

七牛云直播技术分享

pili@qiniu.com

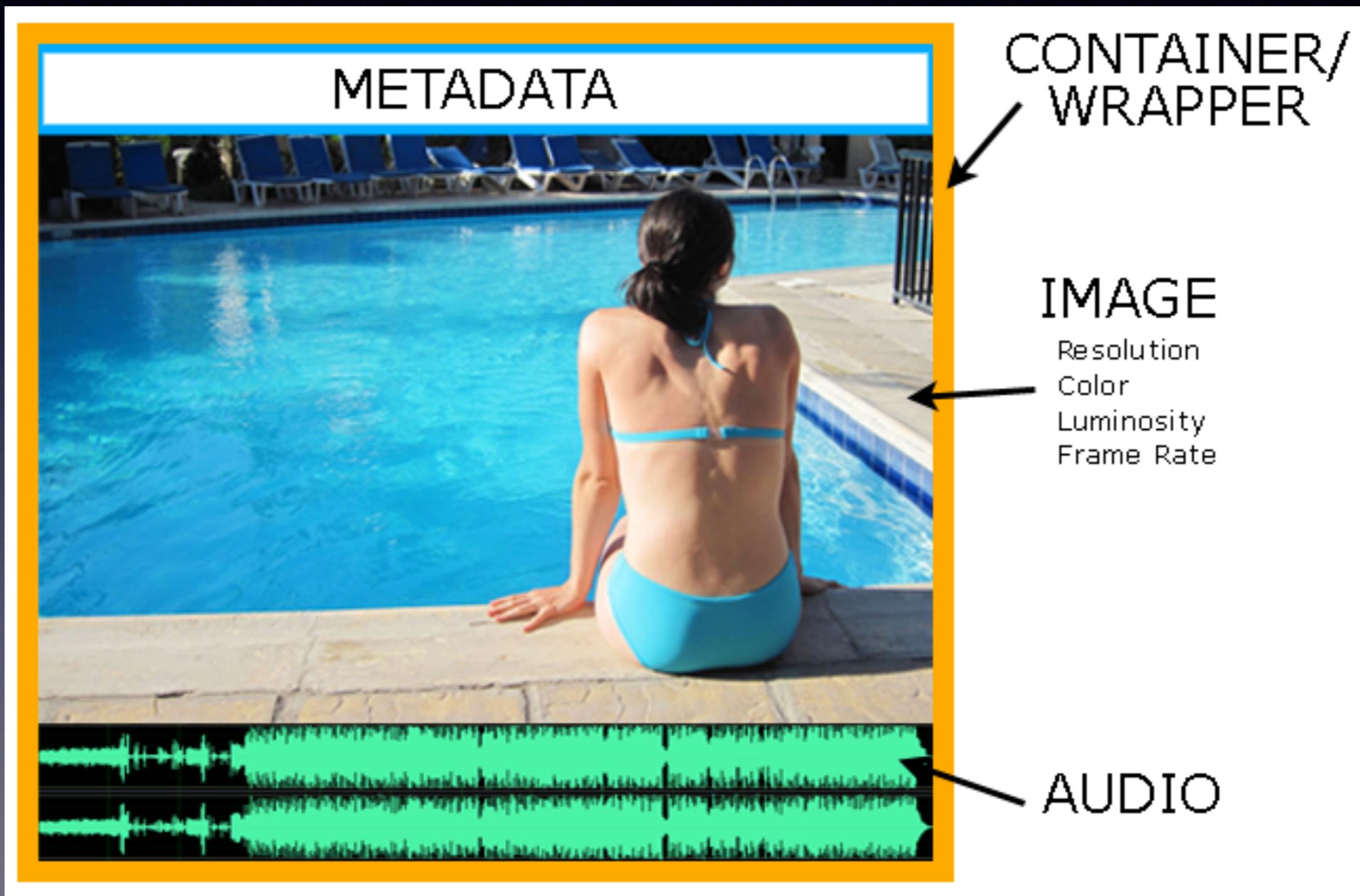
内容消费升级的时代

- 文字 -> 图片 -> 视频 -> AR/VR/直播

直播

- 实时性(Real-Time) 交互性(Interactivity) 的内容
- 载体通常是视频 (Video)

理性认知视频



视频相关动词

术语	典型场景	典型用例
解码(<i>Decoding</i>)	点播、回放(<i>Playback</i>)	源文件 -> <i>Decoding</i> -> 播放
编码(<i>Encoding</i>)	直播(<i>Streaming</i>) -> 录像(<i>Video</i>)	直播流 -> <i>Encoding</i> -> 视频录像
转码(<i>Transcoding</i>)	特效加工处理 文件格式转换	源文件 -> <i>Decoding</i> -> 加工 -> <i>Encoding</i> -> 目标文件

当前形势

- 前提：网络基建带宽充足，资费下降
- 需求：文字、图片无法满足人们对视觉无止境的需求

尝试解决的问题

基于时间序列的音视频流式传输、存储和处理

API based, Live Streaming as-a-Service

开发套件
Development Kit



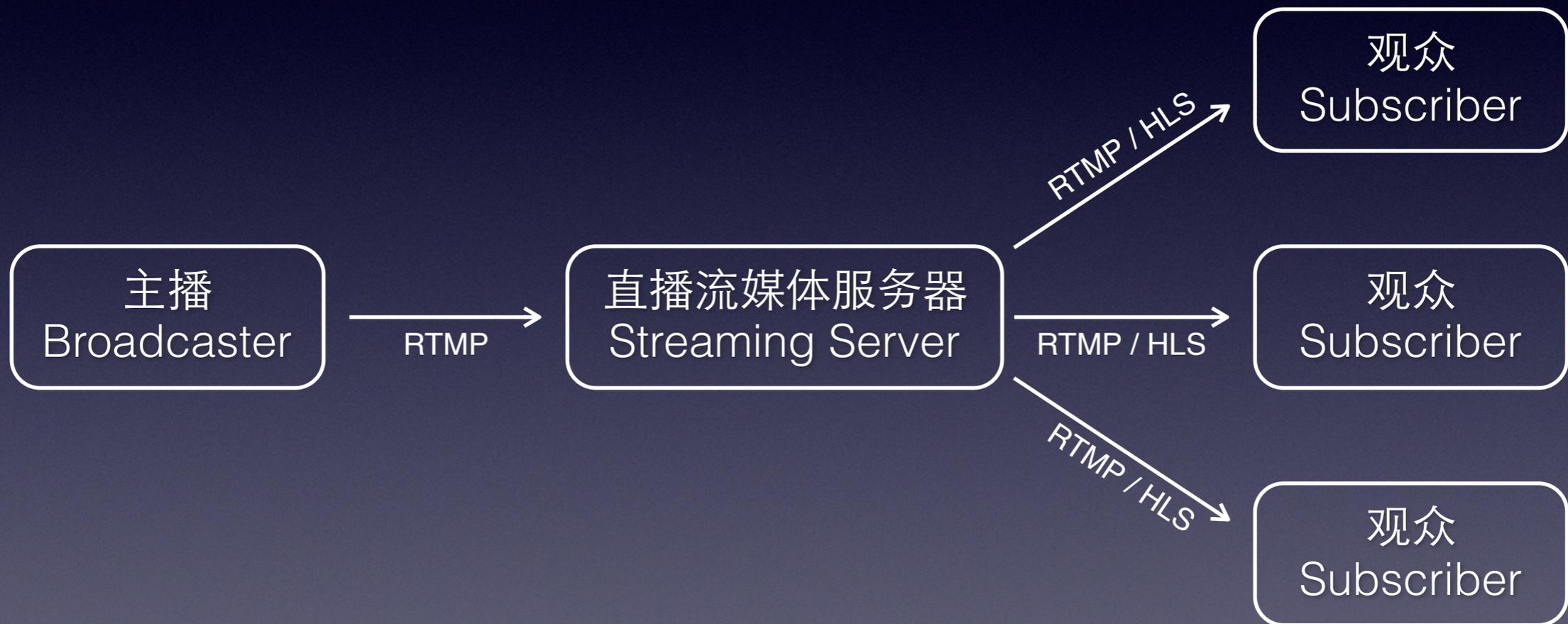
没有找到您熟悉的开发包? 别担心, 还有简洁清晰的 [RESTful API](#) 可以灵活接入

