

live555源代码下载（VC6工程）：<http://download.csdn.net/detail/leixiaohua1020/6374387>

liveMedia 项目(<http://www.live555.com/>)的源代码包括四个基本的库，各种测试代码以及Media Server。四个基本的库分别是: UsageEnvironment &TaskScheduler, groupsock, liveMedia和BasicUsageEnvironment。UsageEnvironment 和TaskScheduler 类用于事件的调度,实现异步读取事件的句柄的设置以及错误信息的输出。另外,还有一个HashTable 类定义了一个通用的hash 表,其它代码要用到这个表。这些都是抽象类,在应用程序中基于这些类来实现自己的子类。groupsock 类是对网络接口的封装,用于收发数据包。正如名字本身,groupsock 主要是面向多播数据的收发的,它也同时支持单播数据的收发。liveMedia 库中有一系列类,基类是Medium,这些类针对不同的流媒体类型和编码。

各种测试代码在testProgram 目录下,比如openRTSP 等,这些代码有助于理解liveMedia 的应用。
Media Server 是一个纯粹的RTSP 服务器。支持多种格式的媒体文件:

- * TS 流文件, 扩展名ts。
- * PS 流文件, 扩展名mpg。
- * MPEG-4视频基本流文件, 扩展名m4e。
- * MP3文件, 扩展名mp3。
- * WAV 文件(PCM), 扩展名wav。
- * AMR 音频文件, 扩展名.amr。
- * AAC 文件, ADTS 格式, 扩展名aac。

用live555开发应用程序

基于liveMedia 的程序,需要通过继承UsageEnvironment 抽象类和TaskScheduler 抽象类,定义相应的类来处理事件调度,数据读写以及错误处理。live 项目的源代码里有这些类的一个基本实现,这就是"BasicUsageEnvironment"库。BasicUsageEnvironment 主要是针对简单的控制台应用程序,利用select 实现事件获取和处理。这个库利用Unix 或者Windows 的控制台作为输入输出,处于应用程序原形或者调试的目的,可以用这个库用户可以开发传统的运行与控制台的应用。

通过使用自定义的"UsageEnvironment"和"TaskScheduler"抽象类的子类,这些应用程序就可以在特定的环境中运行, 不需要做过多的修改。需要指出的是在图形环境(GUI toolkit)下, 抽象类TaskScheduler 的子类在实现doEventLoop()的时候应该与图形环境自己的事件处理框架集成。

基本概念

先来熟悉在liveMedia 库中Source, Sink 以及Filter 等概念。Sink 就是消费数据的对象,比如把接收到的数据存储到文件,这个文件就是一个Sink 。Source 就是生产数据的对象,比如通过RTP 读取数据。数据流经过多个'source'和'sink's,下面是一个示例:

'source1' -> 'source2' (a filter) -> 'source3' (a filter) -> 'sink'

从其它Source 接收数据的source 也叫做"filters"。Module 是一个sink 或者一个filter。数据接收的终点是Sink 类,MediaSink 是所有Sink 类的基类。Sink 类实现对数据的处理是通过实现纯虚函数continuePlaying(),通常情况下continuePlaying 调用fSource->getNextFrame 来为Source 设置数据缓冲区,处理数据的回调函数等, fSource是MediaSink 的类型为FramedSource*的类成员。

基本控制流程

基于liveMedia 的应用程序的控制流程如下:
应用程序是事件驱动的,使用如下方式的循环

```
[cpp]
1. while (1) {
2.     通过查找读网络句柄的列表和延迟队列 (delay queue) 来发现需要完成的任务
3.     完成这个任务
4. }
```

对于每个sink,在进入这个循环之前,应用程序通常调用下面的方法来启动需要做的生成任务: someSinkObject->startPlaying()。任何时候,一个Module 需要获取数据都通过调用刚好在它之前的那个Module 的FramedSource::getNextFrame() 方法。这是通过纯虚函数FramedSource::doGetNextFrame() 实现的,每一个Source module 都有相应的实现。

Each 'source' module's implementation of "doGetNextFrame()" works by arranging for an 'after getting' function to be called (from an event handler) when new data be comes available for the caller.

