Python 实现 RTSP 客户端模拟器（TCP 方式）

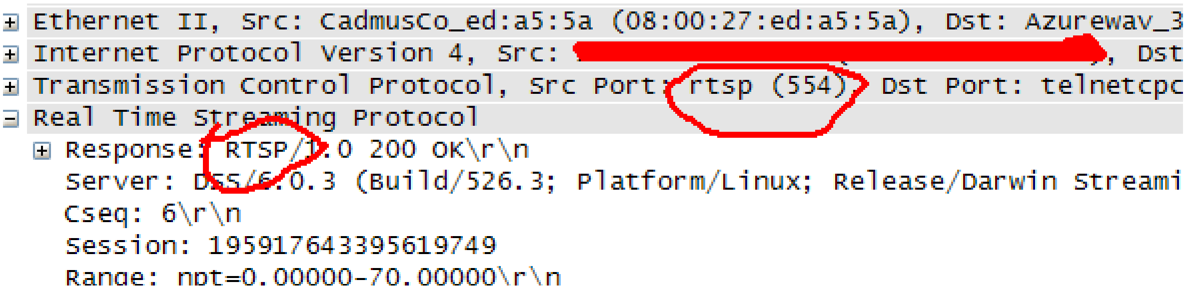
由于某种需求，工作中需要自己要开发 RTSP 客户端模拟器……这里以 DarwinStreamingServer（简称 DSS）为例进行演示，把思路记录下来，算是开发了一个测试工具，也方便我以后查阅。

在我之前的文章（http://www.cnblogs.com/MikeZhang/archive/2012/09/16/RTSPoverTCPUDP20120916.html ）中介绍过怎样通过 TCP 的方式来访问 DSS，在那个实例中，我用的是 VLC 作为客户端，通过命令行参数进行调用的。