<https://github.com/pili-engineering>

直播模型



直播协议



协议差异

	全称	协议	原理	延时
RTMP	Real Time Messaging Protocol	长连接 TCP	每个时刻的数据，收到后立刻转发	1~3 秒
HLS	HTTP Live Streaming	短连接 HTTP	集合一段时间数据，生成 ts 切片文件，更新m3u8	> 10 秒
HTTP-FLV	RTMP over HTTP	长连接 HTTP	同RTMP，使用HTTP协议	1~3 秒

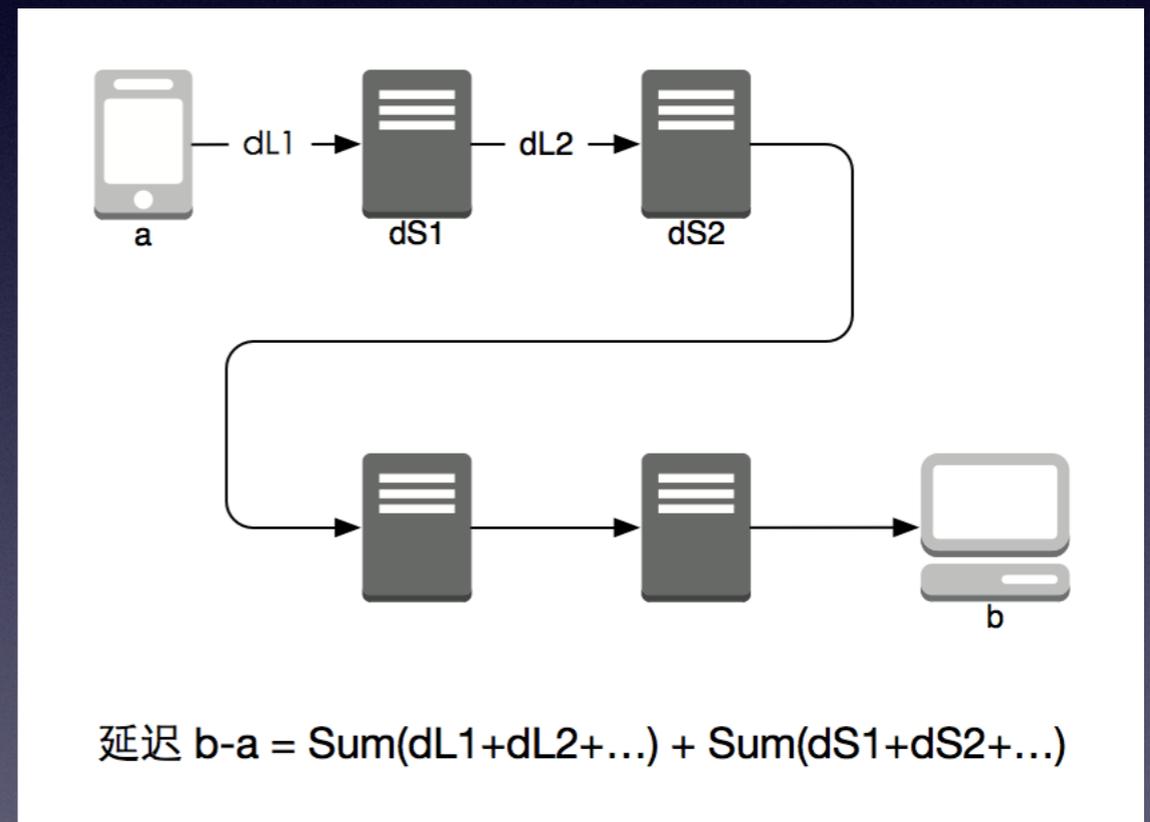
RTMP vs. HLS

	优点	缺点	适用场景
RTMP HTTP-FLV	低延时	跨平台差 Flash Player 以外的 平台都需要做移植	即时，有互动需求
HLS	跨平台 可点播回放	高延时 多次请求，网络质 量影响大	单向广播

