# 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)

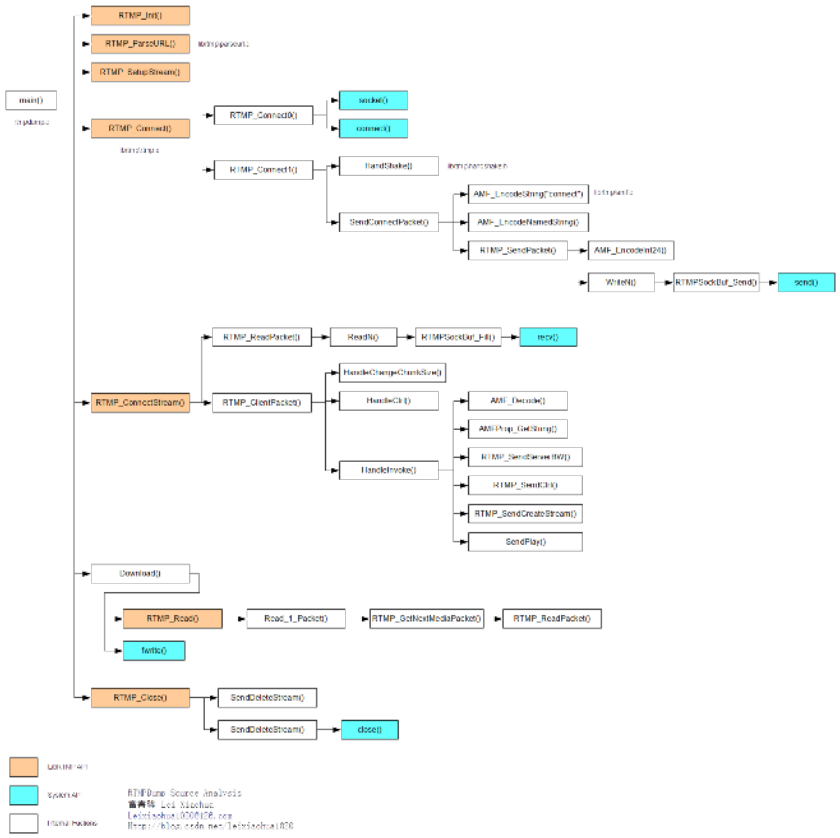
RTMPdump (libRTMP) 源代码分析9： 接收消息 (Message) (接收视音频数据)

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
//连接
int
RTMP_Connect(RTMP *r, RTMPPacket *cp)
{
    //Socket结构体
  struct sockaddr_in service;
  if (!r->Link.hostname.av_len)
    return FALSE;

  memset(&service, 0, sizeof(struct sockaddr_in));
  service.sin_family = AF_INET;

  if (r->Link.socksport)
    {
    //加入地址信息
      /* 使用SOCKS连接 */
      if (!add_addr_info(&service, &r->Link.sockshost, r->Link.socksport))
    return FALSE;
    }
  else
    {
      /* 直接连接 */
      if (!add_addr_info(&service, &r->Link.hostname, r->Link.port))
    return FALSE;
    }
  //-----------------
  r->dlg->AppendCInfo("建立连接：第0次连接。开始建立Socket连接");
  //----------------------------
  if (!RTMP_Connect0(r, (struct sockaddr *)&service)){
    r->dlg->AppendCInfo("建立连接：第0次连接。建立Socket连接失败");
    return FALSE;
  }
  //-----------------
  r->dlg->AppendCInfo("建立连接：第0次连接。建立Socket连接成功");
  //----------------------------
  r->m_bSendCounter = TRUE;