Note that the flow of data from 'sources' to 'sinks' happens within each application, and doesn't necessarily correspond to the sending or receiving of network packets. For example, a server application (such as "testMP3Streamer") that sends RTP packets will do so using one or more "RTPSink" modules. These "RTPSink" modules receive data from other, "Source" modules (e.g., to read data from a file), and, as a side effect, transmit RTP packets.

live555代码解读之一：RTSP 连接的建立过程

RTSPServer 类用于构建一个RTSP 服务器，该类同时在其内部定义了一个RTSPClientSession类，用于处理单独的客户会话。

首先创建RTSP 服务器(具体实现类是DynamicRTSPServer)，在创建过程中，先建立Socket(ourSocket) 在TCP 的554 端口进行监听，然后把连接处理函数句柄(RTSPServer::incomingConnectionHandler)和socket 句柄传给任务调度器(taskScheduler)。

任务调度器把socket 句柄放入后面select 调用中用到的socket 句柄集(fReadSet)中,同时将socket 句柄和incomingConnectionHandler 句柄关联起来。接着，主程序开始进入任务调度器的主循环（doEventLoop），在主循环中调用系统函数select 阻塞，等待网络连接。

当RTSP 客户端输入(rtsp://192.168.1.109/1.mpg)连接服务器时,select 返回对应的socket,进而根据前面保存的对应关系，可找到对应处理函数句柄，这里就是前面提到的incomingConnectionHandler 了。在incomingConnectionHandler 中创建了RTSPClientSession,开始对这个客户端的会话进行处理。

live555代码解读之二：DESCRIBE 请求消息处理过程

RTSP 服务器收到客户端的DESCRIBE 请求后,根据请求URL(rtsp://192.168.1.109/1.mpg),找到对应的流媒体资源，返回响应消息。live555中的ServerMediaSession 类用来处理会话中描述，它包含多个（音频或视频）的子会话描述(ServerMediaSubsession)。

上节我们谈到RTSP 服务器收到客户端的连接请求,建立了RTSPClientSession 类,处理单独的客户会话。在建立RTSPClientSession 的过程中，将新建立的socket 句柄（clientSocket）和RTSP 请求处理函数句柄RTSPClientSession::incomingRequestHandler 传给任务调度器，由任务调度器对两者进行一对一关联。当客户端发出RTSP 请求后,服务器主循环中的select调用返回,根据socket 句柄找到对应的incomingRequestHandler,开始消息处理。先进行消息的解析,如果发现请求是DESCRIBE 则进入handleCmd_DESCRIBE 函数。根据客户端请求URL 的后缀(例如是1.mpg), 调用成员函数DynamicRTSPServer::lookupServerMediaSession查找对应的流媒信息ServerMediaSession。如果ServerMediaSession 不存在,但是本地存在1.mpg 文件,则创建一个新的ServerMediaSession。在创建ServerMediaSession 过程中，根据文件后缀.mpg,创建媒体MPEG-1or2的解复用器(MPEG1or2FileServerDemux)。再由MPEG1or2FileServerDemux 创建一个子会话描述MPEG1or2DemuxedServerMediaSubsession。最后由ServerMediaSession 完成组装响应消息中的SDP 信息(SDP 组装过程见下面的描述),然后将响应消息发给客户端，完成一次消息交互。

SDP 消息组装过程

ServerMediaSession 负责产生会话公共描述信息，子会话描述由MPEG1or2DemuxedServerMediaSubsession 产生。MPEG1or2DemuxedServerMediaSubsession在其父类成员函数OnDemandServerMediaSubsession::sdpLines()中生成会话描述信息。在sdpLines() 实现里面，创建一个虚构(dummy) 的FramedSource(具体实现类为MPEG1or2AudioStreamFramer 和MPEG1or2VideoStreamFramer)和RTPSink（具体实现类为MPEG1or2AudioRTPSink 和MPEG1or2VideoRTPSink），最后调用setSDPLinesFromRTPSink(...)成员函数生成子会话描述。