直播常见问题

延迟 Delay

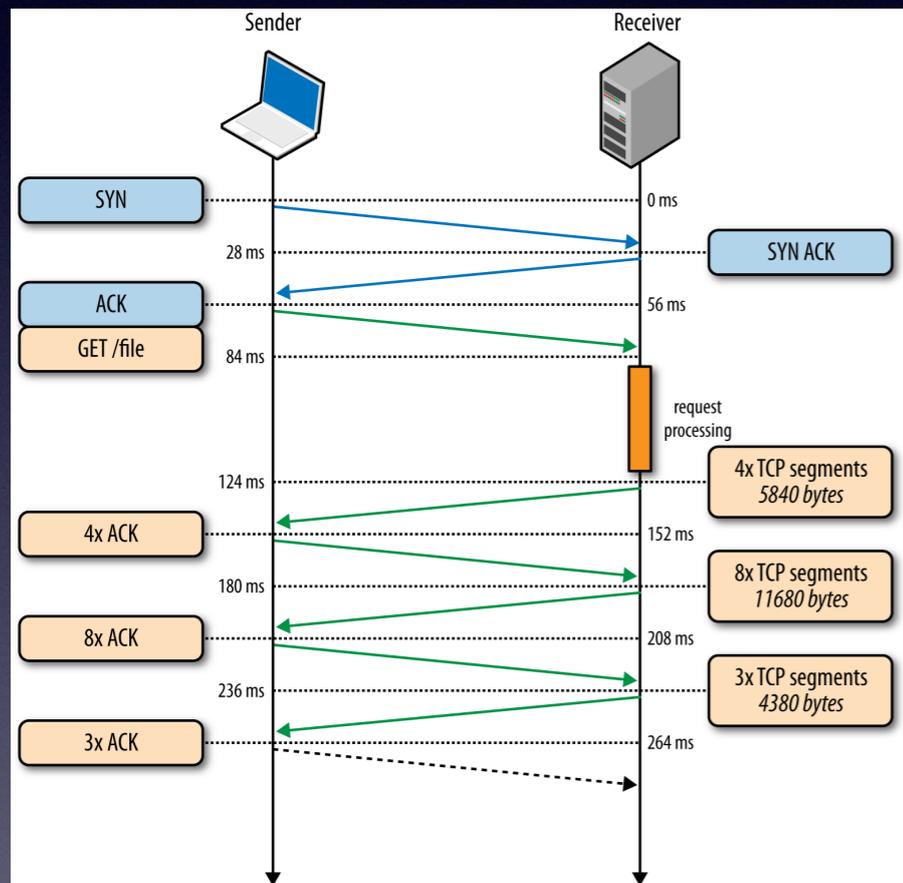
- 延迟指稳定网络下，发送和接收时差
- 转发环节越多，延迟越大
- 可计算



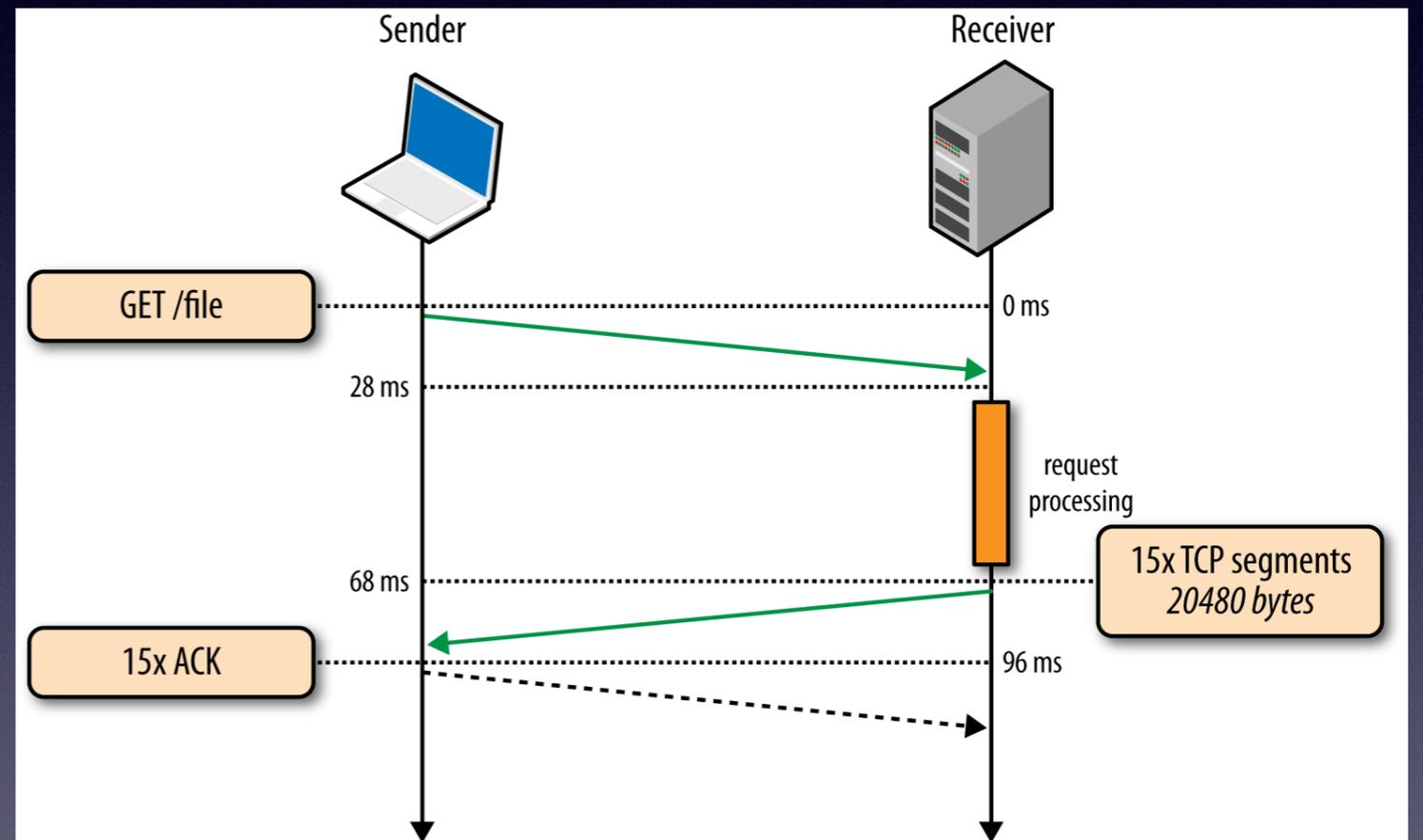
物理延迟

路线	距离 (km)	光在真空中	光在光纤中	光纤中的往返时间
纽约到硅谷	4148	14 ms	21 ms	42 ms
纽约到伦敦	5585	19 ms	28 ms	56 ms
纽约到悉尼	15993	53 ms	80 ms	160 ms
赤道周长	40075	133.7 ms	200 ms	400 ms

逻辑延迟



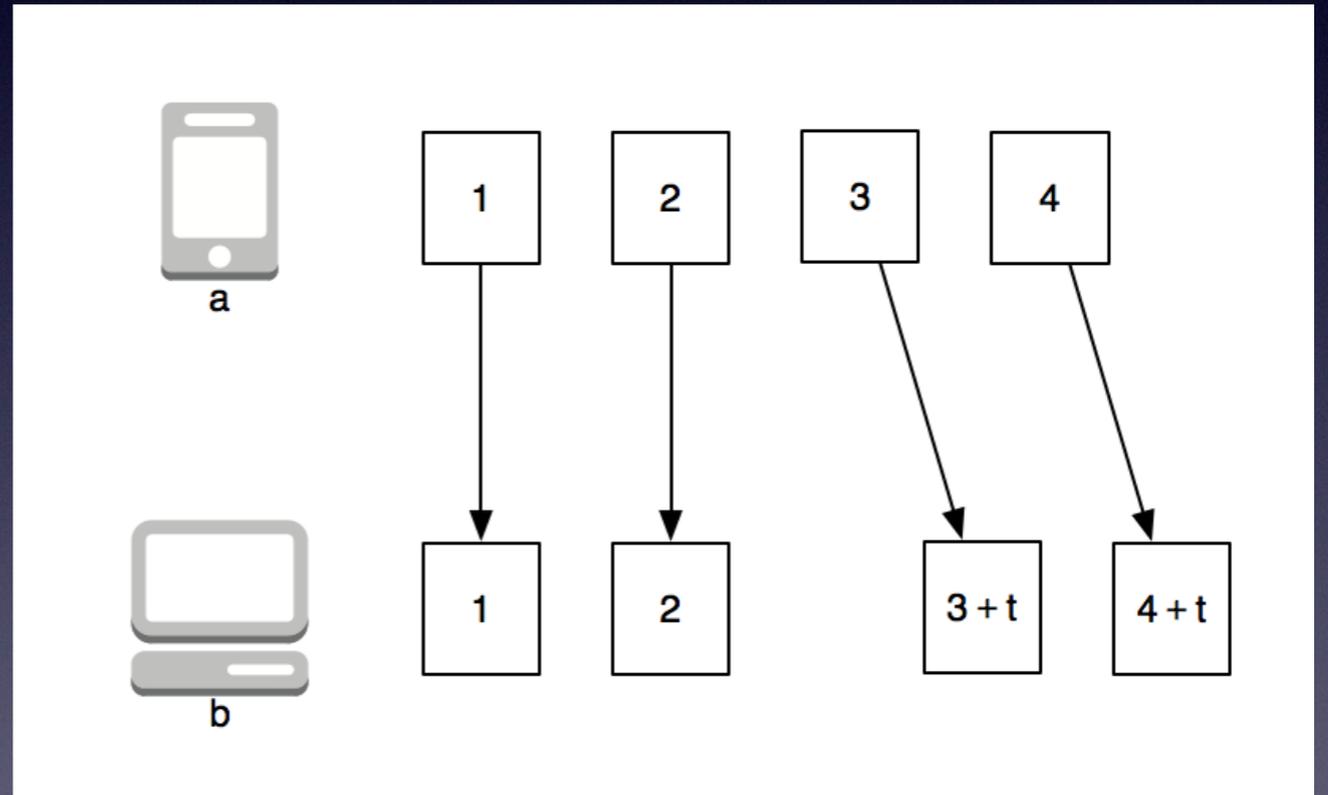
短连接 + TCP 慢启动



长连接 (延迟降低 275%)

