RTP(Real-time Transport Protocol)是用于Internet上针对多媒体数据流的一种传输协议

RTP被定义为在一对一或一对多的传输情况下工作

RTP通常使用UDP来传送数据。但RTP也可以在TCP或ATM等其他协议之上工作

RTCP(Real-time Transport Control Protocol) 提供流量控制和拥塞控制。

在RTP会话期间,各参与者周期性地传送RTCP包.

RTCP包中含有已发送的数据包的数量、丢失的数据包的数量等统计资料

服务器可以利用这些信息动态地改变传输速率，甚至改变有效载荷类型

开始一个RTP会话时将使用两个端口:一个给RTP一个给 RTCP

实时流协议RTSP(Real-time Streaming Protocol 一对多 它使用TCP或RTP完成数据传输

RTSP在体系结构上位于RTP和RTCP之上

HTTP请求由客户机发出，服务器作出响应；使用RTSP时，客户机和服务器都可以发出请求，即RTSP可以是双向的

RTSP协议是有状态的协议，而HTTP是无状态的协议 RTSP通过维护一个session来维护其状态的转换

RTSP协议的默认端口是554，默认的承载协议为TCP

控制流和数据流是分开的

rfc2326（rtsp）；rfc2327（sdp）

//音频是什么格式的。。 比如LATM还是ADTS的的

语法和运作跟HTTP 1.1类似

RTSP协议分析

http://blog.chinaunix.net/uid-25968088-id-3518900.html

RTSP 协议格式：

请求命令的格式为：

**METHOD** **URL** CR LF

**Field1:value** CR LF

Field2:value CR LF

......

Fieldn:value CR LF

CR LF

应答的格式为：

**RTSP**/major\_version.minor\_version **status** CR LF

Field1:value CR LF

Field2:value CR LF

......

Fieldn:value CR LF

CR LF

1.C->S:OPTION request //询问S 有哪些方法可用

1.S->C:OPTION response //S 回应信息中包括提供的所有可用方法

2.C->S:DESCRIBE request //要求得到S 提供的媒体初始化描述信息

2.S->C:DESCRIBE response //S 回应媒体初始化描述信息，主要是sdp

3.C->S:SETUP request //设置会话的属性，以及传输模式，提醒S 建立会

话

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

4.C->S:PLAY request //C 请求播放

4.S->C:PLAY response //S 回应该请求的信息

S->C:发送流媒体数据

5.C->S:TEARDOWN request //C 请求关闭会话

5.S->C:TEARDOWN response //S 回应该请求

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

第3和4步是必需的

命令 与 状态转换

DESCRIBE(协商用于描述媒体信息协议 ) -- > SETUP(协商传输用的协议) --> PLAY --> PLAY(scale=-32) --> PLAY(scale=1.0) --> GET\_PARAMETER --> TEARDOWN

rtsp交互包括options，describe，setup，play，teardown，pause 6个部分

DESCRIBE **rtsp://**118.122.89.27:**554**/**live**/ch10083121594790060557**.sdp**?playtype=1&boid=001&backupagent=118.122.89.27:554&clienttype=1&time=20100929182111+08&life=172800&ifpricereqsnd=1&vcdnid=001&userid=123&mediaid=ch10083121594790060557&ctype=2&TSTVTimeLife=1800&contname=&authid=0&UserLiveType=1&nodelevel=3 RTSP/1.0   
// 媒体URL   
**Accept: application/sdp   
//协商用于描述媒体信息协议**  
CSeq: 1

User-Agent:ZTE Ltd.co RTSP protocal verion 1.0 guid-2.1.1.100/B519D290-C0EC-EE35-7368-893BE4C0B347   
//User Agnet信息，显示是中兴的服务器，1.0的版本。如果是Helix服务器的话会有Helix 服务器的标识。   
x-NAT:2.1.1.100:20081   
//主要用于NAT穿透   
x-zmssRtxSdp: yes

