**问题背景：**

视频监控不仅仅要上云，还要支持多端播放，这一块在移动端APP和PC端难度是不大的，只要能实现一个简单的播放器内核引擎，然后交叉编译为不同端的SDK就可以。随着移动互联网的发展，特别是微信生态的完善。很多客户都在做轻量的微信小程序，所以支持在网页上和微信小程序播放点直播视频就很有必要了。

微信小程序上能实现视频点直播，好处是轻量不用装APP，其次容易分享，随时随地能打开，也容易和公众号打通。但是实际上这块，浏览器上支持视频点直播还是很弱，不仅仅要考虑视频编码类型，视频延时，首屏等常见问题，还有很多一些底层接口的限制，也许你在APP上播放啥问题都没有，但是挪到微信小程序播放问题就很多，这块都需要服务端和前端一起配合才能兼容实现比较好的效果。

**解决方案：**

1. **前端直播方案**

为了在微信小程序上播放视频，我们调研了一下微信小程序的媒体组件，文档接口见

https://developers.weixin.qq.com/miniprogram/dev/component/live-player.html

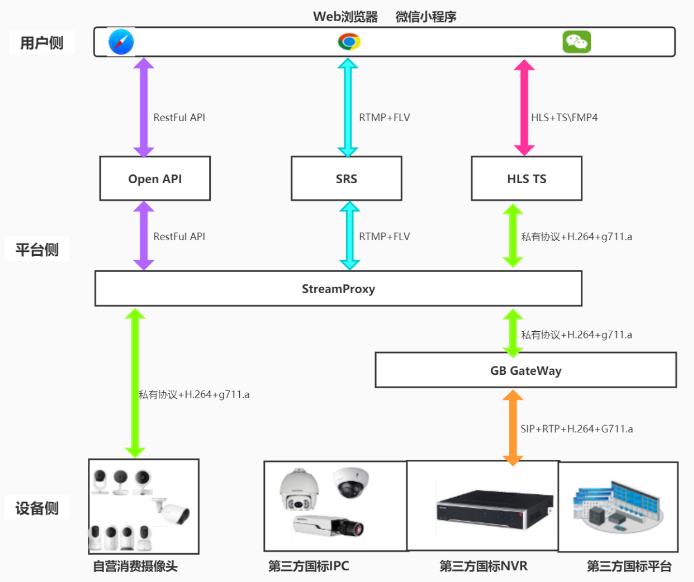
# 发现关于视频这块主要是live-player和live-pusher组件，其中live-player组件主要支持rtmp和flv拉流方式，推流live-pusher组件主要支持rtmp推流，所以直播我们选用live-player组件，选用rtmp-flv分发协议，目的是为了降低延时，一般的rtmp延时能控制在1-3秒，基本能满足使用视频监控中的场景要求。

1. **前端点播方案**

点播其实不关心延时，采用H5中video标签对hls-ts的支持即可，因为HLS兼容性很好，在PC端、Ios和Android，支持都非常好，也是目前唯一跨平台支持比较好的拉流协议，美中不足就是直播延时比较大，延时大的原因是因为这套协议本质采用了小文件片段的分发方式，一个小文件片段基本在2秒左右，这样HLS延时一般下来都在10s+,做的好的也在5s+，综合来看HLS作为直播分发协议不太合理（除非直播时不关心延时），但是用来做点播还是非常好的，省很多兼容处理的工作量。

1. **服务端方案**

服务端一般都要支持多协议多客户端的分发，这里画个示意图来说明下服务端的简单架构，hls-ts点播部分我们采用了自研形式，rtmp-flv采用了开源流媒体SRS和一些配套服务即可。



模块说明：

1. **Open API**，提供一些Restful API接口，允许用户通过该接口创建点直播任务，获取HLS拉流和RTMP拉流URL，任务一旦创建后就唤醒设备，设备通过私有协议或者国标协议，将码流推到平台侧；
2. **GB GateWay**，国标网关，第三方IPC或者平台主要通过该网关收流和对接，其中对接协议主要采用监控协议GB28181协议；
3. **StreamProxy**，流媒体代理，则主要完成私有码流到标准码流的转换工作，其中包含音频G711到AAC的转码，或者H.265到H.264的转码工作等；
4. **SRS**，主要用来接收StreamProxy的推流和允许用户侧的微信小程序过来拉流，实现RTMP-FLV的分发；
5. **HLSTS**，则通过私有协议从StreamProxy拉流，进行HLSTS切片，同时以HLS协议分发给微信小程序或者浏览器；