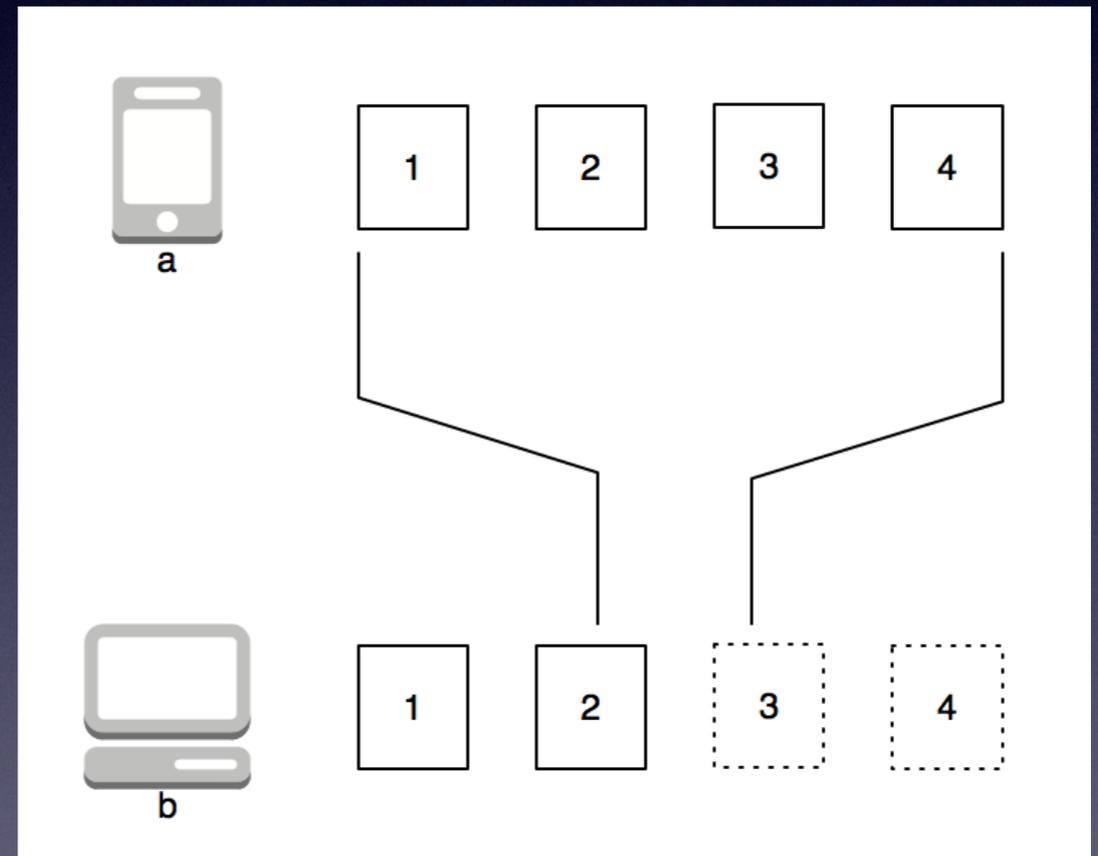
抖动 Jitter

- 突发的转发速度变化
- 增加后续播放的延迟
- 无法预先计算



传输速度

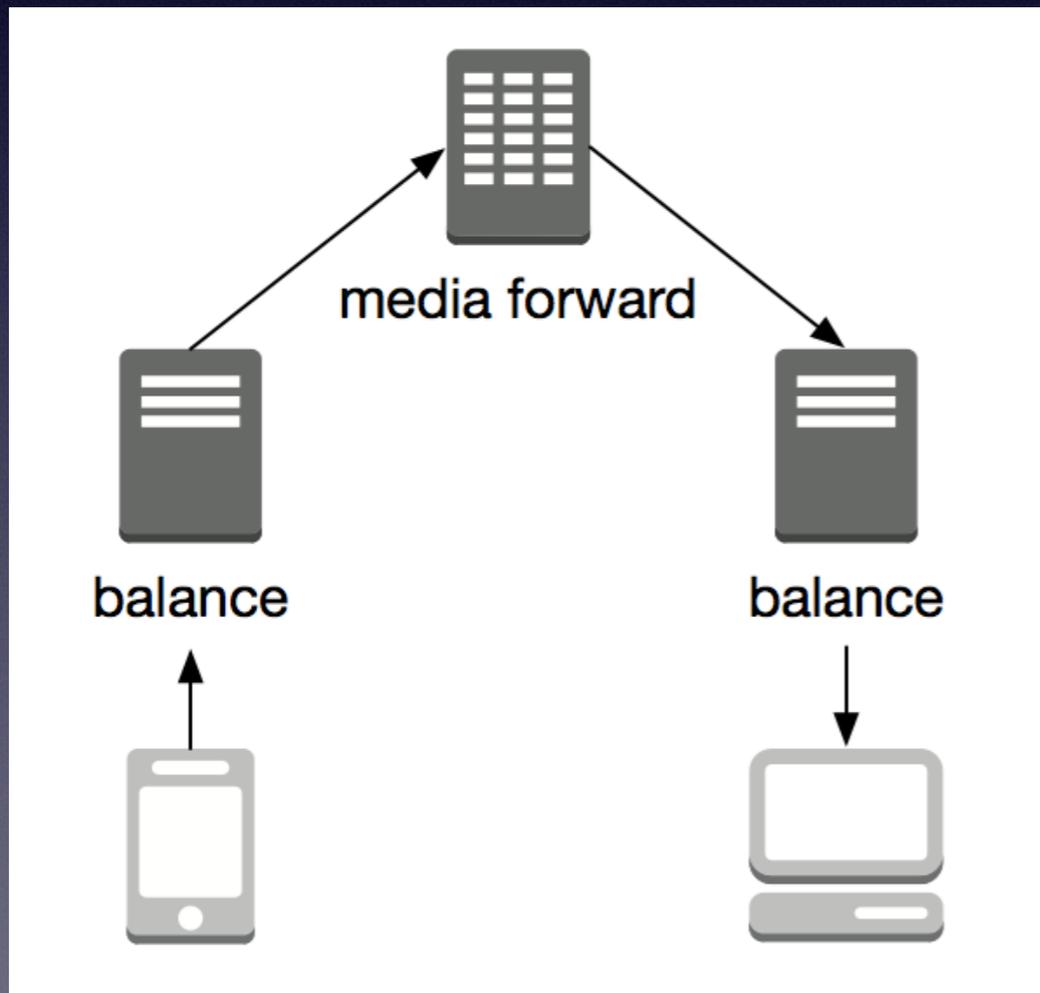
- 推流环境与播放环境不一致，速度不一样
- 推流环境（大部分情况）可预先控制
- 播放环境（几乎）无法控制