RTSP/1.0 200 OK

//应答编号

Server: ZXUSS100 1.0

Cache-Control: no-cache

Content-Base: rtsp://118.122.89.27:554/live/ch10083121594790060557.sdp/

Content-Length: 320

//内容长度信息

Content-Type: application/sdp

**//描述内容信息所用的协议 使用sdp协议**

CSeq: 1

Date: Wed, 29 Sep 2010 10:20:38 GMT

Expires: Wed, 29 Sep 2010 10:20:38 GMT

**// SDP 描述信息 前面都是RTSP 的 field:value**

v=0

o=- 296874273 1 IN IP4 118.122.89.27

s=envivio

c=IN IP4 0.0.0.0

b=AS:1500

t=0 0

a=**range:clock**=20100929T095038.00Z-20100929T102038.00Z

m=video 5140 RTP/AVPF 33 96

a=**control:trackID**=2

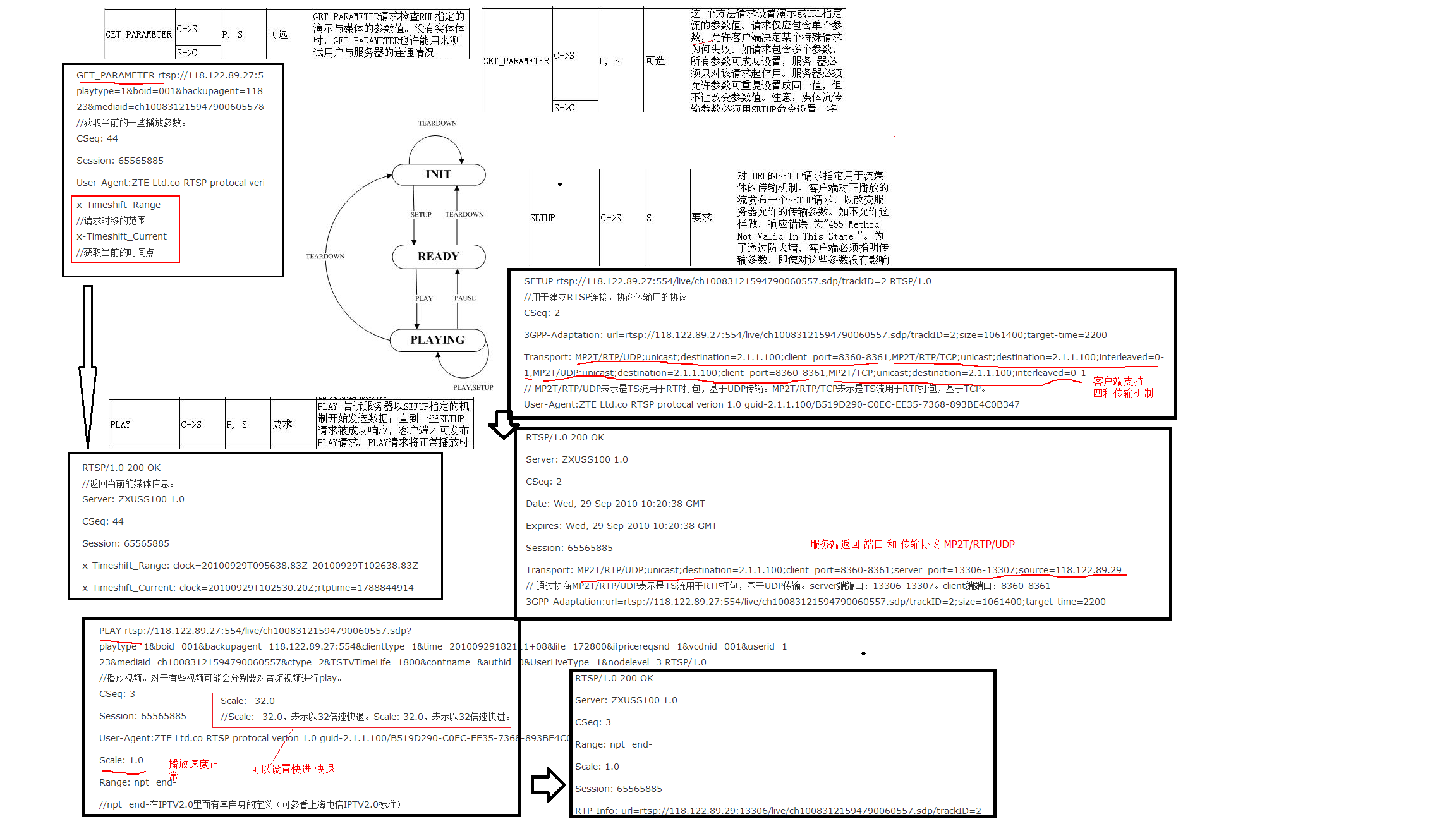
a=rtpmap:33 MP2T/90000

a=3GPP-Adaptation-Support:5

a=rtcp-fb:33 nack

a=rtpmap:96 rtx/90000

a=fmtp:96 apt=33;rtx-time=0



**RTSP状态码**

还有很多）

"200"; OK

"400" ; Bad Request

"403" ; Forbidden

"404" ; Not Found

linux rtsp服务器

linux webrtc服务器

VLC 的全名是 Video Lan Client,是一个开源的、跨平台的视频播放器

http://www.videolan.org/vlc/features.html

http://www.videolan.org/vlc/download-android.html

VLC做流媒体服务器的使用方法

<http://wenku.baidu.com/link?url=o_k4wWdOW6kToEk7CnL2z6_zf_W7giVvmwiUcxAYR7cwKZtkcFnrfK-GsRgBMoFr6cY5TznyiLoEDX3cG7iF6nv1xKQMC9YjXvPXM6WAhtG>

vlc的使用：作为服务器/客户端

1.使用 UDP 直接发送 H264+MP3(TS)/ udp://

2. VLC的RTP(基于UDP)发送 H264+MP3(TS)/ rtp://

3. RTSP方式是通过RTP进行流媒体数据的传输的，VLC的实现也是基于UDP的

<http://www.cnblogs.com/MikeZhang/archive/2012/09/09/vlcStreamingServer20120909.html>

[总结]FFMPEG视音频编解码零基础学习方法

http://blog.csdn.net/leixiaohua1020/article/details/15811977/

