1#yasea分析

虚拟机开机会自动启动SRS服务器。

推流&播放地址：rtmp://192.168.1.xxx/live/livestream，测试推流成功，反复断开重连测试很稳定。

It encodes YUV and PCM data from camera and microphone to H.264/AAC, encapsulates in FLV and transmits over RTMP.

- [x] Android mini API 16.
- [x] H.264/AAC hard encoding.
- [x] H.264 soft encoding.
- [x] RTMP streaming with state callback handler.
- [x] Portrait and landscape dynamic orientation.
- [x] Front and back cameras hot switch.
- [x] Recording to MP4 while streaming.
- [x] Beautiful filters with GPUImage.
- [x] Acoustic echo cancellation and automatic gain control support.

- srs-sea
- SimpleRtmp
- MagicCamera
- x264
- mp4parser

```
private SrsPublisher mPublisher;
```

```
mPublisher = new SrsPublisher((SrsCameraView) findViewById(R.id.glsurfaceview_camera));
mPublisher.setEncodeHandler(new SrsEncodeHandler(this));
mPublisher.setRtmpHandler(new RtmpHandler(this));
mPublisher.setRecordHandler(new SrsRecordHandler(this));
mPublisher.setPreviewResolution(640, 480);
mPublisher.setOutputResolution(720, 1280);
mPublisher.setVideoHDMode();
mPublisher.startCamera();
```

一、采集

http://www.jianshu.com/p/7684cc38a115

1. Camera视频数据源采集

Android中的摄像头Camera提供了回调接口来获取每一帧数据：

setPreviewCallbackWithBuffer方法，同样设置回调接口：PreviewCallback，不过还需要一个方法配合使用：addCallbackBuffer，这个方法接受一个byte数组。

2. 音频数据源采集

AudioRecord 的工作流程如下

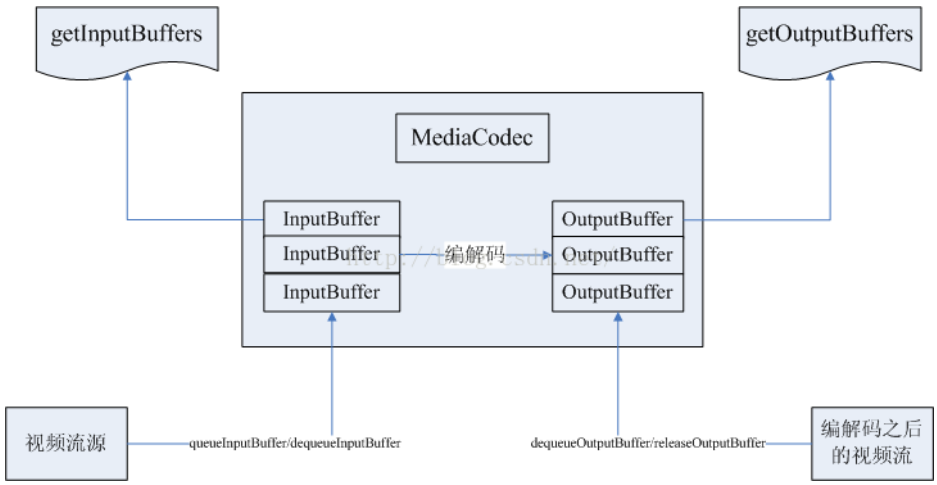
（1） 配置参数，初始化内部的音频缓冲区

（2） 开始采集

（3） 需要一个线程，不断地从 AudioRecord 的缓冲区将音频数据“读”出来，注意，这个过程一定要及时，否则就会出现“overrun”的错误，该错误在音频开发中比较常见，意味着应用层没有及时地“取走”音频数据，导致内部的音频缓冲区溢出。

（4） 停止采集，释放资源

3. 视频编码



对应的方法主要为：

getInputBuffers：获取需要编码数据的输入流队列，返回的是一个ByteBuffer数组

queueInputBuffer：输入流入队列

dequeueInputBuffer：从输入流队列中取数据进行编码操作

getOutputBuffers：获取编解码之后的数据输出流队列，返回的是一个ByteBuffer数组

dequeueOutputBuffer：从输出队列中取出编码操作之后的数据

releaseOutputBuffer：处理完成，释放ByteBuffer数据

接下来分析一下具体的流程：

视频流有一个输入队列，和输出队列，分别对应getInputBuffers和getOutputBuffers这两个方法获取这个队列，然后对于输入流这端有两个方法一个是queueInputBuffers是将视频流入队列，dequeueInputBuffer是从输入流队列中取出数据进行编解码操作，在输出端这边有一个dequeueOutputBuffer方法从输出队列中获取视频数据，releaseOutputBuffers方法将处理完的输出视频流数据ByteBuffer放回视频流输出队列中，再次循环使用。这样视频流输入端和输出端分别对应一个ByteBuffer队列，这些ByteBuffer可以重复使用，在处理完数据之后再放回去即可。

4. 音频编码

二、RTMP推流

1. 握手

在rtmp连接建立后,服务端与客户端需要通过3次交换报文完成握手,握手其他的协议不同,是由三个静态大小的块,而不是可变大小的块组成的,客户端与服务器发送相同的三个chunk,客户端发送c0,c1,c2,服务端发送s0,s1,s2。

通过handshake方法来进行握手协议，接下来开启了两个线程，这两个线程是用来进行读写操作的，读是读取服务器返回的指令，写是向服务器发送指令

2. 建立网络连接

客户端发送命令消息中的“连接”(connect)到服务器，请求与一个服务应用实例建立连接。

3. 建立网络流

创建完网络连接之后就可以创建网络流了

客户端发送命令消息中releaseStream命令到服务器端

客户端发送命令消息中FCPublish命令到服务器端

客户端发送命令消息中的“创建流”（createStream）命令到服务器端。

服务器端接收到“创建流”命令后，发送命令消息中的“结果”(_result)，通知客户端流的状态。

解析服务器返回的消息会得到一个stream ID, 这个ID也就是以后和服务器通信的 message stream ID, 一般返回的是1，不固定。

4. 推流命令

推流准备工作的最后一步是 Publish Stream，即向服务器发一个publish命令，这个命令的message stream ID 就是上面 create stream 之后服务器返回的stream ID，发完这个命令一般不用等待服务器返回的回应，直接下一步发送音视频数据

5. 音视频推流

publishAudioData和publishVideoData

6. 音视频同步问题

播放速度标准量的的选择一般来说有以下三种：

将视频同步到音频上，就是以音频的播放速度为基准来同步视频。视频比音频播放慢了，加快其播放速度；快了，则延迟播放。

将音频同步到视频上，就是以视频的播放速度为基准来同步音频。

将视频和音频同步外部的时钟上，选择一个外部时钟为基准，视频和音频的播放速度都以该时钟为标准。所以只要我们表示上正确的时间戳（dts），这样视频播放器就会根据这个这个时间戳去做音视频同步。