如何打造满足高品质诉求的直播服务？

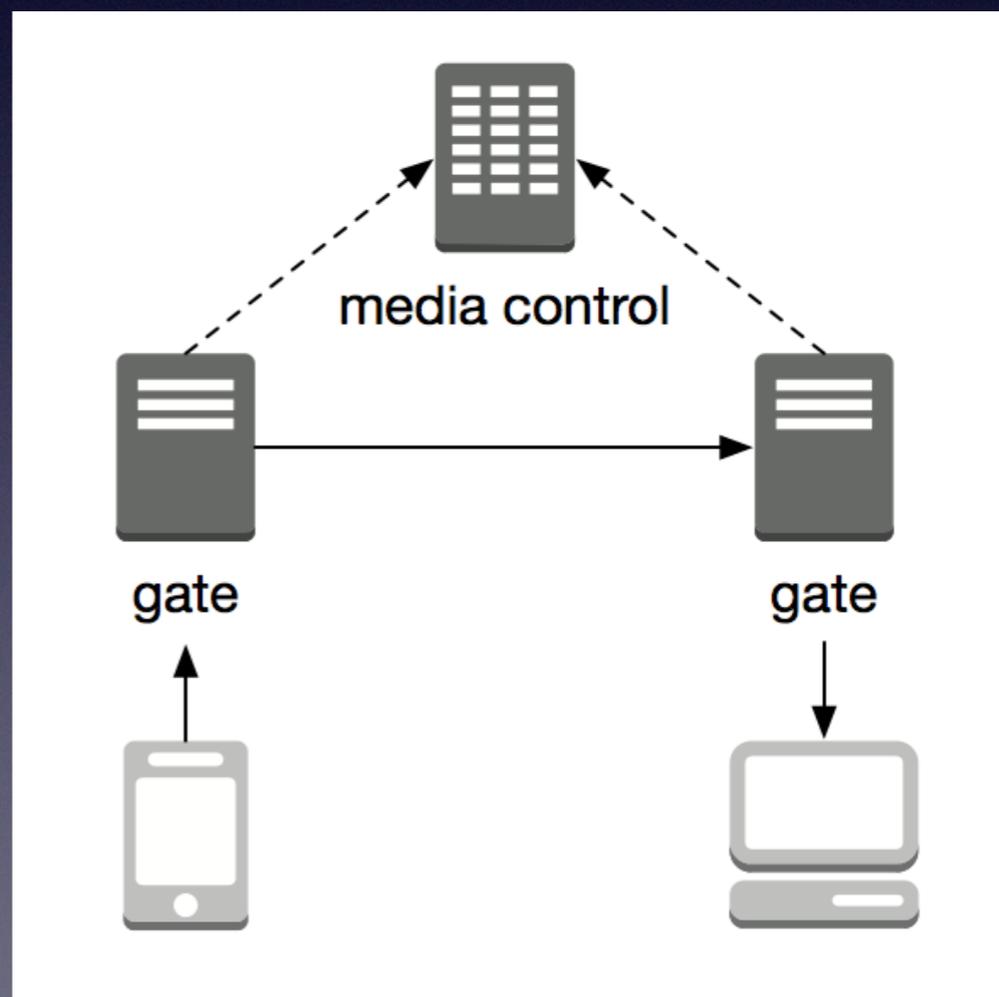
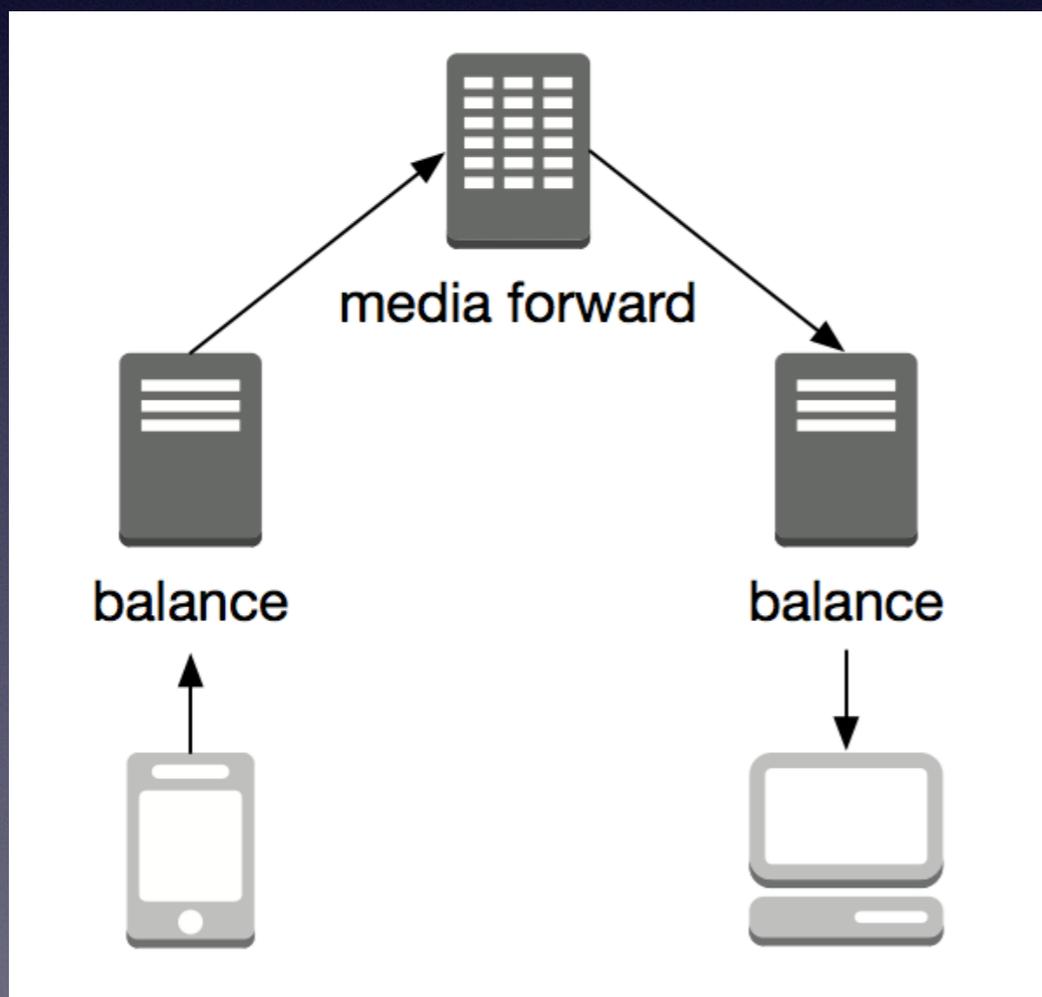
精简服务端转发逻辑