http://blog.csdn.net/leixiaohua1020/article/details/50535082

http://blog.csdn.net/leixiaohua1020/article/details/50535230

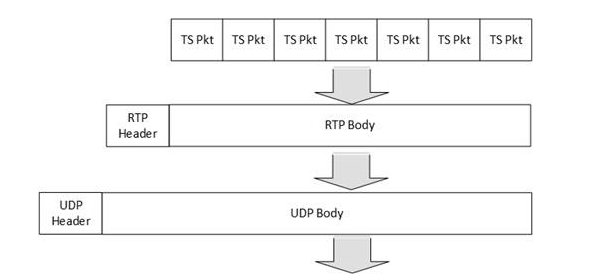
MPEG2-TS（Transport Stream“传输流”；又称TS、TP、MPEG-TS 或 M2T）实时传送的节目，比如实时广播的电视节目。

MPEG2-PS DVD节目中的MPEG2格式，确切地说是MPEG2-PS，全称是Program Stream（程序流）主要应用于存储的具有固定时长的节目，如DVD电影

video/mp2t

MP2T/RTP/TCP表示是TS流用于RTP打包,基于TCP

图中首先每7个MPEG-TS Packet打包为一个RTP，然后每个RTP再打包为一个UDP。其中打包RTP的方法就是在MPEG-TS数据前面加上RTP Header，而打包RTP的方法就是在RTP数据前面加上UDP Header。



spydroid-ipcamera

Android Open Source - Camera stream spydroid-ipcamera

[菜鸟怒读Spydroid源码（一）](http://blog.csdn.net/nmhszyn/article/details/49074027) http://blog.csdn.net/nmhszyn/article/details/49074027

[spydroid源码分析（一）：介绍spydroid每个包的大体功能](http://blog.csdn.net/xiaoliouc/article/details/8493161) http://blog.csdn.net/xiaoliouc/article/details/8493161

[spydroid源码分析（二）：spydroid运行流程](http://blog.csdn.net/xiaoliouc/article/details/8493207) http://blog.csdn.net/xiaoliouc/article/details/8493207

[spydroid源码分析（三）：rtp包头分析以及代码实现](http://blog.csdn.net/xiaoliouc/article/details/8493326) http://blog.csdn.net/xiaoliouc/article/details/8493326

spydroid

spydroid自身内置http服务器， 客户端可以通过在VLC等播放器中输入http://ip：8080/播放

spydroid自身内置rtsp服务器，客户端可以通过在VLC等播放器中输入rtsp://ip：8086/播放

[编解码学习笔记（五）：Mpeg系列——AAC音频](http://blog.csdn.net/flowingflying/article/details/5718594) http://blog.csdn.net/flowingflying/article/details/5718594

