移动流媒体初步研究报告

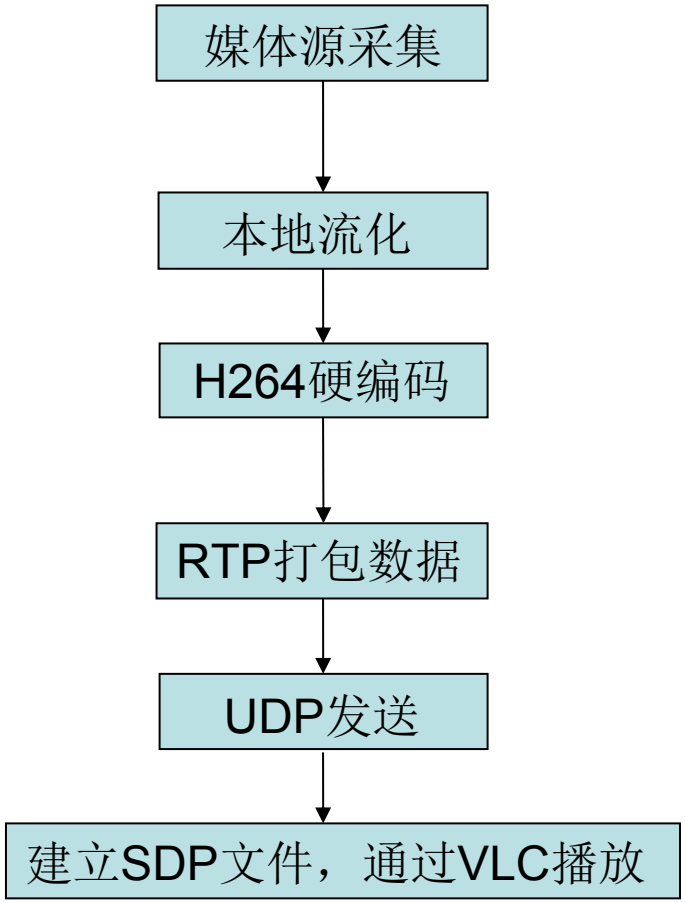
H264、RTP等分析

移动流媒体概述

随着3G移动通信技术的逐渐成熟，以及3G网络设备和终端设备的不断完善，移动通信网不仅能够提供传统的语音服务，还能提供高速率的宽带视频服务，支持高质量的语音、分组数据业务以及实时视频传输。对移动用户而言，流媒体能够播放音/视频和多媒体内容，用户也可以进行点播，具有交互性。这一特点与移动通信固有的移动性相结合，使移动用户能够随时随地获得点播实时的流媒体信息，灵活性很高。

目录

1、android实时录制视频传输播放流程



2、录制视频MediaRecorder

android提供了MediaRecorder这个类来进行音、视频媒体的采集编码处理操作。由于移动设备的硬件支持，所以可以通过摄像头直接进行视频源的采集。值得注意的是采用硬编码的流化方式只支持3gp、mp4视频格式。主要实例代码如下：

```
mMediaRecorder.setPreviewDisplay(mSurfaceHolder.getSurface());//设置显示方式
mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);//设置视频源为摄像头
mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);//设置音频源为麦克风
mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);//设置文件格式为3gp
mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.H264);//采用264编码
mMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
mMediaRecorder.setVideoSize(320, 240);//视频分辨率
mMediaRecorder.setVideoFrameRate(15);//视频帧率
```

3、LocalSocket发送本地流

由于需要实时的流传输，所以需要对视频源进行流化并同步映射到网络传输通道内。Java提供了LocalSocket类来实现。主要代码：

```
lss = new LocalServerSocket("H264");
receiver.connect(new LocalSocketAddress("H264"));
receiver.setReceiveBufferSize(500000);
receiver.setSendBufferSize(500000);
sender = lss.accept();
sender.setReceiveBufferSize(500000);
sender.setSendBufferSize(500000);
```