减少延迟



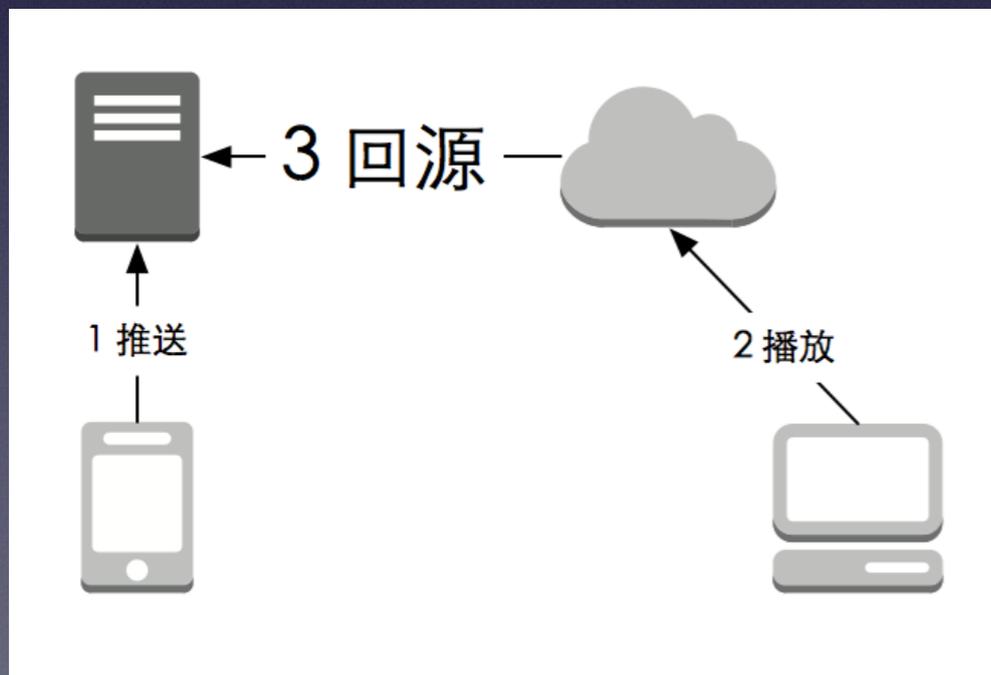
精简服务端转发逻辑

减少延迟



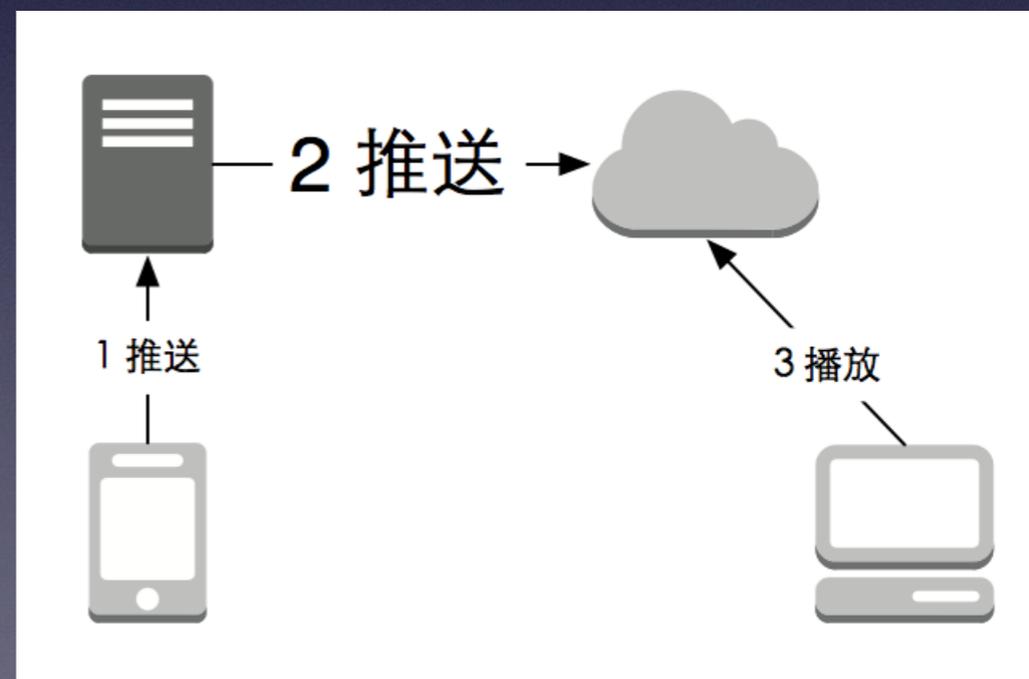
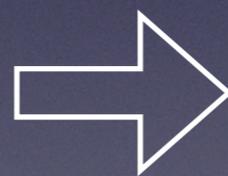
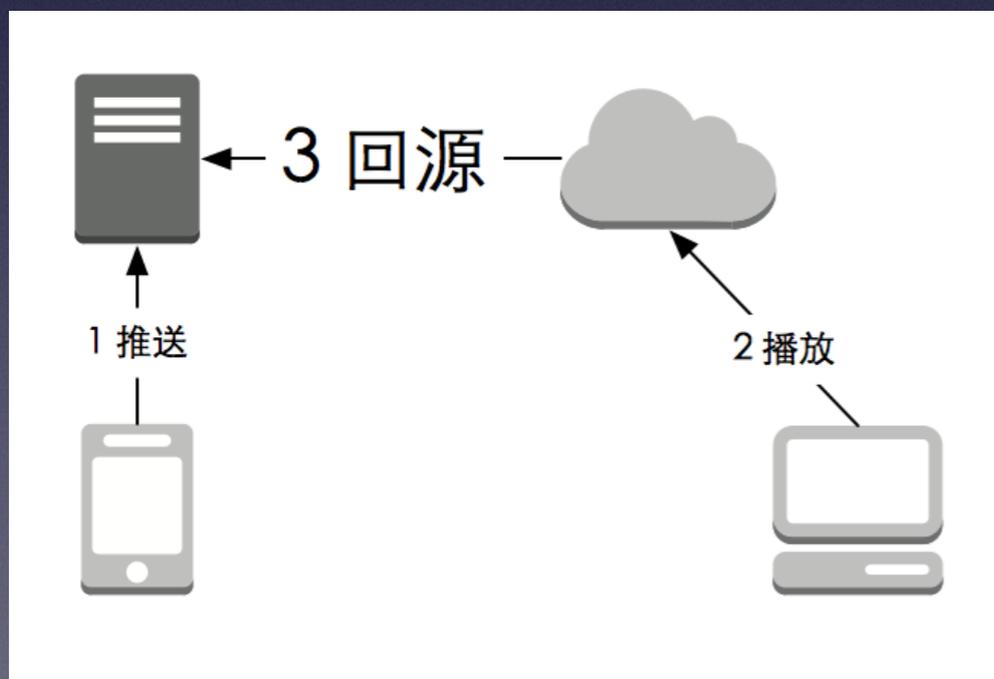
主动推送