以上涉及到的类以及继承关系：

```
Medium <- ServerMediaSession

Medium <- ServerMediaSubsession <- OnDemandServerMediaSubsession <- MPEG1or2DemuxedServerMediaSubsession

Medium <- MediaSource <- FramedSource <- FramedFileSource <- ByteStreamFileSource

Medium <- MediaSource <- FramedSource <- MPEG1or2DemuxedElementaryStream

Medium <- MPEG1or2FileServerDemux

Medium <- MPEG1or2Demux

Medium <- MediaSource <- FramedSource <- MPEG1or2DemuxedElementaryStream

Medium <- MediaSource <- FramedSource <- FramedFilter <- MPEGVideoStreamFramer <- MPEG1or2VideoStreamFramer

Medium <- MediaSink <- RTPSink <- MultiFramedRTPSink <- VideoRTPSink <- MPEG1or2VideoRTPSink
```

live555代码解读之三：SETUP 和PLAY 请求消息处理过程

前面已经提到RTSPClientSession 类，用于处理单独的客户会话。其类成员函数handleCmd_SETUP()处理客户端的SETUP 请求。调用parseTransportHeader()对SETUP 请求的传输头解析,调用子会话(这里具体实现类为OnDemandServerMediaSubsession)的getStreamParameters()函数获取流媒体发送传输参数。将这些参数组装成响应消息，返回给客户端。

获取发送传输参数的过程：

调用子会话(具体实现类MPEG1or2DemuxedServerMediaSubsession) 的createNewStreamSource(...) 创建MPEG1or2VideoStreamFramer,选择发送传输参数,并调用子会话的createNewRTPSink(...)创建MPEG1or2VideoRTPSink。同时将这些信息保存在StreamState 类对象中,用于记录流的状态。

客户端发送两个SETUP 请求，分别用于建立音频和视频的RTP 接收。

PLAY 请求消息处理过程：

RTSPClientSession 类成员函数handleCmd_PLAY()处理客户端的播放请求。首先调用子会话的startStream(), 内部调用MediaSink::startPlaying(...), 然后是MultiFramedRTPSink::continuePlaying(), 接着调用MultiFramedRTPSink::buildAndSendPacket(...)。buildAndSendPacket 内部先设置RTP 包头，内部再调用MultiFramedRTPSink::packFrame()填充编码帧数据。

packFrame 内部通过FramedSource::getNextFrame(), 接着MPEGVideoStreamFramer::doGetNextFrame(), 再接着经过MPEGVideoStreamFramer::continueReadProcessing(), FramedSource::afterGetting(...), MultiFramedRTPSink::afterGettingFrame(...), MultiFramedRTPSink::afterGettingFrame1(...) 等一系列繁琐调用, 最后到了MultiFramedRTPSink::sendPacketIfNecessary(), 这里才真正发送RTP 数据包。然后是计算下一个数据包发送时间, 把MultiFramedRTPSink::sendNext(...)函数句柄传给任务调度器, 作为一个延时事件调度。在主循环中, 当MultiFramedRTPSink::sendNext() 被调度时, 又开始调用MultiFramedRTPSink::buildAndSendPacket(...)开始新的发送数据过程, 这样客户端可以源源不断的收到服务器传来的RTP 包了。发送RTP 数据包的间隔计算方法:

Update the time at which the next packet should be sent, based on the duration of the frame that we just packed into it.

涉及到一些类有:

MPEGVideoStreamFramer: A filter that breaks up an MPEG video elementary stream into headers and frames

MPEG1or2VideoStreamFramer: A filter that breaks up an MPEG 1 or 2 video elementary stream into frames for: Video_Sequence_Header, GOP_Header, Picture_Header

MPEG1or2DemuxedElementaryStream: A MPEG 1 or 2 Elementary Stream, demultiplexed from a Program Stream

MPEG1or2Demux: Demultiplexer for a MPEG 1 or 2 Program Stream

ByteStreamFileSource: A file source that is a plain byte stream (rather than frames)

MPEGProgramStreamParser: Class for parsing MPEG program stream

StreamParser: Abstract class for parsing a byte stream

StreamState: A class that represents the state of an ongoing stream

rtsp 简介(ZT)

Real Time Streaming Protocol 或者RTSP(实时流媒体协议),是由Real network 和Netscape共同提出的如何有效地在IP 网络上传输流媒体数据的应用层协议。RTSP 提供一种可扩展的框架,使能够提供能控制的,按需传输实时数据,比如音频和视频文件。源数据可以包括现场数据的反馈和存贮的文件。rtsp 对流媒体提供了诸如暂停,快进等控制,而它本身并不传输数据,rtsp 作用相当于流媒体服务器的远程控制。传输数据可以通过传输层的tcp, udp协议, rtsp 也提供了基于rtp 传输机制的一些有效的方法。