4、H264硬编码

硬编码：通过调用Android系统自带的Camera录制视频，实际上是调用了底层的高清编码硬件模块，也即显卡，不使用CPU，速度快。

软编码：使用CPU进行编码，如常见C/C++代码，一般编译生成的二进制都是的，速度相对较慢。例如使用Android NDK编译H264生成so库，编写jni接口，再使用java调用so库。

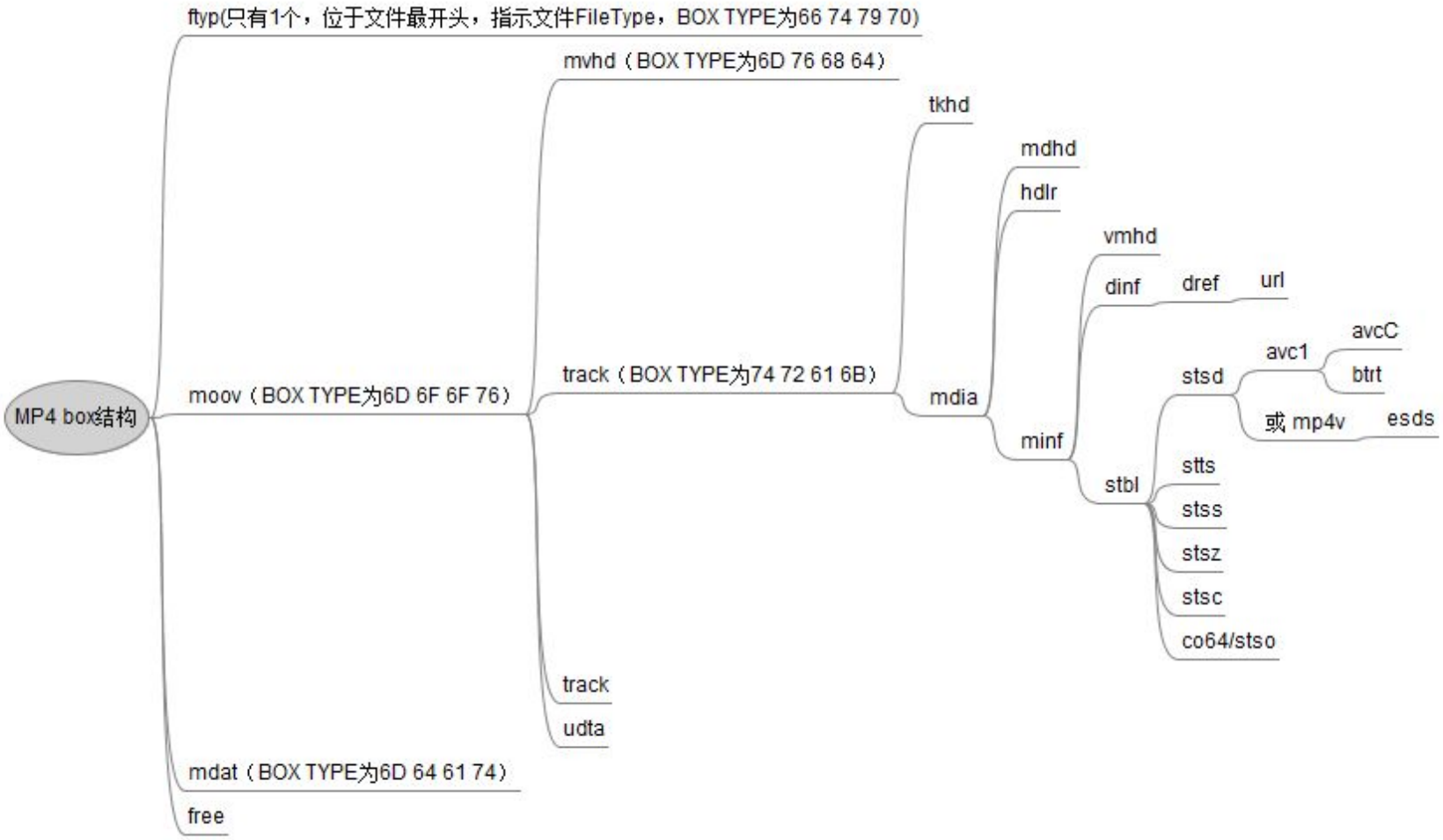
对于隶属于mp4格式的3gp，我们也需要遵循RFC文档中RTP协议规范，提取mp4文件中用于编码的sps、pps以及码流，这里需要对mp4文件格式做一下简单介绍。

