

eMULE 源码分析

Author: 刘刚 ganghust@gmail.com MSN: ganghust@hotmail.com

博客: <http://hustlg.bokee.com>

部分翻译和内容材料来源于网络，一并向原作者表示感谢。

已经查看的源代码的版本包括:

1. eMule 0.42b VeryCD0229
2. eMule(电骡) v0.45b 源码
3. eMule0.47a-Sources
4. eMule0.47b-Sources
5. eMule-0.47c-VeryCD1215-Src

从 0.42B 到 0.47 版本主要增强和修改的地方包括:

- (1) Web 管理功能的增强;
- (2) 对服务器探测时间参数的优化;
- (3) 客户端上传队列的过程进行了优化: 排队机制和规则上的修改;
- (4) 文件缓存 cache 部分改进;
- (5) KAD 网络中搜索功能的改进;

源代码版本: 0.47cVeryCD 版

eMULE 源码分析.....	1
1 目录结构:	1
2 Src\目录下代码结构.....	2
3 重要的功能子类.....	2
4TCP 和 UDP 网络通讯过程详细介绍:	6
5eMule 中信誉机制的实现.....	8
6 下载如上传任务及队列的详细说明:	8
7 其他辅助功能类的说明:	10
8 协议通讯过程的主要约定如下:	11
附录 一 ED2K 通讯报文处理细节:	11
附录二 eMule 中 KAD 网络的说明.....	15
附录: eMule 中内容发布或者搜索.....	19

1 目录结构:

模块 序号	模块大小	模块名称	备注说明
1		src	eMule 源代码主要工程
2	Debug:245K Release:105K	Zlib	数据压缩支持库, 传输过程中支持数据压缩

3	Debug:4404K Release:2171K	Id3lib	是用于读、写和操纵 ID3v1 和 ID3v2 标签的 对于媒体类型的文件，它能够调用 id3lib 库来获取诸如作者，唱片发行年代，风格等 tag 信息。如果是视频媒体文件，它还会去抓图
4	Debug:653K Release:306K	Png	提供对 PNG 文件处理的支持
5	Debug:1432K Release:517K	Resizable Lib	一个界面库,可以根据父窗口的位置和大小动态调整控件窗口的大小.
6	Debug:29284K Release:26305K	Crypto51	密码类库，实现了各种公开密钥算法、对称加密算法、数字签名算法、信息摘要算法。eMule之用主要是实现RSA签名，支持独有的积分机制

2 Src\目录下代码结构

模块序号	模块名称	备注说明
1	CxImage	图像处理库，与Windows、MFC支持极好，支持图像的多种操作（线性滤波、中值滤波、直方图操作、旋转缩放、区域选取、阈值处理、膨胀腐蚀、alpha混合等等）
2	Kademlia	KAD 网络的支持
3	Lang	本地语言化支持
4	Res	eMule 使用的资源包括图标，音乐文件等
5	Wordfilter	搜索关键词语过滤

2. 1 主连接的启动和断开：

1. 启动 `void CemuleDlg::StartConnection()` src/interface/emuleDlg.cpp 2000 行
2. 断开连接 `void CemuleDlg::CloseConnection()` src/interface/emuleDlg.cpp 2021 行
3. 连接服务器动作的发起点在 `sockets.cpp` 的 145 行
4. `EMSocket.cpp` 的 161 行开发发起连接，先检查是否设置了代理。

1. `ProcessCommandline`。Emue.cpp的331行

通过在注册表里添加一些项目可以让一个程序和某种链接或者某个后缀的文件产生关联。具体办法可以参见 `OtherFunctions.cpp` 中的 `Ask4RegFix`，`BackupReg`，`RevertReg` 三个函数的功能。