[H.264 sequence\_parameter\_sets成员值含义学习笔记](http://blog.csdn.net/leesphone/article/details/2159604) http://blog.csdn.net/leesphone/article/details/2159604

使用RTP传输H264的时候,需要用到sdp协议描述

其中有两项::Sequence Parameter Sets (SPS) **序列参数集** 和Picture Parameter Set (PPS) **图像参数集**

SDP中的H.264的**SPS和PPS串**，包含了初始化**H.264解码器**所需要的信息参数，包括编码所用的profile，level，图像的宽和高，deblock滤波器等

SPS: sequence parameter sets

01. **Profile\_idc** 它指的是**码流对应的profile.**

1.1 基线profile(Baseline profile)

遵循基线profile的码流应该遵循以下的约束:

a) 只有I和P切片存在

.....

c) 序列参数集(sps)的frame\_mbs\_only\_fag(之后介绍)应该等于1

d) 图象参数值(psp)的weighted\_pred\_flag和weighted\_bipred\_idc都应该为0

.....

**1**.2 主profile(Main profile)

主profile的码流应该遵循以下的约束:

a) 只有I,P,B切片存在.......

1.3 扩展profile(Extended profile)

.......

02. Constraint\_set0\_flag

当constrained\_set0\_flag值为1的时候，就说明码流**应该遵循基线profile(Baseline profile)的所有约束**.

当constrained\_set0\_flag值为0时，说明码流不一定要遵循基线profile的所有约束。

03. Constraint\_set1\_flag

04. Constraint\_set2\_flag

....

07. **Level\_id:** 12[Level 1.2] 它指的是码流对应的level级. 对于基线profile,主profile和扩展profile的相应的level级约束在H.264的文档中有表可查

....

MulticastSocket多播

单播unicast、广播broadcast和组播multicast

讲组播和普通UDP socket之间的区别在于**必须考虑TTL值**。

这时IP首部中取值 0- 255的一个字节。 它的含义为包**被丢弃前通过的路由数目**。

**每通过一个路由器，其TTL减少1**，有些路由器减少2或更多。当TTL值为0时包就被丢弃

创建：

**MulticastSocket**()throws SocketException:绑定匿名端口

MulticastSocket(int port)throws SocketException:绑定已知端口

MulticastSocket(SocketAddress address)throws IOException:

加入：

public void **joinGroup**(InetAddress address)throws IOException 加入**组播组**

退出：

**leaveGroup**(InetAddress address)

例子：

MulticastSocket ms =new **MulticastSocket**(4000); //新建组播socket

InetAddress ia = InetAddress.getByName("224.2.2.2");//组播地址

ms.**joinGroup**(ia); //加入主播组

byte[] buffer=new byte[512]; //发送数据包

byte[] buf = "Hello,This is a member of multicast!".getBytes();

DategramPacket packet = new **DatagramPacket**( buf, buf.length, ia, 4000);

ms.**send**(packet);//接收数据包

DatagramPacket dp =new **DatagramPacket**(buffer , buffer.length);

ms.**receive**(dp);

String s=new String(dp.getData()).trim(); // 该方法返回一个复制该字符串的开头和结尾的白色空格去掉

System.out.println(s);

ms.leaveGroup(address);

ms.close()

live555

live555 ubuntu下面验证服务端和客户端

http://blog.sina.com.cn/s/blog\_80be40500101evtd.html

[**RTSP监控摄像头web端监控技术**](http://www.cuplayer.com/rtsp/) **http://www.cuplayer.com/player/PlayerCode/live555/**

live555到android上，借助live555里的代码：从本地读取mpg文件推送到darwin，改成从实时流推送到darwin服务器，大致思路是派生一个h264videoSource，从android录制获取流，通过pipe管道发送流，把该流推送到darwin。