MP4封装格式对应标准为 ISO/IEC 14496-12，是基于QuickTime容器格式定义，媒描述与媒体数据分开，目前被广泛应用于封装h.264视频和ACC音频，是高清视频/HDV的代表。

MP4文件中所有数据都封装在box中，（对应QuickTime中的atom），即MP4文件是由若干个box组成，每个box有长度和类型，每个box中还可以包含另外的子box（称container box）。

一个MP4文件首先会有且只有一个“ftyp”类型的box，作为MP4格式的标志并包含关于文件的一些信息；之后会有且只有一个“moov”类型的box（Movie Box），它是一种container box，子box包含了媒体的metadata信息；MP4文件的媒体数据包含在“mdat”类型的box（Midia Data Box）中，该类型的box也是container box，可以有多个，也可以没有（当媒体数据全部引用其他文件时），媒体数据的结构由metadata进行描述。MP4中box存储方式为大端模式。一般，标准的box开头会有四个字节的box size。

box结构图


MP4 box结构
ftyp(只有1个，位于文件最开头，指示文件FileType，BOX TYPE为66 74 79 70)
moov（BOX TYPE为6D 6F 6F 76）
mvhd（BOX TYPE为6D 76 68 64）
track（BOX TYPE为74 72 61 6B）
tkhd
mdia
mdhd
hdlr
minf
vmhd
dinf
dref
url
stbl
stsd
avc1
avcC
btrt
或 mp4v
esds
stts
stss
stsz
stsc
co64/stso
track
udta
mdat（BOX TYPE为6D 64 61 74）
free

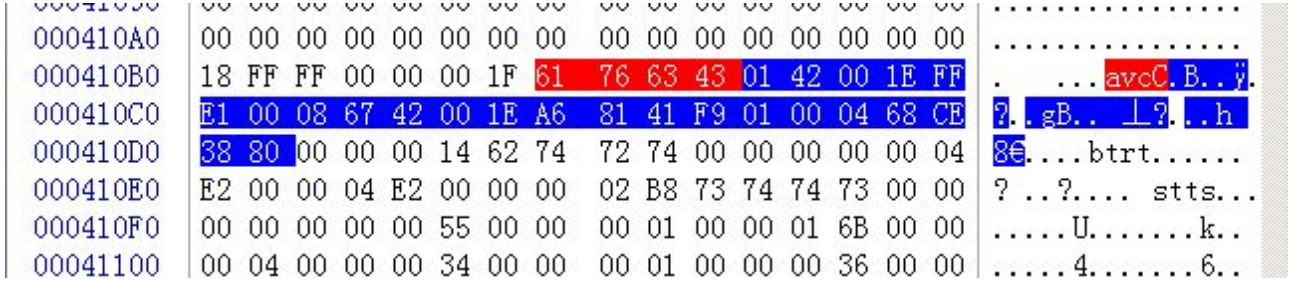
SPS:序列参数集,SPS作用于一系列连续的编码图像。
PPS:图像参数集,PPS作用于编码视频序列中一个或多个独立的图像。