启动流程：

3 重要的功能子类

序号	类名称	备注与说明
1	CPreferences	掌握着程序的大部分配置数据，它们的特点都是有很多的成员变量，而且还是静态的，这种方式可以保证它们的唯一性，而且把这些变量统一到一个类管理。但是实际上并不需要了解每个变量的含义。thePrefs和theStats是这两个类的唯一的实例。
2	CStatistics	后者则进行各种统计，thePrefs和theStats是这两个类的唯一的实例。在CPreferences.cpp的485行Init函数中开始创建原始的配置文件目录；
3	SafeFile StringConversion CFileDataIO DataIO	数据操作的行为和数据操作的对象分割开来， 这些类要读取的数据对象通常有这些，各种整型，字符串，以及Tag类型。整型读写起来比较简单，从1个字节的，2个字节的到4个，8个或者16个字节类型的数据读写方法都比较类似。
4	CKnownFileList CKnownFile(CKnownFile把读到的文件信息都保存成一个一个的Tag。它在运行中会尽量得获取更多的文件信息，例如，对于媒体类型的文件，它能够调用id3lib库来获取诸如作者，唱片发行年代，风格等tag信息。如果是视频媒体文件，它还会去抓图(功能实现：CFrameGrabThread)。CKnownFile还能够随时掌握目前该文件的下载情况(内部有个CUpDownClient的列表)，当然，还会根据要求序列化 and 反序列化自己，LoadFromFile和WriteToFile都以CFileDataIO为参数，这样方便CKnownFileList保存和读取它的列表中的所有文件的信息)	CKnownFileList类使用了MFC的CMap类来维护内部的hash表，它内部维护了一个已知的文件的列表和取消了的文件列表。hash表的关键字都是文件hash值。能够判断出文件名不同而内容相同的文件。 CKnownFile类是一个专门关注某个特定文件的信息的类，它仍然有其基CAbstractFile。但是它和CAbstractFile类的主要区别就是CAbstractFile类只有基本的信息存取的功能，而CKnownFile能够主动的生成这些信息，
5	CKnownFile::CreateFromFile CAICHHash CAICHHashTree CAICHHashAlgo	emule中的分块处理和恢复机制,分块处理以及hash计算相关的类都在SHAHashSet.cpp和SHAHashSet.h中。下面介绍其中几个主要的类：CAICHHash类只负责一块hash值，提供两个CAICHHash类之间的直接赋值，比较等基本操作。CAICHHashAlgo是一个hash算法的通用的接口，
6		兼容CAsyncSocket类，即把应用程序中所