  return RTMP_Connect1(r, cp);
}
```

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
//sockaddr是Linux网络编程的地址结构体一种，其定义如下：
//struct sockaddr{
//  unsigned short sa_family;
//  char sa_data[14];
//};
//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。
//  sa_data：是14字节协议地址。
//有时不使用sockaddr，而使用sockaddr_in（多用在windows）（等价）
//struct sockaddr_in {
//  short int sin_family;              /* Address family */
```

```
//  unsigned short int sin_port;       /* Port number */
//  struct in_addr sin_addr;           /* Internet address */
//  unsigned char sin_zero[8];         /* Same size as struct sockaddr */
//};
//union {
//   struct{
//      unsigned char s_b1,s_b2,s_b3,s_b4;
//         } S_un_b;
//    struct {
//      unsigned short s_w1,s_w2;
//         } S_un_w;
//    unsigned long S_addr;
//   } S_un;
//} in_addr;
//第0次连接,建立Socket连接
int
RTMP_Connect0(RTMP *r, struct sockaddr * service)
{
  int on = 1;
  r->m_sb.sb_timedout = FALSE;
  r->m_pausing = 0;
  r->m_fDuration = 0.0;
  //创建一个Socket，并把Socket序号赋值给相应变量
  //-----------------
  r->dlg->AppendCInfo("建立连接：第0次连接。create一个Socket");
  //----------------------------
  r->m_sb.sb_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (r->m_sb.sb_socket != -1)
    {

    //定义函数  int connect (int sockfd,struct sockaddr * serv_addr,int addrlen);
    //函数说明  connect()用来将参数sockfd 的Socket（刚刚创建）连至参数serv_addr
    //指定的网络地址。参数addrlen为sockaddr的结构长度。
    //连接
    RTMP_LogPrintf("建立Socket连接！\n");
    //-----------------
    r->dlg->AppendCInfo("建立连接：第0次连接。connect该Socket");
    //----------------------------
      if (connect(r->m_sb.sb_socket, service, sizeof(struct sockaddr)) < 0)
    {
        //-----------------
        r->dlg->AppendCInfo("建立连接：第0次连接。connect该Socket失败");
        //----------------------------
      int err = GetSockError();
      RTMP_Log(RTMP_LOGERROR, "%s, failed to connect socket. %d (%s)",
          __FUNCTION__, err, strerror(err));
      RTMP_Close(r);
      return FALSE;
    }
      //-----------------
      r->dlg->AppendCInfo("建立连接：第0次连接。connect该Socket成功");
      //----------------------------
      //指定了端口号。注：这不是必需的。
      if (r->Link.socksport)
    {
      RTMP_Log(RTMP_LOGDEBUG, "%s ... SOCKS negotiation", __FUNCTION__);
      //谈判？发送数据报以进行谈判？！
      if (!SocksNegotiate(r))
        {
          RTMP_Log(RTMP_LOGERROR, "%s, SOCKS negotiation failed.", __FUNCTION__);
          RTMP_Close(r);
          return FALSE;
        }
    }
    }
  else
    {
      RTMP_Log(RTMP_LOGERROR, "%s, failed to create socket. Error: %d", __FUNCTION__,
      GetSockError());
      return FALSE;
    }

  /* set timeout */
  //超时
  {
    SET_RCVTIMEO(tv, r->Link.timeout);
    if (setsockopt
        (r->m_sb.sb_socket, SOL_SOCKET, SO_RCVTIMEO, (char *)&tv, sizeof(tv)))
      {
        RTMP_Log(RTMP_LOGERROR, "%s, Setting socket timeout to %ds failed!",
        __FUNCTION__, r->Link.timeout);
      }
  }

  setsockopt(r->m_sb.sb_socket, IPPROTO_TCP, TCP_NODELAY, (char *) &on, sizeof(on));

  return TRUE;
}
```

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
//第1次连接，从握手开始
int
RTMP_Connect1(RTMP *r, RTMPPacket *cp)
{
  if (r->Link.protocol & RTMP_FEATURE_SSL)
    {
#if defined(CRYPTO) && !defined(NO_SSL)
      TLS_client(RTMP_TLS_ctx, r->m_sb.sb_ssl);
      TLS_setfd((SSL *)r->m_sb.sb_ssl, r->m_sb.sb_socket);
      if (TLS_connect((SSL *)r->m_sb.sb_ssl) < 0)
    {
      RTMP_Log(RTMP_LOGERROR, "%s, TLS_Connect failed", __FUNCTION__);
      RTMP_Close(r);
      return FALSE;
    }
#else
      RTMP_Log(RTMP_LOGERROR, "%s, no SSL/TLS support", __FUNCTION__);
      RTMP_Close(r);
      return FALSE;

#endif
    }
  //使用HTTP
  if (r->Link.protocol & RTMP_FEATURE_HTTP)
    {
      r->m_msgCounter = 1;
      r->m_clientID.av_val = NULL;
      r->m_clientID.av_len = 0;
      HTTP_Post(r, RTMPT_OPEN, "", 1);
      HTTP_read(r, 1);
      r->m_msgCounter = 0;
    }
  RTMP_Log(RTMP_LOGDEBUG, "%s, ... connected, handshaking", __FUNCTION__);
  //握手----------
  r->dlg->AppendCInfo("建立连接：第1次连接。开始握手(HandShake)");
  //----------------------------
  RTMP_LogPrintf("开始握手 (HandShake) ！\n");
  if (!HandShake(r, TRUE))
    {
        //----------
        r->dlg->AppendCInfo("建立连接：第1次连接。握手(HandShake)失败！");
        //----------------------------
      RTMP_Log(RTMP_LOGERROR, "%s, handshake failed.", __FUNCTION__);
      RTMP_Close(r);
      return FALSE;
    }
  //----------
  r->dlg->AppendCInfo("建立连接：第1次连接。握手(HandShake)成功");
  //----------------------------
  RTMP_LogPrintf("握手 (HandShake) 完毕！\n");
  RTMP_Log(RTMP_LOGDEBUG, "%s, handshaked", __FUNCTION__);
  //发送“connect”命令----------
  //----------
  r->dlg->AppendCInfo("建立连接：第1次连接。开始建立网络连接(NetConnection)");
  //----------------------------
  RTMP_LogPrintf("开始建立网络连接 (NetConnection) ！\n");
  //----------
  r->dlg->AppendCInfo("发送数据。消息 命令 (typeID=20) (Connect)。");
  //----------------------------
  if (!SendConnectPacket(r, cp))
    {
        //----------
        r->dlg->AppendCInfo("建立连接：第1次连接。建立网络连接(NetConnection)失败！");
        //----------------------------
      RTMP_Log(RTMP_LOGERROR, "%s, RTMP connect failed.", __FUNCTION__);
      RTMP_Close(r);
      return FALSE;
    }
  //----------
  r->dlg->AppendCInfo("建立连接：第1次连接。建立网络连接(NetConnection)成功");
  //----------------------------
  RTMP_LogPrintf("命令消息“Connect”发送完毕！\n");
  return TRUE;
}
```

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



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