**实践遇到的坑：**

**问题1：**安卓微信小程序每次点播云存储视频看回放时，只能看10几秒的视频，然后又继续从头开始播放，不会从上次播放位置进行播放；

**问题原因：**创建点播任务时，服务端Open API会返回一个拉流会话ID，服务端就是通过这个拉流会话来确定当前客户端的播放进度的，如果后续请求的拉流会话ID总是变化，服务端就觉得这是新的拉流，就会重新切片重新分发导致客户端每次只能播放3个ts的视频，然后就从头开始继续播放导致点播视频不能连续持续播放；

**解决方案：**客户端拉流时，主要在HTTP的头里面把服务端第一次返回的拉流会话ID号固定下来即可，如下图所示：



**问题2：**小程序利用RTMP直播时，没有办法听声音，当前画面只有视频无声音；

**问题原因：**由于国标网关接的第三方国标设备音频大部分都是采用的G711.a的编码方式，而微信小程序live-player组件只支持AAC音频，所以导致解码音频失败；

**解决方案：**StreamProxy代理服务在从国标网关GB GateWay拉流后，需要做一次音频转码动作，需要将音频g711.a的音频转码为AAC的，这里转码库我们采用开源的fffmpeg转码；

**问题3：**小程序利用HLS回放时，跳秒严重，播放的TS文件不能连续播放；

**问题原因：**经过分析后发现微信小程序对TS文件的时间戳要求比较严格，各个文件的时间戳要连续单调递增，不能忽大忽小，这种时间戳打的不规范都有可能导致跳秒现象出现；

**解决方案：**代理在同一路拉流会话中，各个TS文件的时间戳要严格按照采样和帧率打时间戳，保证各个TS文件的时间戳要单调递增，不能出现跳变情况，特别对于TS文件末尾和下一个文件的开头部分时间戳情况；

**问题4：**小程序利用RTMP直播时，音画不同步，声音滞后于视频1秒左右；

**问题原因：**采用音频转码方案后，肯定有一定的延时，这时从国标ps流转为flv流后，需要注意时间戳的填写，这里主要是一帧g711.a的音频时间戳增量和一帧aac的时间戳增量不一样，我们要完成时间基的转换操作；

**解决方案：**转码后要对音频时间戳在服务端处理好，注意采样率、时间基的转换动作，有些转码器允许将原始时间戳带进去，然后填写好采样率，吐出来的码流就回调出新编码格式的时间戳，有些转码器认为时间戳跟转码本身没关系，需要上层业务做好时间戳的转换操作；

**问题5：**小程序利用RTMP直播时，视频没出来黑屏状态，先听到音频；

**问题原因：**这个主要是开源方案SRS服务端的配置问题，为了减小首屏时间，服务端可以不缓存GOP，这样客户端拉流过去的视频可能第一帧不是IDR帧导致解码失败，所以先听到声音没有出现画面，其实服务端只要缓存GOP即可，保证客户端拉流时，第一帧总是IDR帧即可；

**解决方案：**服务端缓存GOP，保证分发第一帧为IDR帧，但同时要注意服务端GOP队列长度对延时的影响；

**问题6：**小程序利用RTMP直播和HLS点播时，如何做防盗链？

**解决方案：**为了保证拉流URl的时效性和对客户端身份的校验，OpenAPI在客户创建拉流会话任务时一般都会在返回的拉流URL里面填上token信息，然后客户端拉流时，我们利用HTTP 回调机制，统一到Open API做身份鉴权，鉴权通过则允许拉流，鉴权失败则返回失败，不允许客户端下行拉流，当然也可以对一些IP开白名单；