	<p>CAsyncSocketEx</p> <p>ThrottledControlSocket</p> <p>ThrottledFileSocket</p> <p>(任何其它的网络套接字类如果想实现限速的功能,只需要在其默认的发送函数(如 Send 或 Sendto)中不发送数据而是把数据缓存起来,然后在实现 ThrottledControlSocket 或者 ThrottledFileSocket 接口中的 SendFileAndControlData 或 SendControlData 方法时才真正把数据发送出去)</p>	<p>以的 CAsyncSocket 换成 CAsyncSocketEx, 程序仍然能够和原来的功能相同, 因此在使用上更加方便。效率更高, 主要是在消息分发机制上, 即它处理和 SOCKET 相关的消息的效率要比原始的 MFC 的 CAsyncSocket 类更高。另外, CAsyncSocketEx 类支持通过实现 CAsyncSocketExLayer 类的方式, 将一个 SOCKET 分成若干个层, 从而可以很方便得实现许多网络功能, 如设置代理, 或者是使用 SSL 进行加密等</p>
7	<p>UploadBandwidthThrottler</p> <p>保存若干Socket队列, 这些队列的处理方式略有不同。在标准队列(m_StandardOrder_list)里面排队的都是实现了ThrottledFileSocket接口的类, 通常这类能够传输文件内容也可以传输控制信息。而其它四个队列都是实现ThrottledControlSocket接口的类的队列, 主要以传输控制信息为主。这四个队列为临时高优先级, 临时普通优先级, 正式高优先级, 正式普通优先级。和把套件字直接添加到普通队列 (AddToStandardList) 不同, QueueForSendingControlPacket把要添加到队列的套接字全部添加到两个临时队列。根据它们的优先级添加到普通的临时队列。在RunInternal的大循环中, 临时队列中的项目先被移到普通队列中, 然后再进行处理。 UploadBandwidthThrottler使用了两个临界区, 两个事件。pauseEvent是用来暂停整个大循环的动作的。而threadEndedEvent是标志整个线程停止的事件。sendLocker是大循环中使用的主要的临界区, 而tempQueueLocker是为两个临时队列额外添加的锁, 这样可以一边发送已有队列中的套接字要发送的数据, 一边把新的套接字加到队列中。</p>	<p>一个 WinThread 的子类, 平时单独运行一个线程。控制全局的上传速度的。 UploadBandwidthThrottler 的 RunInternal 中的大循环是该工作线程的日常操作。这个大循环中做了以下事情, 计算本次配额, 即本次循环中能够发送多少字节, 好安排调度, 计算本次循环应该睡眠多少时间, 然后进行相应的睡眠, 从而进行限速。操作控制信息队列, 发送该队列中的数据, 注意, 控制队列中的套接字 (m_ControlQueueFirst_list 和 m_ControlQueue_list)只使用一次就离开队列。而标准队列中的套接字不会这样。在一轮循环结束后, 如果还有没有用完的发送数据的配额, 则会有部分配额保存到下一轮。</p>
8	<p>CEMSocket</p> <p>CEMSocket 是 CAsyncSocketEx 和 ThrottledFileSocket 的子类。它可以分出状态, 如当前是否在发送控制信息等。 考察它的 SendControl</p>	<p>CEMSocket 的 SendControlData 和 SendFileAndControlData 方法其实都是调用自己的另一个重载的 Send 方法。这个方法是在 UploadBandwidthThrottler 的工作线程中的大循环中被调用的, 而这个 Send 方法的内容本身也是一个大循环, 就是在不</p>