RTSP 消息格式:

RTSP 的消息有两大类,一是请求消息(request),一是回应消息(response),两种消息的格式不同。

请求消息:

方法URI RTSP 版本CR LF

消息头CR LF CR LF

消息体CR LF

其中方法包括OPTION 回应中所有的命令,URI 是接受方的地址,例如:rtsp://192.168.20.136。

RTSP 版本一般都是RTSP/1.0.每行后面的CR LF 表示回车换行,需要接受端有相应的解析,最后一个消息头需要有两个CR LF

回应消息:

RTSP 版本状态码解释CR LF

消息头CR LF CR LF

消息体CR LF

其中RTSP 版本一般都是RTSP/1.0,状态码是一个数值,200表示成功,解释是与状态码对应的文本解释。

简单的rtsp 交互过程:

C 表示rtsp 客户端,S 表示rtsp 服务端

```

[html]
1. 1.C->S:OPTION request //询问S 有哪些方法可用
2. 1.S->C:OPTION response //S 回应信息中包括提供的所有可用方法
3. 2.C->S:DESCRIBE request //要求得到S 提供的媒体初始化描述信息
4. 2.S->C:DESCRIBE response //S 回应媒体初始化描述信息,主要是sdp
5. 3.C->S:SETUP request //设置会话的属性,以及传输模式,提醒S 建立会
6. 话
7. 3.S->C:SETUP response //S 建立会话,返回会话标识符,以及会话相关信息
8. 4.C->S:PLAY request //C 请求播放
9. 4.S->C:PLAY response //S 回应该请求的信息
10. S->C:发送流媒体数据
11. 5.C->S:TEARDOWN request //C 请求关闭会话
12. 5.S->C:TEARDOWN response //S 回应该请求

```

上述的过程是标准的、友好的rtsp 流程,但实际的需求中并不一定按部就班来。其中第3和4步是必需的!第一步,只要服务器客户端约定好,有哪些方法可用,则option 请求可以不要。第二步,如果我们有其他途径得到媒体初始化描述信息(比如http 请求等等),则我们也不需要通过rtsp 中的describe 请求来完成。第五步,可以根据系统需求的设计来决定是否需要。

rtsp 中常用方法:

1.OPTION

目的是得到服务器提供的可用方法:

OPTIONS rtsp://192.168.20.136:5000/xxx666 RTSP/1.0

CSeq: 1 //每个消息都有序号来标记,第一个包通常是option 请求消息

User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)

服务器的回应信息包括提供的一些方法,例如:

RTSP/1.0 200 OK

Server: UServer 0.9.7_rc1

Cseq: 1 //每个回应消息的cseq 数值和请求消息的cseq 相对应

Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, SCALE,GET_PARAMETER //服务器提供的可用的方法

2.DESCRIBE

C 向S 发起DESCRIBE 请求,为了得到会话描述信息(SDP):

DESCRIBE rtsp://192.168.20.136:5000/xxx666 RTSP/1.0

CSeq: 2

token:

Accept: application/sdp

User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)

服务器回应一些对此会话的描述信息(sdp):

RTSP/1.0 200 OK

Server: UServer 0.9.7_rc1

Cseq: 2

x-prev-url: rtsp://192.168.20.136:5000

x-next-url: rtsp://192.168.20.136:5000

x-Accept-Retransmit: our-retransmit

x-Accept-Dynamic-Rate: 1

Cache-Control: must-revalidate

Last-Modified: Fri, 10 Nov 2006 12:34:38 GMT

Date: Fri, 10 Nov 2006 12:34:38 GMT

Expires: Fri, 10 Nov 2006 12:34:38 GMT

Content-Base: rtsp://192.168.20.136:5000/xxx666/

Content-Length: 344

Content-Type: application/sdp

v=0 //以下都是sdp 信息

o=OnewaveUServerNG 1451516402 1025358037 IN IP4 192.168.20.136

s=/xxx666

u=http://

e=admin@

c=IN IP4 0.0.0.0

t=0 0

a=isma-compliance:1,1.0,1

a=range:npt=0-

m=video 0 RTP/AVP 96 //m 表示媒体描述,下面是对会话中视频通道的媒体描述

a=rtpmap:96 MP4V-ES/90000

a=fmtp:96

profile-level-id=245;config=000001B0F5000001B509000001000000012000C888B0E0E0FA62D