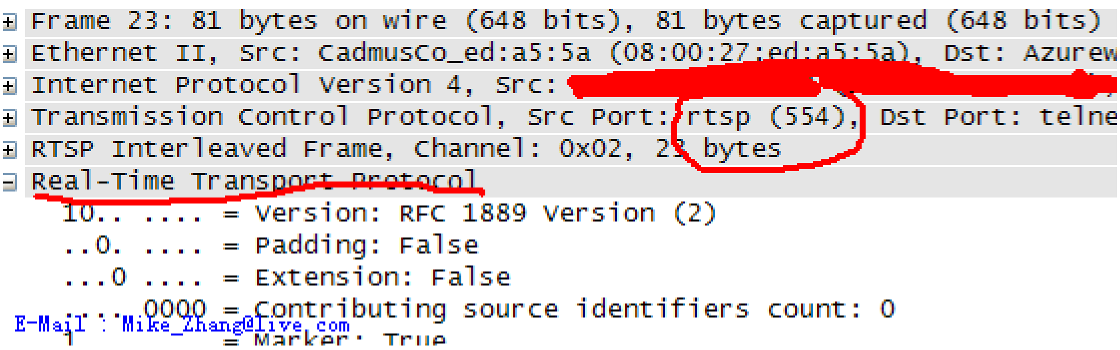
一、通信端口分析

首先通过抓包分析确定数据通信端口。

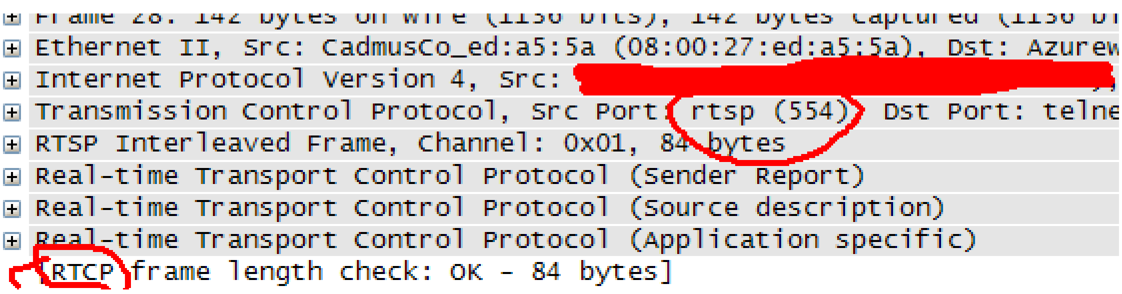
RTSP 数据包截图：



RTP 数据包截图：



RTCP 数据包截图：



由图可知，在以 TCP 方式访问 DSS 时，RTSP 数据、RTP 数据和 RTCP 数据都是通过 554 端

口进行传输的，所以 DSS 服务器只通过 554 端口和客户端通信。

二、通信过程分析

RTSP	179	OPTIONS rtsp:
RTSP	260	Reply: RTSP/1
RTSP	205	DESCRIBE rtsp
RTSP	235	SETUP rtsp://
RTSP	431	Reply: RTSP/1
RTSP	264	SETUP rtsp://
RTSP	431	Reply: RTSP/1
RTSP	225	PLAY rtsp://1
RTSP	401	Reply: RTSP/1
RTSP	210	TEARDOWN rtsp
RTSP	233	Reply: RTSP/1
RTSP/SDP	1230	Reply: RTSP/1

OPTIONS ：查询到服务器所提供的方法；

DESCRIBE ：得到会话描述信息（SDP）；

SETUP ：提醒服务器建立会话，并确定传输模式；

PLAY ：客户端发送播放请求；

TEARDOWN ：客户端发起关闭请求；

当然中间还有 RTP 和 RTCP 的交互，这里就不叙述了。

三、模拟器实现

1、建立链接

```
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((m_Vars["defaultServerIp"],m_Vars["defaultServerPort"]))
```

2、查询服务器所提供的方法

向服务器发送 OPTIONS 请求，得到服务器所提供的方法。

```
s.send(genmsg_OPTIONS(m_Vars["defaultTestUrl"],seq,m_Vars["defaultUserAgent"]))
print s.recv(m_Vars["bufLen"])
```

3、得到会话描述信息

向服务器发送 DESCRIBE 请求，得到 SDP

```
s.send(genmsg_DESCRIBE(m_Vars["defaultTestUrl"],seq,m_Vars["defaultUserAgent"]))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
```

4、提醒服务器建立会话，并确定传输模式

向服务器发送 SETUP 请求，通知服务器产生 session，并和服务器确定传输模式等。

```
s.send(genmsg_SETUP(m_Vars["defaultTestUrl"] +
"/trackID=3",seq,m_Vars["defaultUserAgent"]))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
seq = seq + 1

sessionId = decodeMsg(msg1)['Session']

s.send(genmsg_SETUP2(m_Vars["defaultTestUrl"] +
"/trackID=4",seq,m_Vars["defaultUserAgent"],sessionId))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
```

5、客户端发起播放请求

向服务器发送 PLAY 请求，通知服务器发送 RTP 数据。

```
s.send(genmsg_PLAY(m_Vars["defaultTestUrl"] + "/",seq,m_Vars["defaultUserAgent"],sessionId))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
```

客户端接收 RTP 数据

```
while True :
	msgRcv = s.recv(m_Vars["bufLen"])
	if 0 == len(msgRcv) : break
	print len(msgRcv)
```

6、客户端发起关闭请求

客户端向服务器发送 TREADOWN 请求，通知服务器关闭。

```
s.send(genmsg_TEARDOWN(m_Vars["defaultTestUrl"] +
"/",seq,m_Vars["defaultUserAgent"],sessionId))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
```