MP4文件中，SPS和PPS存在于AVCDecoderConfigurationRecord， 首先要定位avcC。
关于AVCDecoderConfigurationRecord结构定义为：

unsigned int(8) configurationVersion=1	**8bit**	**版本号，1**
unsigned int(8) AVCProfileIndication	8bit	sps[1]
unsigned int(8) profile_compatibility	8bit	sps[2]
unsigned int(8) AVCLevelIndication	8bit	sps[3]
bit(6) reserved='111111'b	6bit	Reserved,111111
unsigned int(2) lengthSizeMinusOne	2bit	H264中NALU长度，计算方法为 1+(lengthSizeMinusOne&3)
bit(3) reserved='111'b	3bit	Reserved,111
unsigned int(5) numOfSequenceParameterSets	5bit	sps个数，numOfSequenceParameterSets&0x1F
unsigned int(16) sequenceParameterSetLength	16bit	sps长度
bit(8*sequenceParameterSetLength) sequenceParameterSetNALUint		sps内容
unsigned int(8) numOfPictureParameterSets	8bit	pps个数，一般为1
unsigned int(16) PictureParameterSetLength	16bit	pps长度
bit(8*sequenceParameterSetLength) pictureParameterSetNALUint		pps内容

用WinHex打开的mp4文件实例图：

avcC



000410A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000410B0	18	FF	FF	00	00	00	1F	61	76	63	43	01	42	00	1E	FF	. ...avcC.B..ÿ
000410C0	E1	00	08	67	42	00	1E	A6	81	41	F9	01	00	04	68	CE	?..gB.. ⊥?..h
000410D0	38	80	00	00	00	14	62	74	72	74	00	00	00	00	00	04	8€....btrt.....
000410E0	E2	00	00	04	E2	00	00	00	02	B8	73	74	74	73	00	00	? ..?.... stts...
000410F0	00	00	00	00	00	55	00	00	00	01	00	00	01	6B	00	00	.....U.......k..
00041100	00	04	00	00	00	34	00	00	00	01	00	00	00	36	00	00	.....4.......6..

依照前面的结构图进行详细分析

61 76 63 43	avcC BOX_TYPE
01	版本号
42	AVCProfileIndication
00	profile_compatibility
1E	AVCLevelIndication
FF	NALU长度，为4
E1	sps个数，低五位，为1
00 08	sps长度，为8
67 42 00 1E A6 81 41 F9	sps内容
01	pps个数，为1
00 04	pps长度，为4
68 CE 38 80	pps内容

所以，提取的SPS和PPS分别为67 42 00 1E A6 81 41 F9和68 CE 38 80

5、RTP分析

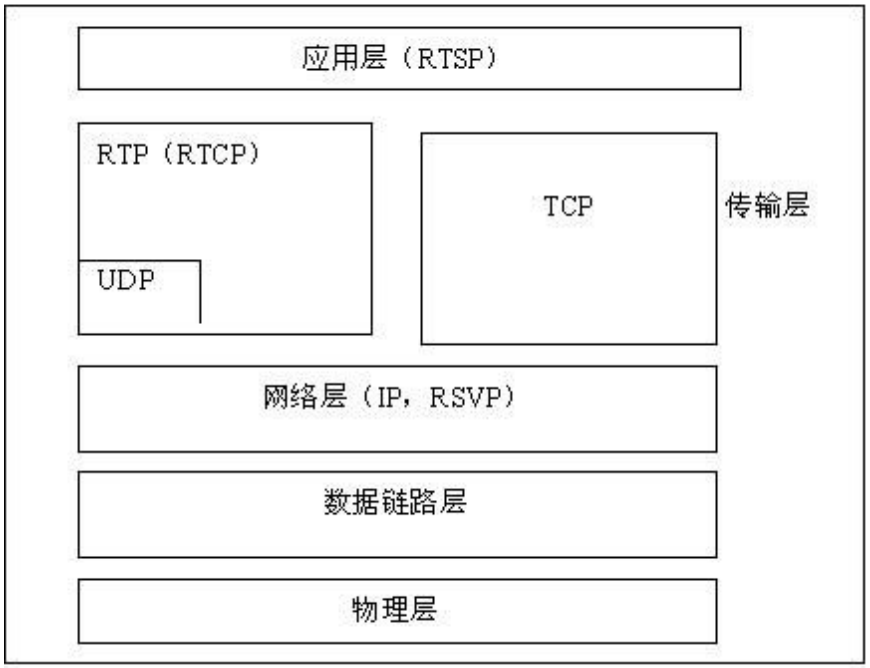
全名Real-time Transport Protocol，是一种实时的网络传输协议。RTP 本身并没有提供按时发送机制或其它服务质量（QoS）保证，它依赖于低层服务去实现这一过程。 RTP 并不保证传送或防止无序传送，也不确定底层网络的可靠性。 RTP 实行有序传送， RTP 中的序列号允许接收方重组发送方的包序列，同时序列号也能用于决定适当的包位置，例如：在视频解码中，就不需要顺序解码。个人理解是建立在传输层UDP协议上的一个中间层协议，面向非连接，通过包序列号和时间戳来保证有序的播放。但是数据可靠性不高，所以控制协议采用RTSP，走的是TCP协议。

