

原 RTMPdump (libRTMP) 源代码分析 5：建立一个流媒体连接 (NetConnection部分)

2013年10月23日 00:15:50 阅读数：14619

RTMPdump(libRTMP) 源代码分析系列文章：

[RTMPdump 源代码分析 1：main\(\)函数](#)

[RTMPDump \(libRTMP\) 源代码分析2：解析RTMP地址——RTMP_ParseURL\(\)](#)

[RTMPdump \(libRTMP\) 源代码分析3：AMF编码](#)

[RTMPdump \(libRTMP\) 源代码分析4：连接第一步——握手 \(HandShake\)](#)

[RTMPdump \(libRTMP\) 源代码分析5：建立一个流媒体连接 \(NetConnection部分\)](#)

[RTMPdump \(libRTMP\) 源代码分析6：建立一个流媒体连接 \(NetStream部分 1\)](#)

[RTMPdump \(libRTMP\) 源代码分析7：建立一个流媒体连接 \(NetStream部分 2\)](#)

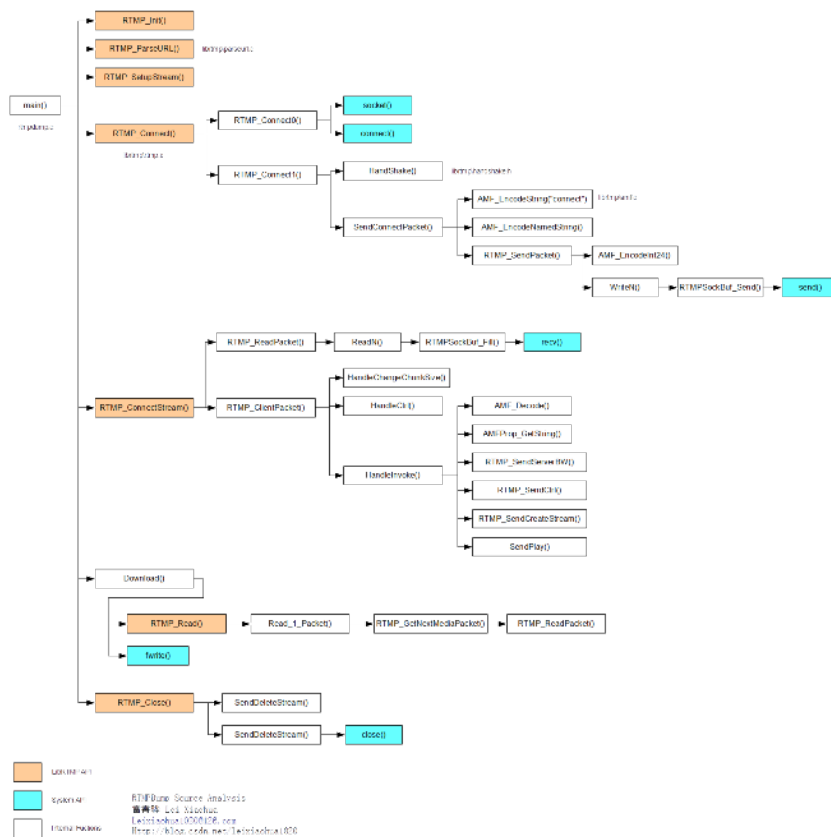
[RTMPdump \(libRTMP\) 源代码分析8：发送消息 \(Message\)](#)

[RTMPdump \(libRTMP\) 源代码分析9：接收消息 \(Message\) \(接收视音频数据\)](#)

[RTMPdump \(libRTMP\) 源代码分析10：处理各种消息 \(Message\)](#)

函数调用结构图

RTMPDump (libRTMP)的整体的函数调用结构图如下图所示。



[单击查看大图](#)

详细分析

本篇文章分析一下RTMPdump里面的建立一个流媒体连接过程中的函数调用。

之前已经简单分析过流媒体链接的建立过程：

[RTMP流媒体播放过程](#)

而且分析过其函数调用过程：

[RTMPDump源代码分析 0：主要函数调用分析](#)

在这里就不详细叙述了，其实主要是这两个函数：

RTMP_Connect()

RTMP_ConnectStream()

第一个函数用于建立RTMP中的NetConnection，第二个函数用于建立RTMP中的NetStream。一般是先调用第一个函数，然后调用第二个函数。

下面先来看看RTMP_Connect()：

注意：贴上去的源代码是修改过的RTMPdump，我添加了输出信息的代码，形如：r->dlg->AppendCInfo("建立连接：第0次连接。开始建立Socket连接");改代码不影响程序运行，可忽略。