完整代码：

```
#! /usr/bin/python

import socket,time,string,random,thread

m_Vars = {
	"bufLen" : 1024 * 10,
	"defaultServerIp" : "192.168.1.100",
	"defaultServerPort" : 554,
```

```
        "defaultTestUrl" : "rtsp://192.168.1.100/test1.mp4",
        "defaultUserAgent" : "LibVLC/2.0.3 (LIVE555 Streaming Media 
v2011.12.23)"
}

def genmsg_OPTIONS(url,seq,userAgent):
        msgRet = "OPTIONS " + url + " RTSP/1.0\r\n"
        msgRet += "CSeq: " + str(seq) + "\r\n"
        msgRet += "User-Agent: " + userAgent + "\r\n"
        msgRet += "\r\n"
        return msgRet

def genmsg_DESCRIBE(url,seq,userAgent):
        msgRet = "DESCRIBE " + url + " RTSP/1.0\r\n"
        msgRet += "CSeq: " + str(seq) + "\r\n"
        msgRet += "User-Agent: " + userAgent + "\r\n"
        msgRet += "Accept: application/sdp\r\n"
        msgRet += "\r\n"
        return msgRet

def genmsg_SETUP(url,seq,userAgent):
        msgRet = "SETUP " + url + " RTSP/1.0\r\n"
        msgRet += "CSeq: " + str(seq) + "\r\n"
        msgRet += "User-Agent: " + userAgent + "\r\n"
        msgRet += "Transport: RTP/AVP/TCP;unicast;interleaved=0-1\r\n"
        msgRet += "\r\n"
        return msgRet

def genmsg_SETUP2(url,seq,userAgent,sessionId):
        msgRet = "SETUP " + url + " RTSP/1.0\r\n"
        msgRet += "CSeq: " + str(seq) + "\r\n"
        msgRet += "User-Agent: " + userAgent + "\r\n"
        msgRet += "Transport: RTP/AVP/TCP;unicast;interleaved=2-3\r\n"
        msgRet += "Session: " + sessionId + "\r\n"
        msgRet += "\r\n"
        return msgRet

def genmsg_PLAY(url,seq,userAgent,sessionId):
        msgRet = "PLAY " + url + " RTSP/1.0\r\n"
        msgRet += "CSeq: " + str(seq) + "\r\n"
        msgRet += "User-Agent: " + userAgent + "\r\n"
        msgRet += "Session: " + sessionId + "\r\n"
        msgRet += "\r\n"
        return msgRet

def genmsg_TEARDOWN(url,seq,userAgent,sessionId):
        msgRet = "TEARDOWN " + url + " RTSP/1.0\r\n"
        msgRet += "CSeq: " + str(seq) + "\r\n"
        msgRet += "User-Agent: " + userAgent + "\r\n"
        msgRet += "Session: " + sessionId + "\r\n"
```

```
        msgRet += "\r\n"
        return msgRet

def decodeMsg(strContent):
        mapRetInf = {}
        for str in [elem for elem in strContent.split("\n") if len(elem) > 1]
[2:-1]:
                #print str
                tmp2 = str.split(":")
                mapRetInf[tmp2[0]]=tmp2[1][:-1]
        #print mapRetInf
        return mapRetInf


s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((m_Vars["defaultServerIp"],m_Vars["defaultServerPort"]))
seq  = 1

print
genmsg_OPTIONS(m_Vars["defaultTestUrl"],seq,m_Vars["defaultUserAgent"])
s.send(genmsg_OPTIONS(m_Vars["defaultTestUrl"],seq,m_Vars["defaultUserAgent"
]))
print s.recv(m_Vars["bufLen"])
seq = seq + 1

s.send(genmsg_DESCRIBE(m_Vars["defaultTestUrl"],seq,m_Vars["defaultUserAgent
"]))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
seq = seq + 1

s.send(genmsg_SETUP(m_Vars["defaultTestUrl"] +
"/trackID=3",seq,m_Vars["defaultUserAgent"]))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
seq = seq + 1

sessionId = decodeMsg(msg1)['Session']

s.send(genmsg_SETUP2(m_Vars["defaultTestUrl"] +
"/trackID=4",seq,m_Vars["defaultUserAgent"],sessionId))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
seq = seq + 1

s.send(genmsg_PLAY(m_Vars["defaultTestUrl"] +
"/",seq,m_Vars["defaultUserAgent"],sessionId))
msg1 = s.recv(m_Vars["bufLen"])
print msg1
seq = seq + 1

while True :
        #s.send(genmsg_ANNOUNCE(m_Vars["defaultServerIp"]))
        msgRcv = s.recv(m_Vars["bufLen"])
        if 0 == len(msgRcv) : break
```

```
        print len(msgRcv)
        #time.sleep(5)

s.send(genmsg_TEARDOWN(m_Vars["defaultTestUrl"] +
"/",seq,m_Vars["defaultUserAgent"],sessionId))
msg1 = s.recv(m_Vars["bufLen"])
print msg1

s.close()
```

四、运行效果

```
Transport: RTP/AVP/TCP;unicast;interleaved=2-3;ssrc=5BF0EE4F


RTSP/1.0 200 OK
Server: DSS/6.0.3 (Build/526.3; Platform/Linux; Release/Darwin Streaming Server;
 State/Development; )
Cseq: 5
Session: 195917643529528495
Range: npt=0.00000-70.00000
RTP-Info: url=[redacted]est1.mp4/trackID=3;seq=59036;rtptime=43990
9911,url=[redacted]test1.mp4/trackID=4;seq=25906;rtptime=2074870102



10240
1796
125
324
655
215
388
826
459
160
```