyncSocketEx \*pSocket, int &nSocketIndex){ m\_nWindowDataSize=512, nSocketIndex=-1, m\_nSocketCount=0, m\_nWindowDataPos=0, m\_pAsyncSocketExWindowData[i % m\_nWindowDataSize].m\_pSocket == NULL,执行if语句块一次，为各变量赋值，nSocketIndex=0, m\_nSocketCount=1, m\_nWindowDataPos=1, m\_pAsyncSocketExWindowData[i % m\_nWindowDataSize].m\_pSocket = 0x0494b3a8 {CListenSocket}, return true;}

3.1.3🡪 CAsyncSocketEx::AsyncSelect(long lEvent /\*= FD\_READ | FD\_WRITE | FD\_OOB | FD\_ACCEPT | FD\_CONNECT | FD\_CLOSE\*/=8(FD\_ACCEPT)){ m\_lEvent=8(FD\_ACCEPT)，WSAAsyncSelect(m\_SocketData.hSocket, GetHelperWindowHandle(), m\_SocketData.nSocketIndex + WM\_SOCKETEX\_NOTIFY, lEvent)，return true}

3.1.4🡪bReuseAddr=0🡪 CAsyncSocketEx::Bind(UINT nSocketPort=55358, LPCSTR lpszSocketAddress=NULL){ sockAddr.sin\_addr.saddr=0.0.0.0}

3.2🡪CAsyncSocketEx::Listen(int nConnectionBacklog=5)

4、CUploadQueue::UploadTimer()🡪CServerConnect::KeepConnectionAlive(){if(false)},定时调用，保持连接

5、CemuleDlg::OnBnClickedConnect()🡪StartConnect()🡪CServerConnect::ConnectToAnyServer（UINT startAt=0, bool prioSort=true, bool isAuto=true, bool bNoCrypt=false）🡪StopConnectionTry(){m\_idRetryTimer=0,for不执行}

🡪Disconnect()(if(false),return false；)

🡪TryAnotherConnectionRequest(){next\_server=first Sever in list}🡪ConnectToServer(CServer\* server=next\_server, bool multiconnect=true, bool bNoCrypt=true)

5.1🡪 CAsyncSocketEx::Create(UINT nSocketPort=0, int nSocketType=1/\*=SOCK\_STREAM\*/,

long lEvent=51 /\*=FD\_READ | FD\_WRITE | FD\_CONNECT | FD\_CLOSE\*/,

LPCSTR lpszSocketAddress=NULL /\*=NULL\*/, BOOL bReuseAddr=0 /\*=FALSE\*/ )

5.1.1🡪 CAsyncSocketEx::InitAsyncSocketExInstance(){ if(m\_spAsyncSocketExThreadDataList)==true,执行if分支,if(!pList)==false}

5.1.2同3.1.2 m\_nWindowDataPos=2

5.1.3同3.1.3 lEvent=51

5.1.4 同3.1.4 nSocketPort=0

5.2🡪 ConnectTo(CServer\* server=0x04652650, bool bNoCrypt=true)

5.2.1🡪 CEncryptedStreamSocket::SetConnectionEncryption(bool bEnabled=false, const uchar\* pTargetClientHash=NULL, bool bServerConnection=true){m\_streamCryptState=ECS\_UNKNOWN,执行最后的else块，置m\_StreamCryptState = ECS\_NONE;}

5.2.2🡪SetConnectionState(CS\_CONNECTING){if,else if均未false,直接返回};

5.2.3🡪CEMSocket::Connect(LPCSTR lpszHostAddress=0x05115568, UINT nHostPort=5041)🡪 InitProxySupport(){直接返回}🡪 CAsyncSocketEx::Connect(LPCSTR lpszHostAddress=0x05115568, UINT nHostPort=5041)🡪 connect(m\_SocketData.hSocket=1136, lpSockAddr, nSockAddrLen=16) 5.3🡪CAsyncSocketExHelperWindow::WindowsProc(message=1284>=WM\_SOCKETEX\_NOTIFY){ hSocket=wParam=1136,nEvent=16(FD\_CONNECT)}🡪CServerSocket::OnConnect(nErrorCode=0)🡪 SetConnectionState(CS\_WAITFORLOGIN)🡪 CServerConnect::ConnectionEstablished(CServerSocket\* sender=0x050c5d10)🡪InitLocalIP(){m\_nLocalIP=192.168.1.115}

🡪if(sender->GetConnectionState() == CS\_WAITFORLOGIN),SendPacket(Packet\* packet,bool delpacket= true, CServerSocket\* sender)🡪 CServerSocket::SendPacket(Packet\* packet, bool delpacket=true, bool controlpacket=true, uint32 actualPayloadSize=0, bool bForceImmediateSend=false)🡪