- 减少回源延迟
- 首屏快速打开



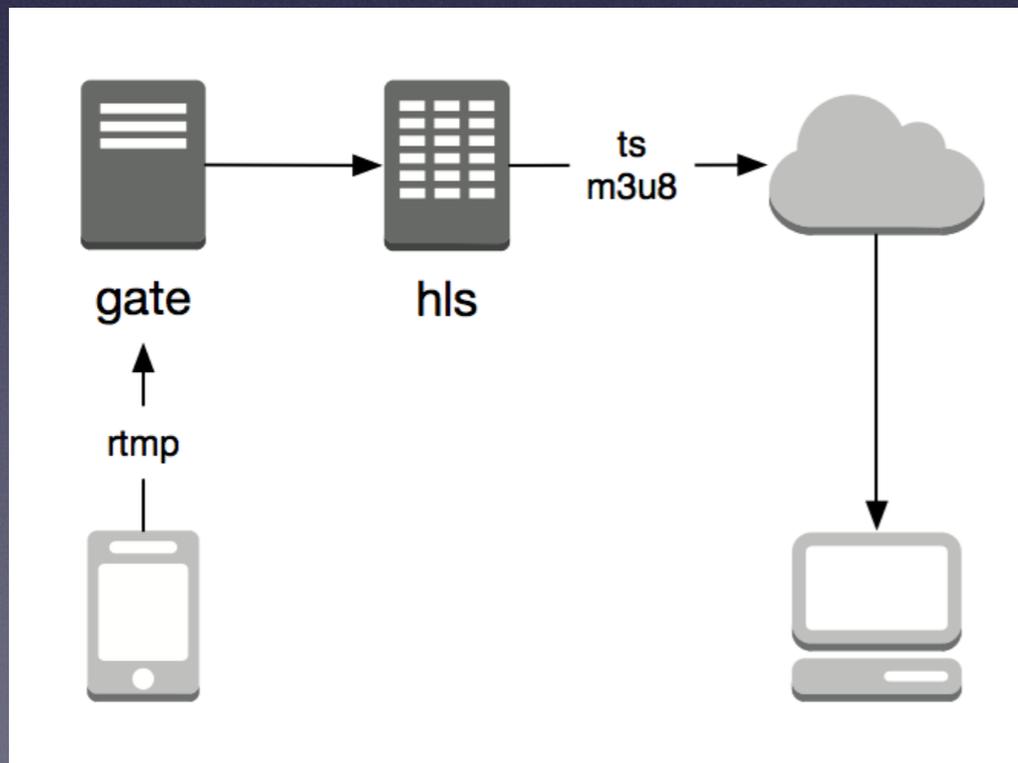
主动推送

- 减少回源延迟
- 首屏快速打开



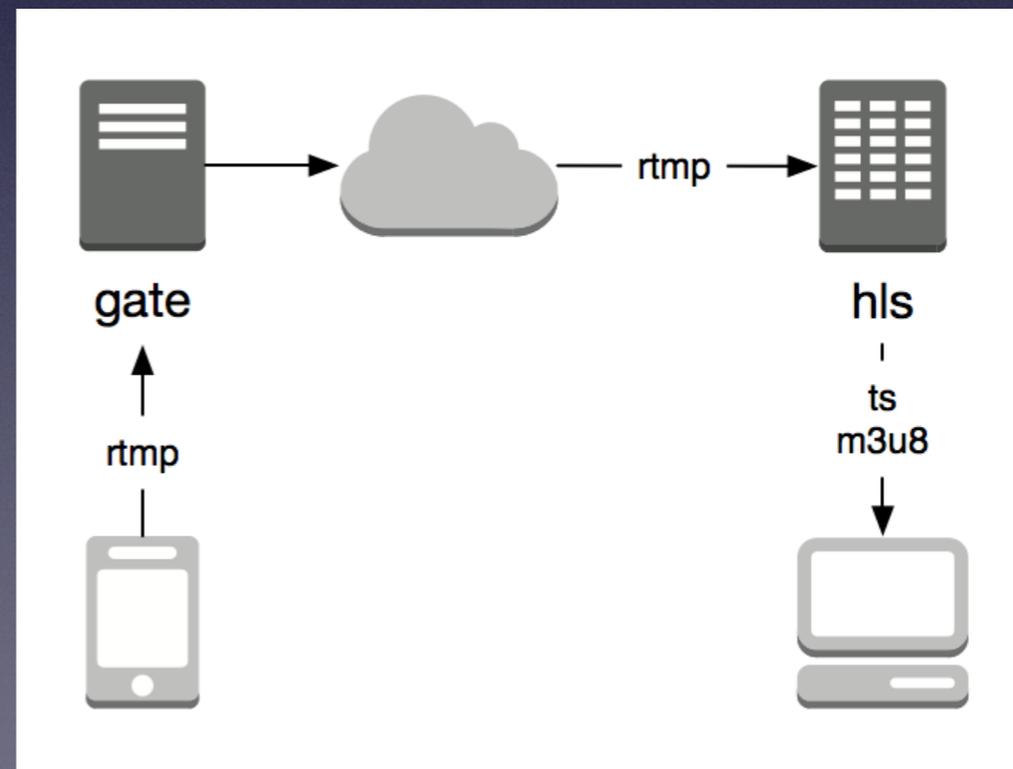
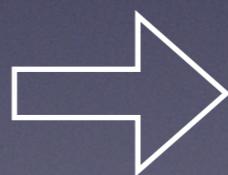
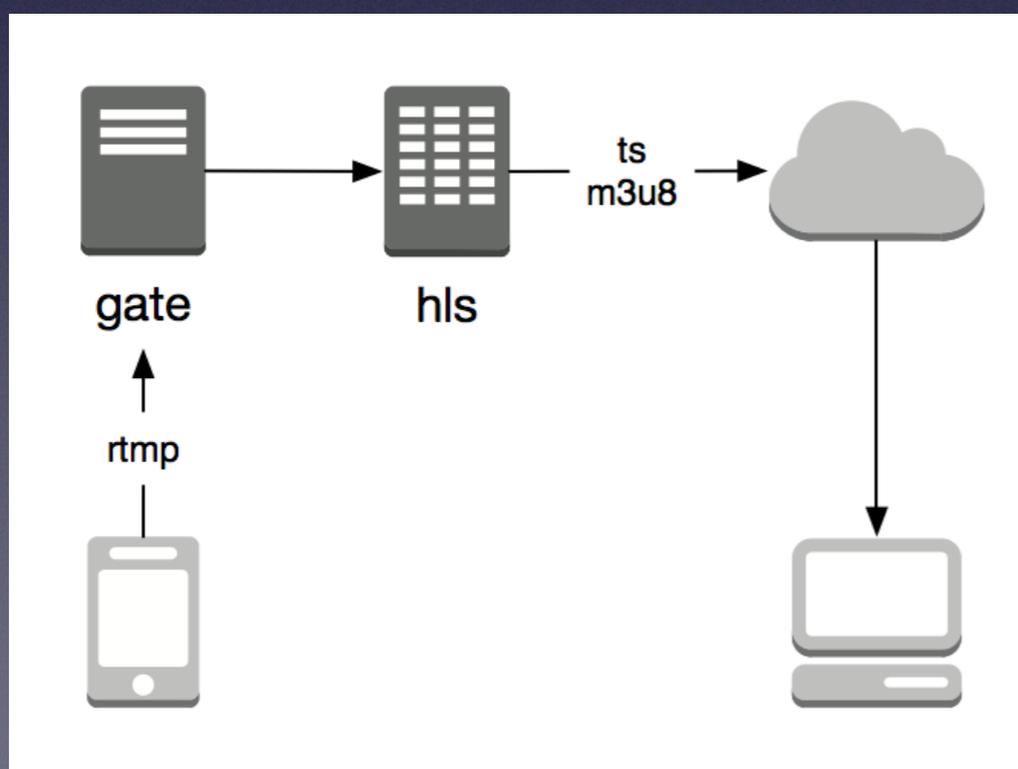
贴近终端就近处理与分发

- 减少延迟
- 减少抖动
- 提高速度



贴近终端就近处理与分发

- 减少延迟
- 减少抖动
- 提高速度



多核时代，编程语言选型

语言差异

- C语言，内存管理
 - 内存抽象为连续空间
 - 不再区分页和数据段
 - 区分堆和栈
 - 内存操作原语
 - malloc/free
 - 由程序员负责内存管理
- Go语言，线程管理
 - 线程抽象为Goroutine
 - 不再区分线程和异步
 - 区分Goroutine和Channel
 - 线程操作原语
 - go/chan <- /<- chan
 - 由程序员负责线程管理

Pili Streaming Cloud

全球首个Go语言实现的企业级直播流媒体云服务



场景篇

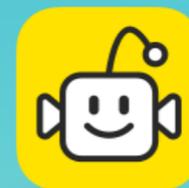
手机直播

移动直播

时下热门



美拍



秒拍直播



龙珠直播



映客



nice好赞



咸蛋家



在直播



熊猫TV



花椒



移动直播SDK

大家都在用

The screenshot shows the GitHub mobile app interface. At the top, the status bar displays '中国电信', 'VPN', '12:24', and '92%' battery. The browser address bar shows 'GitHub, Inc.'. The main header features the 'GitHub' logo and a hamburger menu icon. Below the header, there are two tabs: 'About' and 'Repositories'. The 'About' tab is active, displaying the profile for 'Pili Streaming Cloud' (pili-engineering). The profile includes a logo of two stylized eyes, the organization name, a list of office locations (San Francisco, Shanghai, Beijing, Hangzhou, Shenzhen, Singapore, Sydney), and an email address (pili@qiniu.com). Below the profile, a section titled 'Popular repositories' lists four repositories. A red rectangular box highlights the star counts for these repositories: 578, 186, 180, and 149. The bottom of the screen shows the standard iOS navigation bar with icons for back, forward, share, bookmarks, and tabs.