**问题7：**直播播放时，经常卡顿，转圈不能流畅播放；

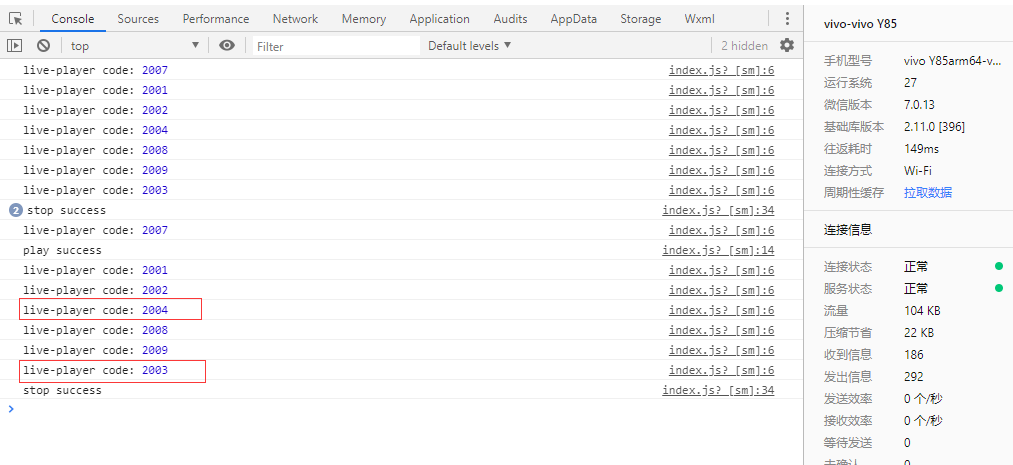
问题原因：由于下行网络等原因或者设备推流原因，导致客户端在播放视频时，暂时没有解码数据导致，为了解决卡顿问题我们可以在播放器端设置播放缓存空间的大小，当下行网络抖动时，可以利用缓存解决卡顿问题，同时live-player接口给了回调状态码可以观察卡顿情况；

**解决方案：**

1. 将客户端播放器组件的live-player组件的播放缓存加大，当然服务端也可以加点GOP缓存应对设备端推流网络问题，但是带来的负面效应就是延时会稍微加大，不能设置过大过小；
2. 也可以调整推流端的上行码率来解决，一般的摄像头都支持自动码率，也可以改变有些设备的编码参数来调整，比如帧率，分辨率，I帧间隔等，服务端可以做一定的丢帧处理也可以应对下行网络差的问题；