CEMSocket:: SendPacket(Packet\* packet, bool delpacket=true, bool controlpacket=true, uint32 actualPayloadSize=0, bool bForceImmediateSend=false){if(controlpacket){ controlpacket\_queue.AddTail(packet); // queue up for controlpacket

theApp.uploadBandwidthThrottler->QueueForSendingControlPacket(this, HasSent());},if(bForceImmediateSend)==false}



6、UploadBandwidthThrottler::RunProc(LPVOID pParam)启动🡪RunInternal()🡪CEMSocket:: SendControlData(uint32 maxNumberOfBytesToSend=9, uint32 minFragSize=536)🡪Send(uint32 maxNumberOfBytesToSend=9, uint32 minFragSize=536, bool onlyAllowedToSendControlPacket=true)

🡪CAsyncSocketExHelperWindow::WindowProc(){case FD\_WRITE}🡪CEMSocket::OnSend(0)

🡪 CEncryptedStreamSocket::OnSend(int nErrorCode=0){nothing to do}

byConnected = ES\_CONNECTED;

7、UploadBandwidthThrottler::RunProc(LPVOID pParam)启动🡪RunInternal()🡪CEMSocket:: SendControlData(uint32 maxNumberOfBytesToSend=33800, uint32 minFragSize=1300)🡪Send(uint32 maxNumberOfBytesToSend=33800, uint32 minFragSize=1300, bool onlyAllowedToSendControlPacket=true)🡪CEncryptStreamSocket::Send(const \*lpBuf=0x051fbc40,int nBuflen=86,int nFlags=0)🡪CAsyncSocketEx:Send(const \*lpBuf=0x051fbc40,int nBuflen=86,int nFlags=0)

# UploadBandwidthThrottler工作流程

一、各变量的含义

1、m\_StandardOrder\_list：标准列表，可发送控制包和标准包

2、

二、节流阀工作原理

1、统计一个循环经过的时间timeSinceLastLoop，再统计一个循环发送的字节数bytesToSpend，计算出传输速率datarate=bytesToSpend/timeSinceLastLoop，与允许传输速率比较，快了则休眠一段时间。需要注意的是，设定的速率是以秒为单位的，而循环里的计时则是以毫秒为单位的，字节数也是有以Byte为单位的，有以KB为单位的，需要换算。

假设限速300K/S，则换算成B/ms则为300\*1024/10000=307b/ms。如果前200ms传输了200K，则速率达到了1MB/S，还剩100K还需100ms，休眠时间则为1000-300=700毫秒。当然一个循环需要时间没有这么多，（调试跟踪到底需要多少时间）假设需要20ms，则能发送的字节为6K。如果上一个循环发送了7K字节，则本循环仅需要发送5K，需要时间为5/6\*20=17ms，则本次循环需要休眠的时间为3ms。而如果上一个循环发送了5K字节，则本循环需要发送7K，则本次循环不需要休眠，相反可以提高传输速率到350K/S。

假设限速300K/S，上一个循环到现在经历了20ms，则理论应该发送6K，

2、计时部分

2.1 thisLoopTick 在while()中定义const DWORD thisLoopTick = timeGetTime();代码块常变量，初始化之后没再赋值

2.2 lastLoopTick

2.2.1RunInternal局部变量，初始化DWORD lastLoopTick = timeGetTime();

2.2.2

UINT UploadBandwidthThrottler::RunInternal()

{

DWORD lastLoopTick = timeGetTime();

……

While(doRun)

{

DWORD timeSinceLastLoop = timeGetTime() - lastLoopTick;

……..

const DWORD thisLoopTick = timeGetTime();

if(allowedDataRate != \_UI32\_MAX) //在有速度限制的情况下，

{ …….

else if(\_I64\_MAX/timeSinceLastLoop > allowedDataRate && \_I64\_MAX-allowedDataRate\*timeSinceLastLoop > realBytesToSpend)

{

if(timeSinceLastLoop > sleepTime + 2000) ///snow:超过2秒

{ ……

timeSinceLastLoop = sleepTime + 2000;

lastLoopTick = thisLoopTick - timeSinceLastLoop;

}

…..…

}

………..

}

lastLoopTick = thisLoopTick;

if(bytesToSpend >= 1 || allowedDataRate == 0)

{

发送数据 ，分四批发送，首先发送控制包信息，其次发送m\_StandardOrder\_list中长时间没发送过数据的socket，第三再正常发送m\_StandardOrder\_list中各socket，最后如果带宽没用完，就再次发送m\_StandardOrder\_list中的各socket，尽量用完带宽。如果实在用不完，则允许在下一循环中多发送999字节。

}

….