网页 android推送实时视频流到darwin流媒体服务器的思路

live555 移植 <http://blog.csdn.net/baby313/article/details/7289489>

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

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。

https://github.com/spex66/RTSP-Camera-for-Android/issues/1 ，里面有人成功过实现把h264流推送到darwin流媒体服务器

rtspcamera 开源项目

spydroid开源项目

流媒体服务器

Helix Darwin

AAC（Advanced Audio Coding），中文称为“高级音频编码”，出现于1997年， 基于 MPEG-2的音频编码技术。

MPEG-4标准 出现后，AAC 重新集成了其特性，加入了SBR技术和PS技术，为了区别于传统的 MPEG-2 AAC 又称为 MPEG-4 AAC

有损数据压缩

ISO/IEC 13818-7（MPEG-2第7部）, ISO/IEC 14496-3（MPEG-4第3部）

主要扩展名有三种：

AAC - 使用MPEG-2 Audio Transport Stream( ADTS，参见MPEG-2 )容器，区别于使用MPEG-4容器的MP4/M4A格式，属于传统的AAC编码（FAAC默认的封装，但FAAC亦可输出 MPEG-4 封装的AAC）

MP4 - 使用了MPEG-4 Part 14（第14部分）的简化版 即3GPP Media Release 6 Basic (3gp6，参见3GP ) 进行封装的AAC编码（Nero AAC 编码器仅能输出MPEG-4封装的AAC）；

M4A - 为了区别纯音频MP4文件和包含视频的MP4文件而由苹果(Apple)公司使用的扩展名，Apple iTunes 对纯音频MP4文件采用了".M4A"命名。M4A的本质和音频MP4相同，故音频MP4文件亦可直接更改扩展名为M4A。

9 种规格

MPEG-2 AAC LC 低复杂度规格（Low Complexity）

MPEG-2 AAC Main 主规格

MPEG-2 AAC SSR 可变采样率规格（Scaleable Sample Rate）

MPEG-4 AAC LC 低复杂度规格（Low Complexity），现在的手机比较常见的 MP4 文件中的音频部份就包括了该规格音频文件

MPEG-4 AAC Main 主规格

MPEG-4 AAC SSR 可变采样率规格（Scaleable Sample Rate）

MPEG-4 AAC LTP 长时期预测规格（Long Term Predicition）

MPEG-4 AAC LD 低延迟规格（Low Delay）

MPEG-4 AAC HE 高效率规格（High Efficiency）

上述的规格中，主规格（Main）包含了除增益控制之外的全部功能，其音质最好，而低复杂度规格（LC）则是比较简单，没有了增益控制，但提高了编码效率，至‘SSR’对‘LC’规格大体是相同，但是多了增益的控制功能，另外，MPEG-4 AAC/LTP/LD/HE，都是用在低比特率下编码，特别是‘HE’是有 Nero ACC 编码器支持，是近来常用的一种编码器，不过通常来说，Main 规格和 LC 规格的音质相差不大，因此目前使用最多的 AAC 规格多数是‘LC’规格，因为要考虑手机目前的存储器能力未达合理水准。

以上的这两个都是基于局域网的，那么外网怎么解决了？下面我提供几个我自己的灵活方案：

1、vpn，全部都在同一个vpn网络上，不就同一个局域网了吗。

2、花生壳域名映射，现在很多路由器都支持，如果不支持就搞台电脑吧。

3、路由IP映射。

1、rstp协议，网络上有个开源项目，基于Android，且这个项目也是一个服务端，里面也集成了http访问页面，可以通过http或者rstp直连手机：

<https://github.com/fyhertz/spydroid-ipcamera>

2、udp协议，网络上流传的基于udp进行局域网直播方案，采用的是广播：

<https://github.com/bytestar/android-h264-stream-demo>

[http://blog.csdn.net/mirkerson/article/details/38520175](http://images.tuxi.com.cn/blog.csdn.net/mirkerson/article/details/38520175)

udp协议，网络上流传的基于udp进行局域网直播方案，采用的是广播：

[VLC-Android编译与运行](http://blog.csdn.net/dtryl/article/details/50681612)

<git://git.videolan.org/vlc-ports/android.git>

android源码和apk下载

http://www.videolan.org/vlc/download-android.html 或者直接到http://get.videolan.org/vlc-android/2.0.6/ 有不同芯片apk和源码

<http://blog.csdn.net/dtryl/article/details/50681612> 编译

ubuntu 上安装vlc

vlc+ffmpeg+live555\_install

[http://www.doc88.com/p-673129826059.html](%20http:/www.doc88.com/p-673129826059.html)

VLC 播放器

1.live555的效率有些低，建议不使用...