	<p>Data 和 SendFileAndControlData 方法,这些方法是用 来和 UploadBandwidthThrottler 进行配合,以便完 成全局的限速功能的。它的功能应该是按照 UploadBandwidthThrottler 的要求,在本次轮到它发 送数据时发送指定数量的字节数。</p> <p>因此,应用程序的其它部分在使用 CEMSocket 时, 如果要达到上传数据限速的目的,不应该直接调用 标准的 Send 或者 SendTo 方法,而是调用 SendPacket。这里就有了另外一个结构 Packet, 它 通常包含一个 emule 协议中完整的包,例如有协议 的头部数据等,还内置了 PackPacket 和 UnPackPacket 方法,可以自行进行压缩和解压的功 能。SendPacket 把要发送的 Packet 放到自己的队列 中,这个队列也有两个,控制信息包队列,和标准 信息包队列。如果有必要,把自己加入到 UploadBandwidthThrottler 的队列中。</p>	<p>超过自己本次发送的配额的情况下,把自 己的包队列中的包取出来,并且发出去。 用到了一个临界区,它是为了保证从包队 列中取出包来发送和把包往队列中放的操 作是互斥的。把它和 UploadBandwidthThrottler 结合起来,就看 到了两个两层的队列,即所有的套接字组 成了一个发送队列,在 UploadBandwidthThrottler 的控制下保证 了对速度的限制,而每个套接字即将发送 的数据包又组成了一个队列,保证了每次进 行数据发送的时候都会满足 UploadBandwidthThrottler 的要求。</p>
9	<p>CSearchList</p> <p>CSearchFile 是 CAbstractFile 的另一个子类 (CKnownFile 也是),它保存了某个文件和搜索相 关的信息,而不是这个文件本身的信息,就是都在哪 些机器上有这个文件,以及哪个服务器上搜到的这 个文件。甚至还可以向搜索文件添加预览。在这个 类的定义中嵌套定义了两个简单的结构 SServer 和 SClient,表示了该搜索文件的可能来源,服务器或 者其它客户端。m_aClients 和 m_aServers 是这两个 简单结构的一个数组,CSearchFile 自然也提供了对 这个数组的操作的接口,方便 CSearchList 使用。 CSearchList 对外提供了搜索表达的接口,即每当有 一个新的搜索提交时 CSearchList::NewSearch 会建 立一个新的搜索项,但是此时还没有任何对应的搜 索文件,因此只是在文件个数和搜索 ID 的对应表 (m_foundFilesCount 和 m_foundSourcesCount)中 建立新的项目。</p>	<p>CSearchList 是 emule 中的搜索列表,掌管 emule 中所有的搜索请求。CSearchFile 是 这个列表中的元素,代表了一次搜索的相 关信息。它们的关系和之前描述的已知文 件和已知文件列表有一些类似的地方。 CSearchList 的主要任务就是对其一个叫做 list 的类型为 CSearchFile 列表的内部变量 进行维护,提供很方便得往这个列表中添加, 删除,查询,变更等操作的接口。另外, 每一个搜索都有一个 ID,是一个 32 位的整数。CSearchList 中记录了每个搜索 目前搜到的文件个数和源的个数 (m_foundFilesCountm_foundSourcesCount) 当有搜索结果返回时 ProcessSearchAnswer 或 ProcessUDPSearchAnswer 能够对返回的 包直接做处理,创建相应的搜索文件信息 CSearchFile 对象,并加入到自己的列表中。 当然,要把重复的搜索结果去除,发现同 一个 hash 的文件的多个源时也会给它们建 立一个二级列(CSearchFile::m_list_parent)。 CSearchList 只负责和搜索有关的信息的储 存和读取,本身并不进行搜索。</p>
10	<p>CServerList</p> <p>CServerList 除了提供通常的 CServer 信息外,还 提供一些统计信息诸如所有的服务器的用户数,共 享的文件数等。这些统计信息也是基于每个单独 的 CServer 的相关信息计算出来的。</p>	<p>CServerList 是 emule 中负责管理服务器列 表的类。CServerList 需要对外提供列表的 增加,删除,查找,修改等接口。在 CServerList 中,每个服务器的信息是一个 CServer 类。和搜索信息不一样,但是和已 知文件列表一样,服务器的信息列表是需</p>

		<p>要长期保留的，因此 CServerList 和 CKnownFileList 类一样提供了把它所包含的所有信息保存到一个文件中，以及从这个文件中读回其信息的功能。CServer 中的结构比较简单，只需要保留服务器的各种信息即可。它可以通过 IP 地址和端口来创建，也可以通过一个简单的结构 ServerMet_Struct 来创建，其中后者是用来直接从文件中读取的。该结构仅仅包含 IP 地址和端口以及属性的个数，CServer 中其它的属性在保存到文件中时，均采用 Tag 方式保存。</p>
11	<p>Packet</p> <p>它内部实现了压缩和解压的方法，该方法直接调用 zlib 库中的压缩方法，可以减少数据的传输量。这里要注意一点的就是压缩的时候协议簇代码是不参与压缩的，压缩完毕后会更换协议簇代码，例如代码为标准 edonkey 协议 0xE3 的包在压缩后，协议代码就变成 0xD4 了，这里进行协议代码变化是为了使接受方能够正确识别并且进行相应的解压操作。</p>	<p>它是 emule 的通信协议的最小单位。我们可以看出，它的构造函数有多个版本，这也是为了可以用不同的方式来创建 Packet。例如只包含一个头部信息的缓冲区，或者只是指定协议簇代码等。</p>