RTP报文格式两部分组成：报头和有效载荷。RTP报头格式如图所示：

1	2	3	8	9	16bit
V	P	X	CSRC Count	M	Payload Type
Sequence number				Timestamp	
SSRC				CSRC (variable 0 - 15 items 32bits each)	

V：版本号;Version(2)，占2个bit，数值为2，二进制表示10
P：填充字段标识； Padding(0)，占1个 bit，数值为0，二进制表示0
X：扩展头标识；Extension(0)，占1个bit ，数值为0，二进制表示0
CSRC count(CC)：CSRC计数器，CSRC，贡献源，指的是不同步的源。在网络中，可能会有混合器将来自 不同地点的RTP流混合成一个RTP流以节省带宽, CSRC用来区分不同的源；占4个bit，数值为0，二进制表示为0000
M：标记一些重要的事件（由应用程序定义）；占1个bit
PT：净荷数据类型； Payload Type，占7个bit
SN：序列号，每个分组的序列号（初始值随机），用来检测分组的丢失并恢复分组的序列；在rtp包中占第2,3这2个字节。
TS：时间戳，反映RTP净荷中的第一个采样数据的采样时间。时间的粒度是净荷类型相关的。在rtp包中，占4,5,6,7这4个字节。
SSRC：同步源标识符，用于标识同步源。同步源指的是，例如，一段影片的音频和视频通过不同的RTP流传输，它们是同步的。每个同步源是负责发送RTP分组并在RTP中设置序列号和时间戳的实体。在rtp包中，占8,9,10,11这4个字节。

流媒体网络传输层级图



RTSP实时流协议本身不传递具体媒体数据，主要是起到端对端的会话和远程控制流的功能。RTSP的请求主要有DESCRIBE,SETUP,PLAY,PAUSE,TEARDOWN,OPTIONS等。RTSP的对话过程中SETUP可以确定RTP/RTCP使用的端口，PLAY/PAUSE/TEARDOWN可以开始或者停止RTP的发送，等等。

RTCP控制协议需要与RTP数据协议一起配合使用，当应用程序启动一个RTP会话时将同时占用两个端口，分别供RTP和RTCP使用。RTP本身并不能为按序传输数据包提供可靠的保证，也不提供流量控制和拥塞控制，这些都由RTCP来负责完成。通常RTCP会采用与RTP相同的分发机制，向会话中的所有成员周期性地发送控制信息，应用程序通过接收这些数据，从中获取会话参与者的相关资料，以及网络状况、分组丢失概率等反馈信息，从而能够对服务质量进行控制或者对网络状况进行诊断。