089028307

a=control:trackID=0//trackID=0表示视频流用的是通道0

3.SETUP

客户端提醒服务器建立会话,并确定传输模式:

SETUP rtsp://192.168.20.136:5000/xxx666/trackID=0 RTSP/1.0

CSeq: 3

Transport: RTP/AVP/TCP;unicast;interleaved=0-1

User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)

//uri 中带有trackID=0,表示对该通道进行设置。Transport 参数设置了传输模式,包的结构。接下来的数据包头部第二个字节位置就是interleaved,它的值是每个通道都不同的,trackID=0的interleaved 值有两个0或1,0表示rtp 包,1表示rtcp 包,接受端根据interleaved 的值来区别是哪种数据包。

服务器回应信息:

RTSP/1.0 200 OK

Server: UServer 0.9.7_rc1

Cseq: 3

Session: 6310936469860791894 //服务器回应的会话标识符

Cache-Control: no-cache

Transport: RTP/AVP/TCP;unicast;interleaved=0-1;ssrc=6B8B4567

4.PLAY

客户端发送播放请求:

PLAY rtsp://192.168.20.136:5000/xxx666 RTSP/1.0

CSeq: 4

Session: 6310936469860791894

Range: npt=0.000- //设置播放时间的范围

User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)

服务器回应信息:

RTSP/1.0 200 OK

Server: UServer 0.9.7_rc1

Cseq: 4

Session: 6310936469860791894

Range: npt=0.000000-

RTP-Info: url=trackID=0;seq=17040;rtptime=1467265309

//seq 和rtptime 都是rtp 包中的信息

5.TEARDOWN

客户端发起关闭请求:

TEARDOWN rtsp://192.168.20.136:5000/xxx666 RTSP/1.0

CSeq: 5

Session: 6310936469860791894

User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)

服务器回应:

RTSP/1.0 200 OK

Server: UServer 0.9.7_rc1

Cseq: 5

Session: 6310936469860791894

Connection: Close

以上方法都是交互过程中最为常用的,其它还有一些重要的方法如get/set_parameter,pause,redirect 等等

附录

sdp 的格式



| | |
|-----|--|
| | [html]   |
| 1. | v=<version> |
| 2. | o=<username> <session id> <version> <network type> <address type> <address> |
| 3. | s=<session name> |
| 4. | i=<session description> |
| 5. | u=<URI> |
| 6. | e=<email address> |
| 7. | p=<phone number> |
| 8. | c=<network type> <address type> <connection address> |
| 9. | b=<modifier>:<bandwidth-value> |
| 10. | t=<start time> <stop time> |
| 11. | r=<repeat interval> <active duration> <list of offsets from start-time> |
| 12. | z=<adjustment time> <offset> <adjustment time> <offset> |
| 13. | k=<method> |
| 14. | k=<method>:<encryption key> |
| 15. | a=<attribute> |
| 16. | a=<attribute>:<value> |
| 17. | m=<media> <port> <transport> <fmt list> |
| 18. | v = (协议版本) |
| 19. | o = (所有者/创建者和会话标识符) |
| 20. | s = (会话名称) |
| 21. | i = * (会话信息) |
| 22. | u = * (URI 描述) |
| 23. | e = * (Email 地址) |
| 24. | p = * (电话号码) |
| 25. | c = * (连接信息) |
| 26. | b = * (带宽信息) |
| 27. | z = * (时间区域调整) |
| 28. | k = * (加密密钥) |
| 29. | a = * (0 个或多个会话属性行) |
| 30. | 时间描述： |
| 31. | t = (会话活动时间) |
| 32. | r = * (0或多个重复次数) |
| 33. | 媒体描述： |
| 34. | m = (媒体名称和传输地址) |
| 35. | i = * (媒体标题) |
| 36. | c = * (连接信息- 如果包含在会话层则该字段可选) |
| 37. | b = * (带宽信息) |
| 38. | k = * (加密密钥) |
| 39. | a = * (0 个或多个媒体属性行) |

