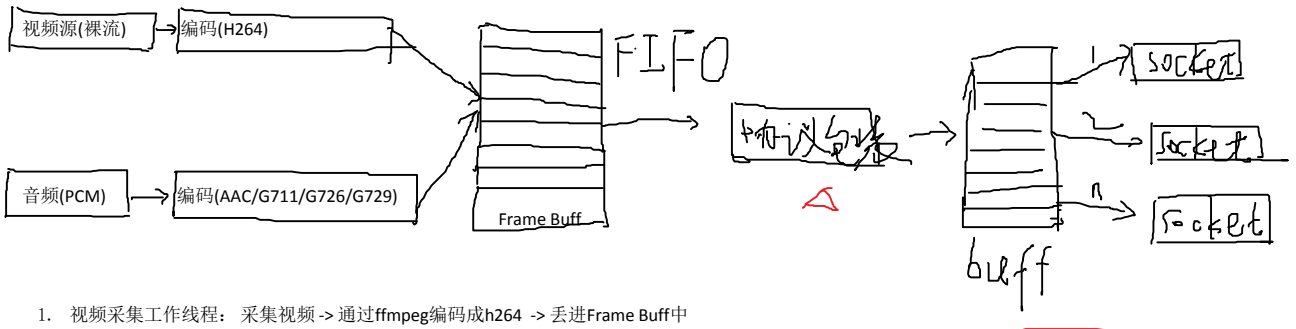

1. 视频采集工作线程： 采集视频 -> 通过ffmpeg编码成h264 -> 丢进Frame Buff中
2. 音频采集工作线程： 采集音步-> 通过faac/faad编码成aac -> 丢进Frame Buff中
3. Frame Buff: 模拟视频及音频的采集顺序(物理顺序）--》 转成逻辑关系， 它是先进先出FrameBuff 中的Frame 可以封装成一个类(实现FIFO)

4. 网络打包线程：
   a. 循环从Frame Buff中取出Frame
   b. 进行协议封装（rtmp, rtsp, rtp, http, hls, ts)
      i）视频H264：加上相关的Nal头，sps&&pps, 时间戳
      ii) 音频aac: 根把协议进行协议头的打包（采集率等
   c. 封装好的Frame, 扔进发送buff中， 此buff如果做实时播放只存放固定大小的帧，来了新的帧要丢掉旧的帧。 如果做回放，可以等待取走再丢进新包（卡住读文件的进度）

5. 服务器接受到socket连接 socket1, socket2, socket3
   （sock服务器的实现 ）
6. 每个socket从发送buff拷贝数据 发送出去（ 同一份视频，进行多路客户端分发）

网络打包是最难的部分（主要是协议实现）
rtmp: adobe
rtsp: 实时点播用的多
ts流： 这个带有时间戳，可以很好的同步
hls流： 苹果 （ 这个你首要关注一下）

1． 打包可以用ffmpeg, 可以在不同平台下编译(window/linux/mac)
2． 显示的渲染的话最好的可以用opengl ，现在手机都自带，各个平台也都支持（现在学习的难度已经比较低了）

这程架构方式，服务器可以满足不同的语言来写