}

…….

}

2.3 在循环开始之前，记下当前时刻，存入lastLoopTick，作为初始值（其实有必要吗？）；每次循环开始执行，用循环开始时间，减去lastLoopTick，为运行已过时间，存入timeSinceLastLoop。因为第一次循环时，计算timeSinceLastLoop的语句紧挨着lastLoopTick赋值语句，所以timeSinceLastLoop=0。在循环执行中，当准备发送数据 的时候，记下当前时刻，用当前时刻的值去更新lastLoopTick，这样在第二次或以后的循环时，timeSinceLastLoop的值就相当于上次循环时发送数据的时间。当需要限速时(allowedDataRate != \_UI32\_MAX)，计算spentBytes/timeSinceLastLoop就可以获取当前上传速度了。

# Emule.cpp

CemuleApp

## 构造函数流程

1. InitDEP
2. InitHeapCorruptionDetection
3. 加载gdiplus.dll，什么作用？
4. 版本检查—主程序版本，协议版本
5. EnableHtmlHelp

## Initinstance()流程

1. 加载配置文件preferences.ini
2. Install crash dump creation
3. Locale initialization
4. AfxOleInit(); //snow: ATL 初始化
5. // snow : 检查Comctl32和shell32版本
6. AfxSocketInit() //snow: Socket初始化
7. AfxEnableControlContainer();
8. thePrefs.Init();
9. theStats.Init();
10. 配置LOG文件
11. // snow : 检查防火墙设置
12. 初始化UPNP
13. Highres scheduling //干什么的？
14. 上传带宽设置
15. clientlist = new CClientList();
16. friendlist = new CFriendList();
17. searchlist = new CSearchList();
18. knownfiles = new CKnownFileList();
19. serverlist = new CServerList();
20. serverconnect = new CServerConnect();
21. sharedfiles = new CSharedFileList(serverconnect);
22. listensocket = new CListenSocket();
23. clientudp = new CClientUDPSocket();
24. clientcredits = new CClientCreditsList();
25. downloadqueue = new CDownloadQueue(); // bugfix - do this before creating the uploadqueue
26. uploadqueue = new CUploadQueue();
27. ipfilter = new CIPFilter();
28. webserver = new CWebServer(); // Webserver [kuchin]
29. mmserver = new CMMServer();
30. scheduler = new CScheduler();
31. m\_pPeerCache = new CPeerCacheFinder();
32. thePerfLog.Startup();
33. dlg.DoModal();

# DH交换密钥建立加密连接流程

## 客户端到服务器的呼出连接

1、CServerSocket::ConnectTo(CServer\* server, bool bNoCrypt)

根据bNoCrypt，客户端设置里是否要求加密连接，以及服务器是否支持加密连接

是：SetConnectionEncryption(true, NULL, true);

否：SetConnectionEncryption(false, NULL, true);

**m\_StreamCryptState为ECS\_UNKNOWN 或 ECS\_NONE**

2、CEncryptedStreamSocket::SetConnectionEncryption(bool bEnabled, const uchar\* pTargetClientHash, bool bServerConnection)

/// snow :不是到服务器的连接，目标客户端的ID哈希值不是NULL,且目的是要启动加密连接

if (bEnabled && pTargetClientHash != NULL && !bServerConnection){

**m\_StreamCryptState = ECS\_PENDING;** ///snow:是outgoing connection，状态为连接等待

///snow start：建立混淆密钥 - Client A (Outgoing connection):

/// Sendkey: Md5(<UserHashClientB 16><MagicValue34 1><RandomKeyPartClientA 4>) 21

/// Receivekey: Md5(<UserHashClientB 16><MagicValue203 1><RandomKeyPartClientA 4>) 21

m\_pRC4SendKey = RC4CreateKey(md5.GetRawHash(), 16, NULL);

m\_pRC4ReceiveKey = RC4CreateKey(md5.GetRawHash(), 16, NULL);

}

3、CServerSocket:: Connect(CStringA(server->GetAddress()), nPort))🡪CEMSocket::Connect(LPCSTR lpszHostAddress, UINT nHostPort)🡪CEncryptedStreamSocket::Connect(lpszHostAddress, nHostPort);🡪 CAsyncSocketEx::Connect(LPCSTR lpszHostAddress, UINT nHostPort)🡪 connect(m\_SocketData.hSocket, lpSockAddr, nSockAddrLen)

向服务器发出连接请求

4、

## 客户端到客户端的呼出连接

## 客户端到客户端的呼入连接