****

****

****

****

**问题8：**对接过程出现问题时，怎么定位问题和排查思路是什么？

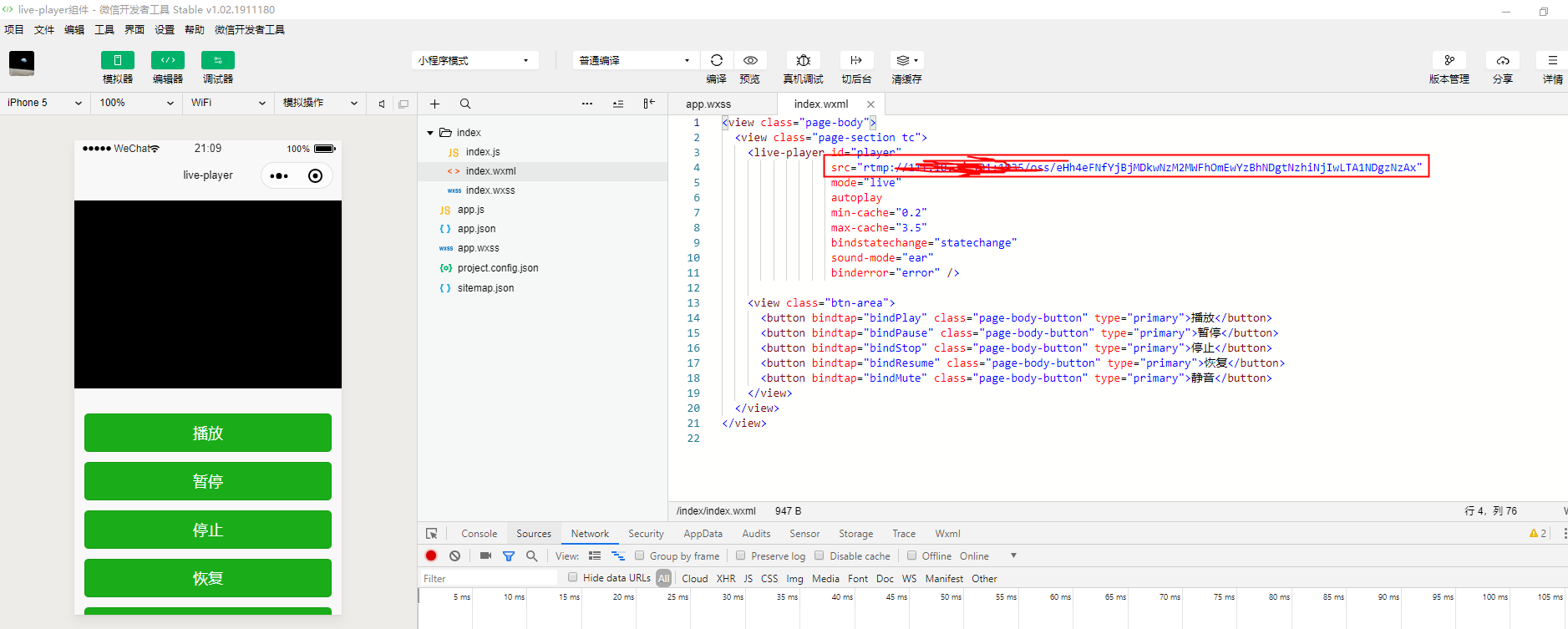
**解决方案：**

1. 以前写过一篇文章，讲过排查问题思路可参照，基本就是根据客户端的回调信息或者系统错误，进行分段排查、不同服务端写码流文件、抓包以及做对比试验；
2. 排查播放器的问题，可以用同样的URl在其它端进行播放，比如VLC，ffplayer和flash,其它端播放没问题基本就是前端小程序的问题；
3. 其它播放器有问题，根据回调状态错误码，进行服务端抓包以及服务端写文件形式排查，基本可以将问题定位到哪一个模块上；
4. 服务端加调试日志等进行进一步大判断和分析，用专业工具分析写下来的码流，排查问题是时间戳问题，还是编解码问题还是服务自己内部处理问题；