4TCP 和 UDP 网络通讯过程详细介绍：

通讯的双方及方式	完成此功能的主要类	说明
Client—Server TCP	<p>CServerConnect</p> <p>（ 通 过 分 析 CServerSocket::ProcessPacket 就可以直接把emule客户端和服务端之间的通信协议理解清楚，这里是服务器发回的包。TCP连接建立后的第一个包 是在 CServerConnect::ConnectionEstablished中发出的，即向服务器发出登陆信息。如果登陆成功，则能够从服务器处获取自己的ID，这是一个32位的长</p>	<p>CServerConnect 本身不是套接字的子类，但是它的成员变量 CServerSocket 类型的 connectedsocket 是 。CServerConnect 内部有一列表，可以保存若干 CServerSocket类型的指针。但是这并不说明它平时连接到很多服务器上。它只是可以同时试图连接到若干个服务器上，这只是因为连接到服务器上的行为不一定能成功。CServerSocket类是 CEMSock</p>

	<p>整数。如果这个数小于16777216，那么我们称它为LowID。具有LowID的客户端通常情况下其它客户端将不能直接连接它。得到LowID的原因比较多，例如当自己处于NAT的后端的时候。获取自己的ID后将会向服务器发送自己的共享文件列表，这一动作由共享文件列表类CSharedFileList来完成。)</p>	<p>et的子类，它比CEMSocket要多保存一些状态，比如当前的服务器连接状态。它同时还保留它当前所连接的服务器的信息。</p>
Client-Client TCP	<p>CListenSocket CClientReqSocket</p> <p>(CListenSocket 和 CClientReqSocket类之间的关系和前面分析的列表类和它对应的成员类的关系是相似的，CListenSocket提供对自身的CClientReqSocket列表中的元素的增加，查询，删除等操作。同时也维护关于这些成员的一些统计信息。我们注意到CListenSocket在其构造函数中就把自己添加到CListenSocket类(theApp.listensocket，该类的唯一实际示例)的列表中。</p> <p>CClientReqSocket类和CUpDownClient类之间存在着对应关系。它们都表示了另外一个客户端的一些信息，但是CClientReqSocket类主要侧重在网络数据方面，即负责两边的互相通信，而CUpDownClient类负责的是从逻辑上对网络另一边的一个客户端进行表达。)</p>	<p>由 CListenSocket 和 CClientReqSocket完成。这也是提供网络服务的应用程序的典型写法。其中 CListenSocket 只是 CAsyncSocketEx的子类，只负责监听某个TCP端口。它只是内部有一个 CClientReqSocket类的列表。而 CClientReqSocket 是 CEMSocket的子类，因此它能够自动完成emule的packet识别工作。它有ProcessPacket和ProcessExtPacket来处理客户端和客户端之间的包，其中前者是经典的eDonkey协议的包，后者是emule扩展协议的包。</p>
Client—Server UDP	CUDPSocket	<p>走UDP协议的包，因为UDP本来就是以一个包一个包作为单位在网络上流传的，不需要在包的内容中再包含表示长度的字段。每个UDP包的第</p>
Client-Client UDP	<p>CClientUDPSocket Kademlia::CKademliaUDPListener</p>	

		一个字节是协议簇代码,其它内容就是包的内容。 CKademliaUDPListener类,专门处理和Kademlia协议相关的UDP包。
--	--	---

5eMule 中信誉机制的实现

	主要实现的类	说明
1	CClientCreditsList	和信誉相关的信息是需要永久保存的,这样才有意义,因此 CClientCreditsList 提供了 LoadList 和 SaveList 方法。
2	CClientCredits	CClientCredits 类可以使用 CreditStruct 结构来创建,而 CreditStruct 结构只包含静态信息,主要是上传量和下载量等。

信誉机制的信息需要有一定的可靠性,在emule中采用了数字签名的方式来做这一点。Crypto++库为emule全程提供和数字签名验证相关的功能。CClientCreditsList在创建时,会装载自己的公钥私钥,如果没有的话,会创建一对。CClientCreditsList中包含的有效的信息都是经过其它人数字签名的,所以更加有信服力。在实际使用中,这些信息和自己的私钥要注意保存。重装emule后应该把配置文件目录先备份,这样能够保留自己辛辛苦苦积攒的信誉。