6、SDP分析

先说下SDP的作用就是在媒体会话中，传递媒体流信息，允许会话描述的接受者去参与会话。是由服务器发送给客户端的。

具体格式为：

（1）会话的名称和目的

（2）会话存活时间

（3）包含在会话中的媒体信息，包括：

媒体类型（vedio,audio,etc）

传输协议（RTP/UDP/IP,H.320,ETC）

媒体格式（H.261 video,MPEG video,etc）

多播或远端（单播）地址和端口

（4）为接收媒体而需的信息（addresses,ports,formats and so on）

（5）使用的宽带信息

（6）可信赖的接洽信息（Contact information）

这是在RTSP会话的DESCRIPTION流程传递的。

以下是实际的会话实例：

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=000001B0F5000001B509000001000000012000C888B0E0E0FA62D089028307
a=control:trackID=0//trackID＝0表示视频流用的是通道0

必要字段的描述

1．Version（必选）

v=0

SDP的版本号，不包括次版本号。

2．origion（必选）

o=<username> <session id> <version> <network type> <address type> <address>

o=<用户名> <session id> <会话版本> <网络类型> <地址类型> <地址>

“o=”项对会话的发起者进行了描述。

<username>是用户的登录名。如果主机不支持<username>，则为”－”。注意：<username>不能含空格。

<session id>：是一个数字串。在整个会话中，必须是唯一的。为了确保其唯一，建议使用NTP(Network Time Protocol) timestamp。

<version>：该会话公告的版本，供公告代理服务器检测同一会话的若干个公告哪个是最新公告。基本要求是会话数据修改后该版本值递增，建议用NTP时戳。

<network type>：网络类型，一般为”IN”，表示”internet”

<address type>：地址类型，一般为IP4

<address>：地址

3．Session Name（必选）

s=<session name> 会话名，在整个会话中有且只有一个”s=”。

4．Times（必选）， Repeat Timesand Time Zones

t=<start time> <stop time>

描述了会话的开始时间和结束时间。

<start time> 和<stop time> 为NTP时间，单位是秒。假如<stop time>为零表示过了<starttime>时间后会话一直持续。当<start time> 和<stop time>均为零时表示持久会话。建议start time和stop time不要设为0。因为不知道此会话的开始和结束时间，增加了调度（scheduling）的难度。

5．Media Announcements （必选）

m=<media> <port> <transport> <fmt list>

一个会话描述包括几个媒体描述。一个媒体描述以”m=”开始到下一个”m=”结束。

<media>：表示媒体类型。有"audio", "video", "application"（例白板信息）, "data"（不向用户显示的数据）和"control"（描述额外的控制通道）。

<port>：媒体流发往传输层的端口。取决于Connection Data规定的网络类型和接下来的传送层协议：对UDP为1024-65535；对于RTP为偶数。当分层编码流被发送到一个单播地址时，需要列出多个端口。方式如下：