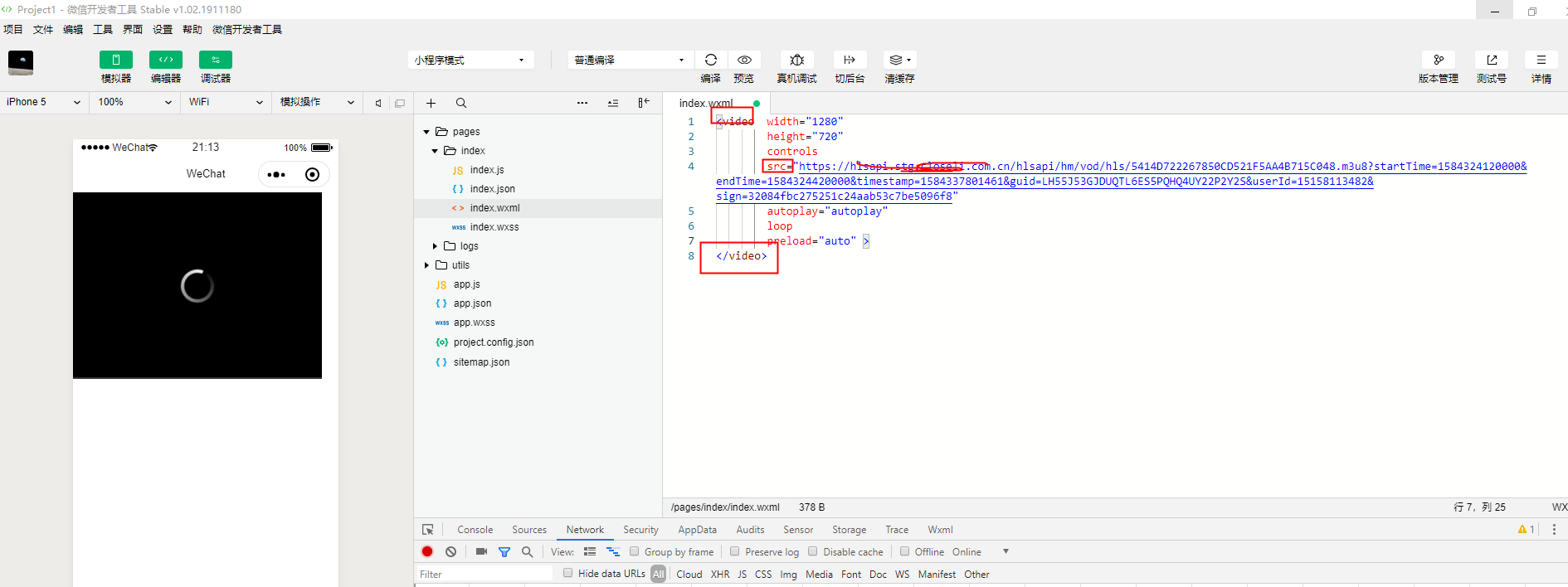
**Demo代码：**

**基于微信小程序Live-player组件：**

**这个要绑定微信开发者账号，最后上架要提交前端代码审核才可以发布小程序：**

****

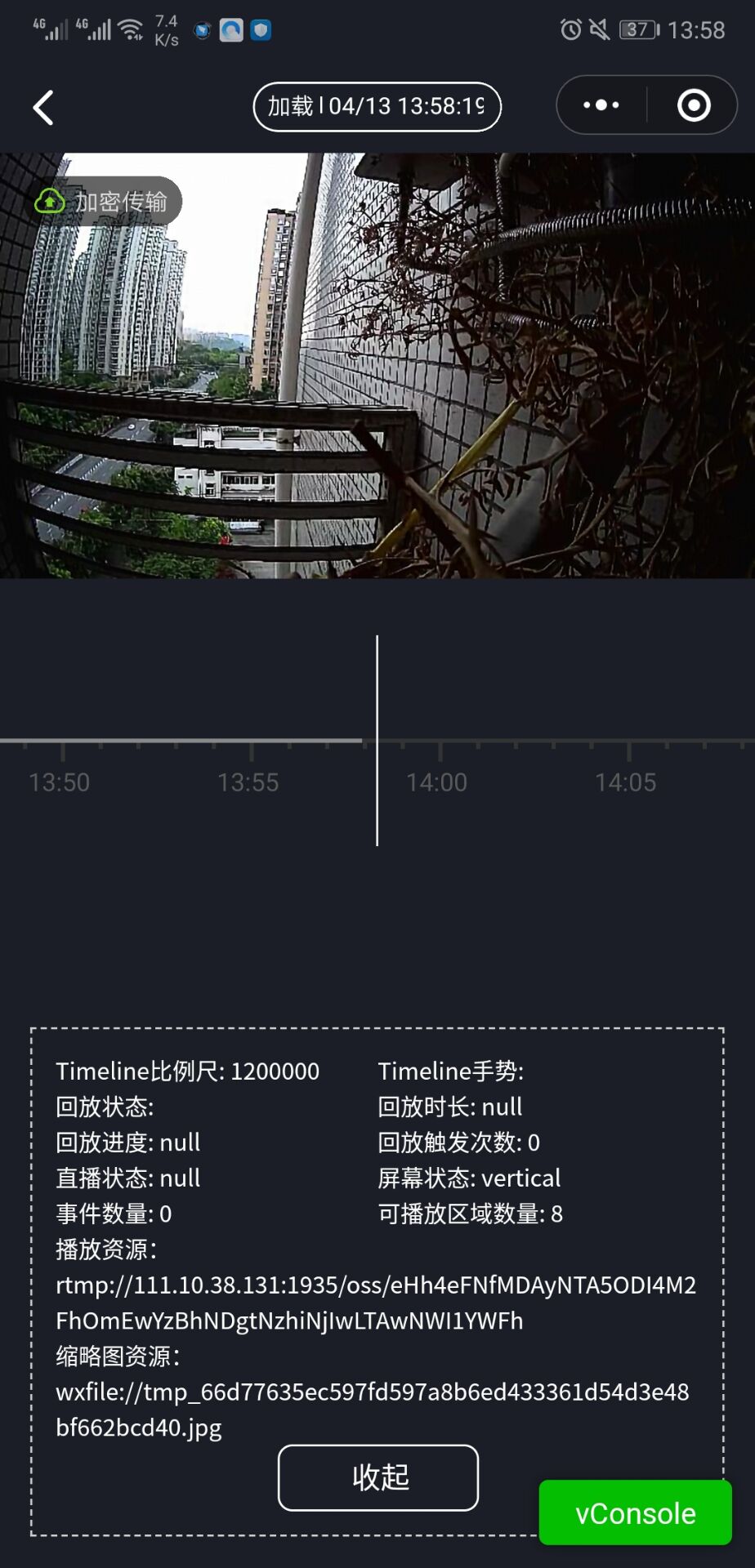
**基于H5的Video标签形式，这个也可以直接在PC的相关插件上调试：**

****

**最终效果：**

上面代码只是为了验证下服务端功能，前端我们除了增加点直播功能，还增加了随机拖动，花式播放、缩略图和timeline等功能​。​

**Rtmp直播：**

****

**HLS点播：**



**本文总结：**

本文主要讲了下基于微信小程序的点直播方案，其中点播基于H5 Video标签用了HLS-TS方案，直播基于微信live-player媒体组件用了RTMP-FLV方案。同时给了服务端方案示意和前端Demo示例代码。对接过程中也有很多细小问题，本文例举了典型的7个问题并一一给出了这些问题的解决方案，当然还有很多细小问题你可以后台继续咨询我，相互交流学习。