6 下载如上传任务及队列的详细说明:

功能说明	类	说明
部分文件表示	CPartFile (CPartFile 它的创建就有几种可能,从搜索文件 CSearchFile 中创建,这种情况发生在用户搜索到他想要的文件后点击下载时发生。从一个包含了 ed2k 链接的字符串中创建,它会提取出该 ed2k 链接中的信息,并用来创建 CPartFile。剩下的一种,就是当 emule 程序重启后,恢复以前的下载任务。这时就是去	CPartFile 类是 emule 中用来表示一个下载任务的类。这就是一个还没有完成的文件。当一个下载任务被创建时,emule 会在下载目录中创建两个文件,以三位数字加后缀 part 的文件,例如 001.part, 002.part 等。还有一个以同样的数字加上.part.met 的文件,表示的是对应文件的元信息。part 文件会创建得和原始文

	<p>下载目录中寻找那些.part 和.met 文件了。另外它还需要不断得处理下载到的数据，为了减少磁盘开销，使用了 Requested_Block_Struct 结构来暂存写入的数据。它内部维护一个 CUpDownClient 的列表，如果知道了该文件的一个新的来源信息，就会创建一个对应的 CUpDownClient。后者是 emule 中代码量最大的类。它还要把它的状态用彩色的条装物显示出来提供给 GUI。)</p>	<p>件大小一样，当下载完成后，文件名会修改成它本来的名称。而事实上，诸如这个文件原来叫什么名称，修改日期等等信息都在对应的 .part.met 元文件中。 .part.met 中还包含了该文件中那些部分已经下载完成的信息。CPartFile 类中 Gap_Struct 来表示文件的下载情况，一个 Gap_Struct 就是一个坑，它表示该文件从多少字节的偏移多少字节偏移是一个坑。下载的过程就是一个不断填坑的过程。CPartFile 类中有个成员变量 gaplist 就是该文件目前的坑的状况列表。需要主要的是有时填了坑的中间部分后，会把一个坑变成两个坑。坑的列表也会被存进 .part.met 中。</p>
下载	<p>CDownloadQueue (CDownloadQueue 中的 Process 方法的主要任务就是把它的列表中的 CPartFile 类中的 Process 方法都调一遍，另外主要的一些关于下载情况的统计信息也是在每一轮的 Process 后进行更新的。从这里我们也可以看出 Process 方法在 emule 中的意义，就是一个需要经常执行的方法，通过经常执行它们来完成日常工作，而且所有的这些 Process 方法肯定是顺序执行，因此可以减少很多多线程的同步之类的问题。emule 中已经尽量减少了多线程的使用，但是在很多地方如果多线程是不可避免的话，也不会排斥。)</p>	<p>CDownloadQueue 是下载队列类。这个队列中的项目是 CPartFile 指针。它能够提供一个对列表中的元素进行增加，查询，删除的功能。例如查询的时候能够根据该文件的 hashID 或者索引来进行查询。CDownloadQueue 同时还要完成一些统计工作。和其它的列表类不一样的是，它的所有元素的信息并不是集中存放于一个文件，而是对应于每一个下载任务，单独得存放在一个元信息文件(.part.met)中，因此当该类进行初始化的时候，它需要寻找所有可能的下载路径，从那些路径中找到所有的 .part.met 文件，并且试图用这些文件来生成 CPartFile 类，并且将这些通过 .part.met 文件正确生成的 CPartFile 类添加到自己的列表中，同样，在退出时，所有的下载任务的元信息也是自行保存，不会合成为一个文件。</p>
上传	<p>CUploadQueue (CUploadQueue 的 Process 方法就相对简单了，那就是向上传队列中的所有客户端依次发送数据，而排队的客户端是不会得到这个机会的。另外它还需要完成关于上传方面的一些统计信息。在 CUploadQueue 的构造函数里面，创建了一个以 100 毫秒为间隔的定时器，这个定时器成为以上所有的 Process 所需要的基础。)</p>	<p>CUploadQueue 是上传队列类。这个列表类中只有以 CUpDownClient 为元素的列表，它和其它列表类还有一个很大的不同就是它所保存的信息都不需要持久化，即不需要在当前的 emule 退出后还记住自己正在给谁上传文件，然后下次上线的时候再继续给他们传，这在大部分情况下是没有意义的。他有两个列表，上传列表和排队列表。当收到一个新的下载请求后，它会把对应的客户端先添</p>