m=<media> <port>/<number of ports> <transport> <fmt list>

对于RTP，偶数端口被用来传输数据，奇数端口用来传输RTCP包。例：

m=video 49170/2 RTP/AVP 31

端口49170和49171为第一对RTP/RTCP端口，49172和49173为第二对的端口。传输协议是RTP/AVP，媒体格式为31。

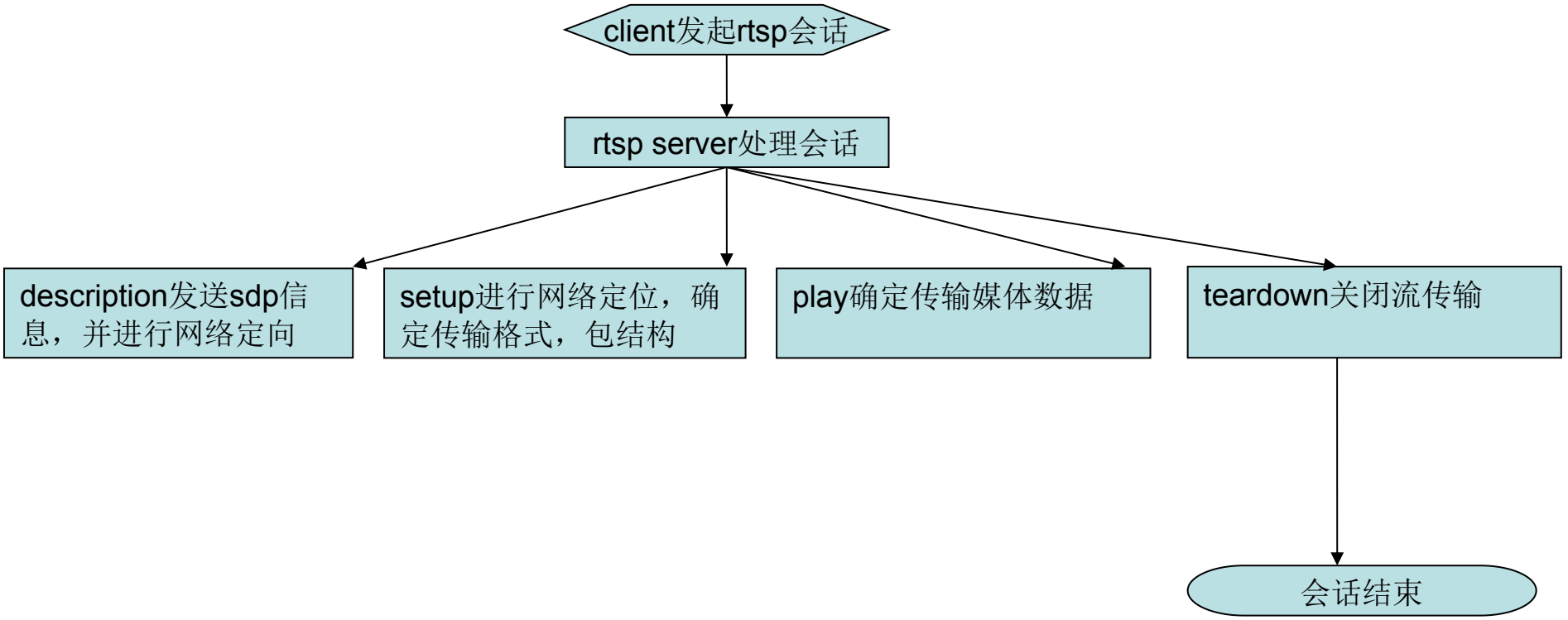
<transport>：传输协议，与c=行的地址类型有关。两种：RTP/AVP，表示Realtime Transport Protocol using the Audio/Video profile carried over UDP；UDP。
<fmt list>：媒体格式。对于音频和视频就是在RTP Audio/Video Profile定义的负载类型(payload type)。但第一个为缺省值，分为静态绑定和动态绑定：静态绑定即媒体编码方式与RTP负载类型有确定的一一对应关系，动态绑定即媒体编码方式（如时钟频率，音频信道数等）没有完全确定，需要进一步的属性说明（用rtpmap）。分别举例如下，静态绑定的例子：u_law的PCM编码单信道Audio，采样率8KHZ。在RTP Audio/Video profile中对应的payload type为0。即：

```
m=audio 49232 RTP/AVP 0
```

动态绑定的例子：16位线形编码，采样率为16KHZ，假如我们希望动态RTP/AVP 类型98表示此此流，写法如下：

```
m=video 49232 RTP/AVP 98
 a=rtpmap:98 L16/16000/2
```

事务流程图


client发起rtsp会话
rtsp server处理会话
description发送sdp信息，并进行网络定向
setup进行网络定位，确定传输格式，包结构
play确定传输媒体数据
teardown关闭流传输
会话结束

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