RTMP_Connect()

```
[cpp]
1. //连接
2. int
3. RTMP_Connect(RTMP *r, RTMPPacket *cp)
4. {
5.     //Socket结构体
6.     struct sockaddr_in service;
7.     if (!r->Link.hostname.av_len)
8.         return FALSE;
9.
10.    memset(&service, 0, sizeof(struct sockaddr_in));
11.    service.sin_family = AF_INET;
12.
13.    if (r->Link.socksport)
14.    {
15.        //加入地址信息
16.        /* 使用SOCKS连接 */
17.        if (!add_addr_info(&service, &r->Link.sockshost, r->Link.socksport))
18.            return FALSE;
19.    }
20.    else
21.    {
22.        /* 直接连接 */
23.        if (!add_addr_info(&service, &r->Link.hostname, r->Link.port))
24.            return FALSE;
25.    }
26.    //-----
27.    r->dlg->AppendCInfo("建立连接：第0次连接。开始建立Socket连接");
28.    //-----
29.    if (!RTMP_Connect0(r, (struct sockaddr *)&service)){
30.        r->dlg->AppendCInfo("建立连接：第0次连接。建立Socket连接失败");
31.        return FALSE;
32.    }
33.    //-----
34.    r->dlg->AppendCInfo("建立连接：第0次连接。建立Socket连接成功");
35.    //-----
36.    r->m_bSendCounter = TRUE;
37.
38.    return RTMP_Connect1(r, cp);
39. }
```

我们可以看出调用了两个函数RTMP_Connect0()以及RTMP_Connect1()。按照按先后顺序看看吧：

RTMP_Connect0()

```
[cpp]
1. //sockaddr是Linux网络编程的地址结构体一种，其定义如下：
2. //struct sockaddr{
3. //     unsigned short sa_family;
4. //     char sa_data[14];
5. //};
6. //说明：sa_family：是地址家族，也称作，协议族，一般都是“AF_xxx”的形式。通常大多用的是都是AF_INET。
7. //     sa_data：是14字节协议地址。
8. //有时不使用sockaddr，而使用sockaddr_in（多用在windows）（等价）
9. //struct sockaddr_in {
10. //     short int sin_family;           /* Address family */
```

```

11. // unsigned short int sin_port;          /* Port number */
12. // struct in_addr sin_addr;             /* Internet address */
13. // unsigned char sin_zero[8];          /* Same size as struct sockaddr */
14. //};
15. //union {
16. //    struct{
17. //        unsigned char s_b1,s_b2,s_b3,s_b4;
18. //        } S_un_b;
19. //    struct {
20. //        unsigned short s_w1,s_w2;
21. //        } S_un_w;
22. //    unsigned long S_addr;
23. //    } S_un;
24. //} in_addr;
25. //第0次连接,建立Socket连接
26. int
27. RTMP_Connect0(RTMP *r, struct sockaddr * service)
28. {
29.     int on = 1;
30.     r->m_sb.sb_timedout = FALSE;
31.     r->m_pausing = 0;
32.     r->m_fDuration = 0.0;
33.     //创建一个Socket, 并把Socket序号赋值给相应变量
34.     //-----
35.     r->dlg->AppendCInfo("建立连接：第0次连接。create一个Socket");
36.     //-----
37.     r->m_sb.sb_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
38.     if (r->m_sb.sb_socket != -1)
39.     {
40.
41.         //定义函数 int connect (int sockfd,struct sockaddr * serv_addr,int addrlen);
42.         //函数说明 connect()用来将参数sockfd 的Socket (刚刚创建) 连至参数serv_addr
43.         //指定的网络地址。参数addrlen为sockaddr的结构长度。
44.         //连接
45.         RTMP_LogPrintf("建立Socket连接!\n");
46.         //-----
47.         r->dlg->AppendCInfo("建立连接：第0次连接。connect该Socket");
48.         //-----
49.         if (connect(r->m_sb.sb_socket, service, sizeof(struct sockaddr)) < 0)
50.         {
51.             //-----
52.             r->dlg->AppendCInfo("建立连接：第0次连接。connect该Socket失败");
53.             //-----
54.             int err = GetSockError();
55.             RTMP_Log(RTMP_LOGERROR, "%s, failed to connect socket. %d (%s)",
56.                 __FUNCTION__, err, strerror(err));
57.             RTMP_Close(r);
58.             return FALSE;
59.         }
60.         //-----
61.         r->dlg->AppendCInfo("建立连接：第0次连接。connect该Socket成功");
62.         //-----
63.         //指定了端口号。注：这不是必需的。
64.         if (r->Link.socksport)
65.         {
66.             RTMP_Log(RTMP_LOGDEBUG, "%s ... SOCKS negotiation", __FUNCTION__);
67.             //谈判？发送数据报以进行谈判？！
68.             if (!SocksNegotiate(r))
69.             {
70.                 RTMP_Log(RTMP_LOGERROR, "%s, SOCKS negotiation failed.", __FUNCTION__);
71.                 RTMP_Close(r);
72.                 return FALSE;
73.             }
74.         }
75.     }
76.     else
77.     {
78.         RTMP_Log(RTMP_LOGERROR, "%s, failed to create socket. Error: %d", __FUNCTION__,
79.             GetSockError());
80.         return FALSE;
81.     }
82.
83.     /* set timeout */
84.     //超时
85.     {
86.         SET_RCVTIMEO(tv, r->Link.timeout);
87.         if (setsockopt
88.             (r->m_sb.sb_socket, SOL_SOCKET, SO_RCVTIMEO, (char *)&tv, sizeof(tv)))
89.         {
90.             RTMP_Log(RTMP_LOGERROR, "%s, Setting socket timeout to %ds failed!",
91.                 __FUNCTION__, r->Link.timeout);
92.         }
93.     }
94.
95.     setsockopt(r->m_sb.sb_socket, IPPROTO_TCP, TCP_NODELAY, (char *) &on, sizeof(on));
96.
97.     return TRUE;
98. }