		加到排队列表中，以后再根据情况，把它们不断添加到上传列表中。在这里，信誉机制将会对此产生影响。
上传和下载的辅助类	<p>CUpDownClient</p> <p>BaseClient.cpp 中实现的是该类的一些基本的功能，包括基本的各种状态信息的获取和设置，以及按照要求处理和发送各种请求。在这里，逻辑实现和网络进行了区分，CUpDownClient 类本身不从网络接受或者发送消息，它只是提供各种请求的处理接口，以及在发送请求时，构造好相应的 Packet，并交给自己对应的网络套接字发出去。</p> <p>DownloadClient.cpp 中实现的是和下载相关的功能，它包括了各种下载请求的发送以及相应的数据的接收。另外还有一个 A4AF 的机制，它是 emule 中的一个机制，因为一个客户端在同一个时间内只能向另外一个客户端请求同一个文件。这样，对于很多个下载任务(CPartFile)，有可能出现它们的源(即有该文件的客户端)有部分重叠的现象，而这时，如果其它下载任务正在从这个源下载，那么当前的下载任务就不能从这个源下载了。但是 emule 允许用户对其手动进行控制，如对下载任务的优先级进行区分，这样他就可以将一个源从另外一个下载任务那里切换过来。A4AF 其实就是 ask for another file 的简称。</p> <p>UploadClient.cpp 中实现的是上传相关功能，即接受进来的下载请求，并且生成相应的文件块发送出去。</p>	<p>CUpDownClient 类的作用是从逻辑上表示一个其它的客户端的各种信息，它是 emule 中代码量最大的类。我们注意到，定义它的头文件是 UpDownClient.h，但是却没有对应的 CUpDownClient.cpp，而它的实现，都分散到 BaseClient.cpp, DownloadClient.cpp, PeerCacheClient.cpp, UploadClient.cpp 和 URLClient.cpp 中。</p> <p>PeerCacheClient.cpp 实现的是和 PeerCache 相关的功能，PeerCache 是一个由 Joltid 公司开发的技术，它可以允许你从 ISP 提供的一些快照服务器上快速得上传或者下载一些文件(或者是一部分)，这个技术的好处是可以减少骨干网络的带宽消耗，将部分本来需要在骨干网上走的流量转移到 ISP 的内部。当然这个功能需要 ISP 的配合。如果发现 ISP 提供了这项服务的话，emule 会利用它来减少骨干网的带宽消耗。</p> <p>URLClient.cpp 实现的功能是利用 http 协议对原有的 emule 协议进行包装，以便使它能够尽可能地穿越更多的网络的防火墙。</p>

7 其他辅助功能类的说明：

在 emule 中使用到的其它类及其功能。我们可以看到，如果单纯只是为了能够搜到以及下载到文件的话，有不少类是可以精简的，但是，正是由于它们的存在，使得 emule 的功能更加的完善。**CIPFilter**，IP 地址过滤器，通过识别各种类型的 IP 地址过滤信息，它能够把不希望连接的网络地址过滤掉，emule 中所有需要连接网络的地方使用的都是统一的过滤数据。**CWebServer** 能够在本地打开一个 Web 服务器，然后你可以通过浏览器来控制你的 emule。**CScheduler** 能够实现下载任务的定时下载。**CPeerCacheFinder** 为前面提到的 PeerCache 技术的主控制类。另外，emule 还内置了一个 IRC 客户端，一个主要成员函数都为静态的