2.编码后，需将码流放在一个缓冲区，然后rtsp再从缓冲区获取再打包传输

3.vlc设置里将延时改成小值

VLC缓存设置为300ms

M-JPEG的主要缺点是，压缩效率是低的，是基于M-JPEG压缩算法的每帧的图像的内容上，而不是基于相邻帧之间的差异，以被压缩，因此而引起的特定的冗余信息被重复存储

不用帧间压缩(mpeg h264等用)

Mjpeg是Motion JPEG 的简称。它的原理是把视频镜头拍成的视频分解成一张张分离的jpg数据发送到客户端。当客户端不断显示图片，即可形成相应的图像

大部分低端的网络摄像头都支持的MJPG[**协议**](http://www.edenw.com/tech/net/col/),较为高级的摄像头支持是H.264.

MJPG的优点是实现简单，客户端实现简单，不会出现马赛克的情况。缺点就是占带宽比较大，因为是一帧一帧按图像来传输的.

MJPG可以在多种传输[**协议**](http://www.edenw.com/tech/net/col/)上传输，比如TCP/UDP，最常见是在HTTP上采用传输。大部分的摄像头也是采用HTTP+MJPG的传输形式的。

mjpg的在http的mime type是"x-mixed-replace"

如果ipcam返回200，表示已经接收的请求，并在返回的头里指明边界字符串,这是在context type的boundary子属性来指明的。

这里的边界字符串是ThisRandomString.

　　GET /xxx.jpg HTTP/1.1

　　HTTP/1.0 200 OK Server: Digital Video Technology PUSH1 Expires: 0 Pragma: no-cache

　　Content-Type: multipart/x-mixed-replace;boundary=ThisRandomString

　　--ThisRandomString

　　Content-Type: image/jpeg

　　Content-Length: 6280

　　...(jpeg数据)

　　--ThisRandomString

[MJPEG协议入门介绍](http://blog.chinaunix.net/uid-22670933-id-1771591.html)

<http://blog.chinaunix.net/uid-22670933-id-1771591.html>

JPEG（英文全称：Joint　Photographic　Experts　Group）压缩技术可以说是所有图像压缩技术的基础。

它适合静态图像的压缩，直接处理整个画面，压缩倍数为20-80倍，分辨率没有选择的余地。所以要等到整个压缩档案传输完成才开始进行解压缩成影像画面，而这样的方式造成传输一个高解析画面时须耗时数十秒甚至数分钟。

高清网络[摄像机](http://dv.zol.com.cn/)视频编码标准主要有 MJPEG和H.264两种。

MJPEG(Motion JPEG)是在JPEG基础发展起来的动态图像压缩技术，它只单独的对某一帧进行压缩，而基本不考虑视频流中不同帧之间的变化 但它的缺陷也非常明显，其一：丢帧现象严重、实时性差，在保证每路都必须是高清晰的前提下，很难完成实时压缩。其二：压缩效率低，存储占用空间较大。由于它在运动性、带宽占用以及存储空间占用方面均有致命缺陷，高清网络摄像机依然以 H.264为主导。

后来又出现了多层式JPEG（ML-JPEG）压缩技术，它采取渐层式技术，先传输低解析的图档，然后再补送更细节的压缩资料，使画面品质改善。这种方式所需的时间虽然与原先的方式一样。但由于可以先看到画面，所以使用者会觉得这种方式较好。

OpenGL ES --> ANativeWindow http://cn.khronos.org/

:OpenMAX DL - 驱动层OpenMAX IL - 整合层OpenMAX AL

OpenMAX 是一套嵌入式的多媒体应用标准, 从驱动层到应用层都有详述的规定

http://blog.csdn.net/bonderwu/article/details/6149332 [OpenMax多媒体引擎](http://blog.csdn.net/bonderwu/article/details/6149332)

### [Android MediaCodec/NdkMediaCodec](http://www.baidu.com/link?url=n78e7DvyCJvpHzt6bsoGfbHu8WHJsFIbacwI6umTzHA7WIuXs20k7EcMCaZ9fzZDCrMqxMhSQDmpPFBdxEHY_a)

./android-21/arch-arm/usr/include/media/NdkMediaCodec.h

使用OpenGL ES 编解码

http://blog.csdn.net/augusdi/article/details/12574679