参考文章：rfc2326（rtsp）；rfc2327（sdp）



RTSP 点播消息流程实例

（客户端：VLC，RTSP 服务器：LIVE555 Media Server）



1)C(Client)-> M(Media Server)

| | |
|----|--|
| | [html]   |
| 1. | OPTIONS rtsp://192.168.1.109/1.mpg RTSP/1.0 |
| 2. | CSeq: 1 |
| 3. | user-Agent: VLC media player(LIVE555 Streaming Media v2007.02.20) |
| 4. | 1)M -> C |
| 5. | RTSP/1.0 200 OK |
| 6. | CSeq: 1 |
| 7. | Date: wed, Feb 20 2008 07:13:24 GMT |
| 8. | Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE |



2)C -> M

| | |
|----|--|
| | [html]   |
| 1. | DESCRIBE rtsp://192.168.1.109/1.mpg RTSP/1.0 |
| 2. | CSeq: 2 |
| 3. | Accept: application/sdp |
| 4. | User-Agent: VLC media player(LIVE555 Streaming Media v2007.02.20) |



2)M -> C

```
[html]  
1. RTSP/1.0 200 OK
2. CSeq: 2
3. Date: wed, Feb 20 2008 07:13:25 GMT
4. Content-Base: rtsp://192.168.1.109/1.mpg/
5. Content-type: application/sdp
6. Content-length: 447
7. v=0
8. o=- 2284269756 1 IN IP4 192.168.1.109
9. s=MPEG-1 or 2 program Stream, streamed by the LIVE555 Media Server
10. i=1.mpg
11. t=0 0
12. a=tool:LIVE555 Streaming Media v2008.02.08
13. a=type:broadcast
14. a=control:*
15. a=range:npt=0-66.181
16. a=x-qt-text-nam:MPEG-1 or Program Stream, streamed by the LIVE555 Media Server
17. a=x-qt-text-inf:1.mpg
18. m=video 0 RTP/AVP 32
19. c=IN IP4 0.0.0.0
20. a=control:track1
21. m=audio 0 RTP/AVP 14
22. c=IN IP4 0.0.0.0
23. a=control:track2
```



3)C -> M

```
[html]  
1. SETUP rtsp://192.168.1.109/1.mpg/track1 RTSP/1.0
2. CSeq: 3
3. Transport: RTP/AVP; unicast;client_port=1112-1113
4. User-Agent: VLC media player(LIVE555 Streaming Media v2007.02.20)
```


3)M -> C

```
[html]  
1. RTSP/1.0 200 OK
2. CSeq: 3
3. Date: wed, Feb 20 2008 07:13:25 GMT
4. Transport:
5. RTP/AVP;unicast;destination=192.168.1.222;source=192.168.1.109;client_port=1112-1113;server
6. _port=6970-6971
7. Session: 3
```



4)C -> M

```
[html]  
1. SETUP rtsp://192.168.1.109/1.mpg/track2 RTSP/1.0
2. CSeq: 4
3. Transport: RTP/AVP; unicast;client_port=1114-1115
4. Session: 3
5. User-Agent: VLC media player(LIVE555 Streaming Media v2007.02.20)
```


4)M -> C

```
[html]  
1. RTSP/1.0 200 OK
2. CSeq: 4
3. Date: wed, Feb 20 2008 07:13:25 GMT
4. Transport:
5. RTP/AVP;unicast;destination=192.168.1.222;source=192.168.1.109;client_port=1114-1115;server
6. _port=6972-6973
7. Session: 3
```

5)C -> M

```
[html]  
1. PLAY rtsp://192.168.1.109/1.mpg/ RTSP/1.0
2. CSeq: 5
3. Session: 3
4. Range: npt=0.000-
5. User-Agent: VLC media player(LIVE555 Streaming Media v2007.02.20)
```

5)M -> C

[html]  

```
1. RTSP/1.0 200 OK
2. CSeq: 5
3. Range: npt=0.000-
4. Session: 3
5. RTP-Info:
6. url=rtsp://192.168.1.109/1.mpg/track1;seq=9200;rtptime=214793785,url=rtsp://192.168.1.109/1.
7. mpg/track2;seq=12770;rtptime=31721
8. (开始传输流媒体...)
```

文章标签：[live555](#) [源代码](#) [概述](#)

个人分类：[Live555](#)

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