```

可见RTMP_Connect0()主要用于建立Socket连接，并未开始真正的建立RTMP连接。

再来看看RTMP_Connect1()，这是真正建立RTMP连接的函数：

RTMP_Connect1()

```
[cpp]
1. //第1次连接，从握手开始
2. int
3. RTMP_Connect1(RTMP *r, RTMPPacket *cp)
4. {
5.     if (r->Link.protocol & RTMP_FEATURE_SSL)
6.     {
7.         #if defined(CRYPTO) && !defined(NO_SSL)
8.             TLS_client(RTMP_TLS_ctx, r->m_sb.sb_ssl);
9.             TLS_setfd((SSL *)r->m_sb.sb_ssl, r->m_sb.sb_socket);
10.            if (TLS_connect((SSL *)r->m_sb.sb_ssl) < 0)
11.            {
12.                RTMP_Log(RTMP_LOGERROR, "%s, TLS_Connect failed", __FUNCTION__);
13.                RTMP_Close(r);
14.                return FALSE;
15.            }
16.        #else
17.            RTMP_Log(RTMP_LOGERROR, "%s, no SSL/TLS support", __FUNCTION__);
18.            RTMP_Close(r);
19.            return FALSE;
20.        #endif
21.    }
22.    //使用HTTP
23.    if (r->Link.protocol & RTMP_FEATURE_HTTP)
24.    {
25.        {
26.            r->m_msgCounter = 1;
27.            r->m_clientID.av_val = NULL;
28.            r->m_clientID.av_len = 0;
29.            HTTP_Post(r, RTMPT_OPEN, "", 1);
30.            HTTP_read(r, 1);
31.            r->m_msgCounter = 0;
32.        }
33.        RTMP_Log(RTMP_LOGDEBUG, "%s, ... connected, handshaking", __FUNCTION__);
34.        //握手-----
35.        r->dlg->AppendCInfo("建立连接：第1次连接。开始握手(HandShake)");
36.        //-----
37.        RTMP_LogPrintf("开始握手 (HandShake) !\n");
38.        if (!HandShake(r, TRUE))
39.        {
40.            //-----
41.            r->dlg->AppendCInfo("建立连接：第1次连接。握手(HandShake)失败！");
42.            //-----
43.            RTMP_Log(RTMP_LOGERROR, "%s, handshake failed.", __FUNCTION__);
44.            RTMP_Close(r);
45.            return FALSE;
46.        }
47.        //-----
48.        r->dlg->AppendCInfo("建立连接：第1次连接。握手(HandShake)成功");
49.        //-----
50.        RTMP_LogPrintf("握手 (HandShake) 完毕！\n");
51.        RTMP_Log(RTMP_LOGDEBUG, "%s, handshaked", __FUNCTION__);
52.        //发送“connect”命令-----
53.        //-----
54.        r->dlg->AppendCInfo("建立连接：第1次连接。开始建立网络连接(NetConnection)");
55.        //-----
56.        RTMP_LogPrintf("开始建立网络连接 (NetConnection) !\n");
57.        //-----
58.        r->dlg->AppendCInfo("发送数据。消息 命令 (typeID=20) (Connect)。");
59.        //-----
60.        if (!SendConnectPacket(r, cp))
61.        {
62.            //-----
63.            r->dlg->AppendCInfo("建立连接：第1次连接。建立网络连接(NetConnection)失败！");
64.            //-----
65.            RTMP_Log(RTMP_LOGERROR, "%s, RTMP connect failed.", __FUNCTION__);
66.            RTMP_Close(r);
67.            return FALSE;
68.        }
69.        //-----
70.        r->dlg->AppendCInfo("建立连接：第1次连接。建立网络连接(NetConnection)成功");
71.        //-----
72.        RTMP_LogPrintf("命令消息“Connect”发送完毕！\n");
73.        return TRUE;
74.    }
```

该函数做了以下事情：

HandShake()完成握手，之前已经分析过：[RTMPdump 源代码分析 4：连接第一步——握手 \(Hand Shake\)](#)

SendConnectPacket()发送包含“connect”命令的数据报，用于开始建立RTMP连接。具体该函数是怎么调用的，以后有机会再进行分析。

至此RTMP_Connect()分析完毕。

rtmpdump源代码（Linux）：<http://download.csdn.net/detail/leixiaohua1020/6376561>

rtmpdump源代码（VC 2005 工程）：<http://download.csdn.net/detail/leixiaohua1020/6563163>

版权声明：本文为博主原创文章，未经博主允许不得转载。<https://blog.csdn.net/leixiaohua1020/article/details/12957291>

文章标签：[RTMPdump](#) [rtmp](#) [源代码](#) [连接](#)

个人分类：[libRTMP](#)

所属专栏：[开源多媒体项目源代码分析](#)

此PDF由[spygg](#)生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com