Repository Name	Star Count
pili-engineering/PLCameraStreamingKit	578
pili-engineering/PLDroidCameraStreaming	186
pili-engineering/PLPlayerKit	180
pili-engineering/PLDroidPlayer	149

SDK 都干了啥

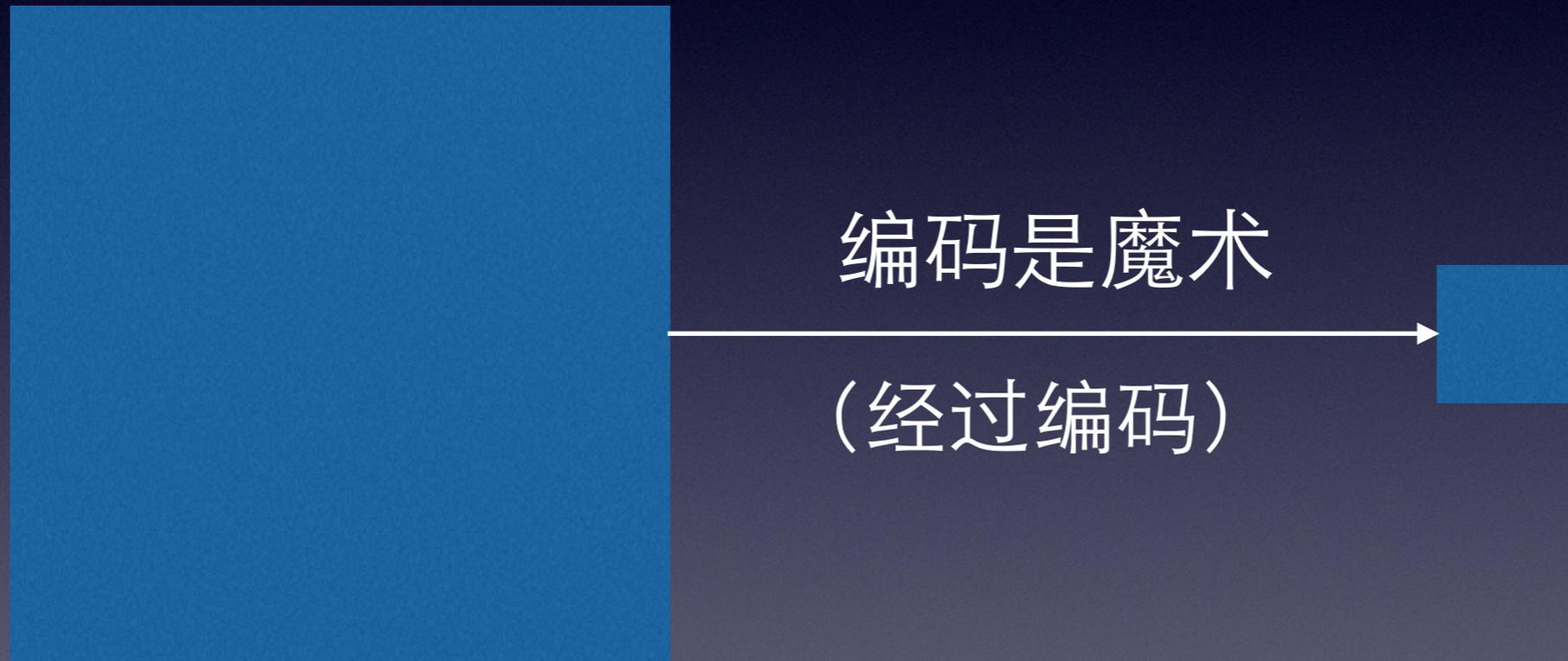
SDK	平台	采集	编码	推流
PLDroidCameraStreaming	Android	√	√	√
PLCameraStreamingKit	iOS	√	√	√
PLStreamingKit	iOS	X	√	√

采集的功能边界

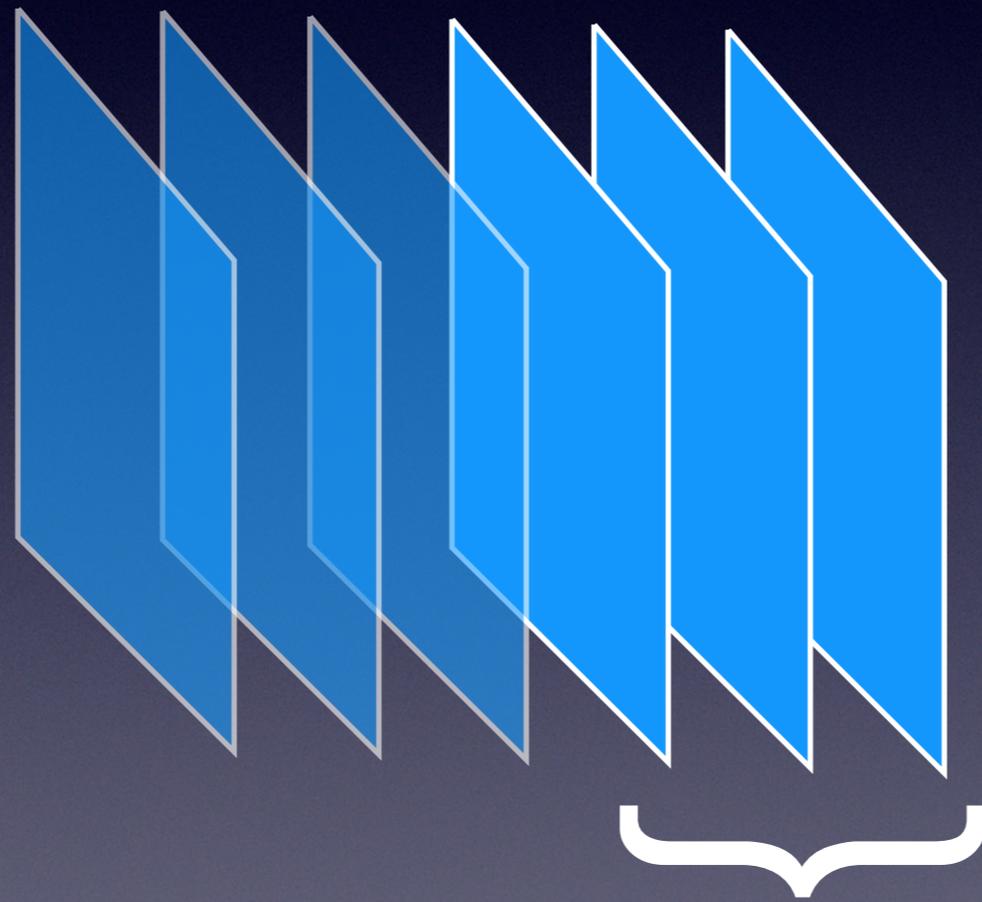
设备	输入	输出
视频	摄像头	图片
音频	麦克风	PCM 音频数据

编码有什么作用？

编码有什么作用



编码时发生了什么

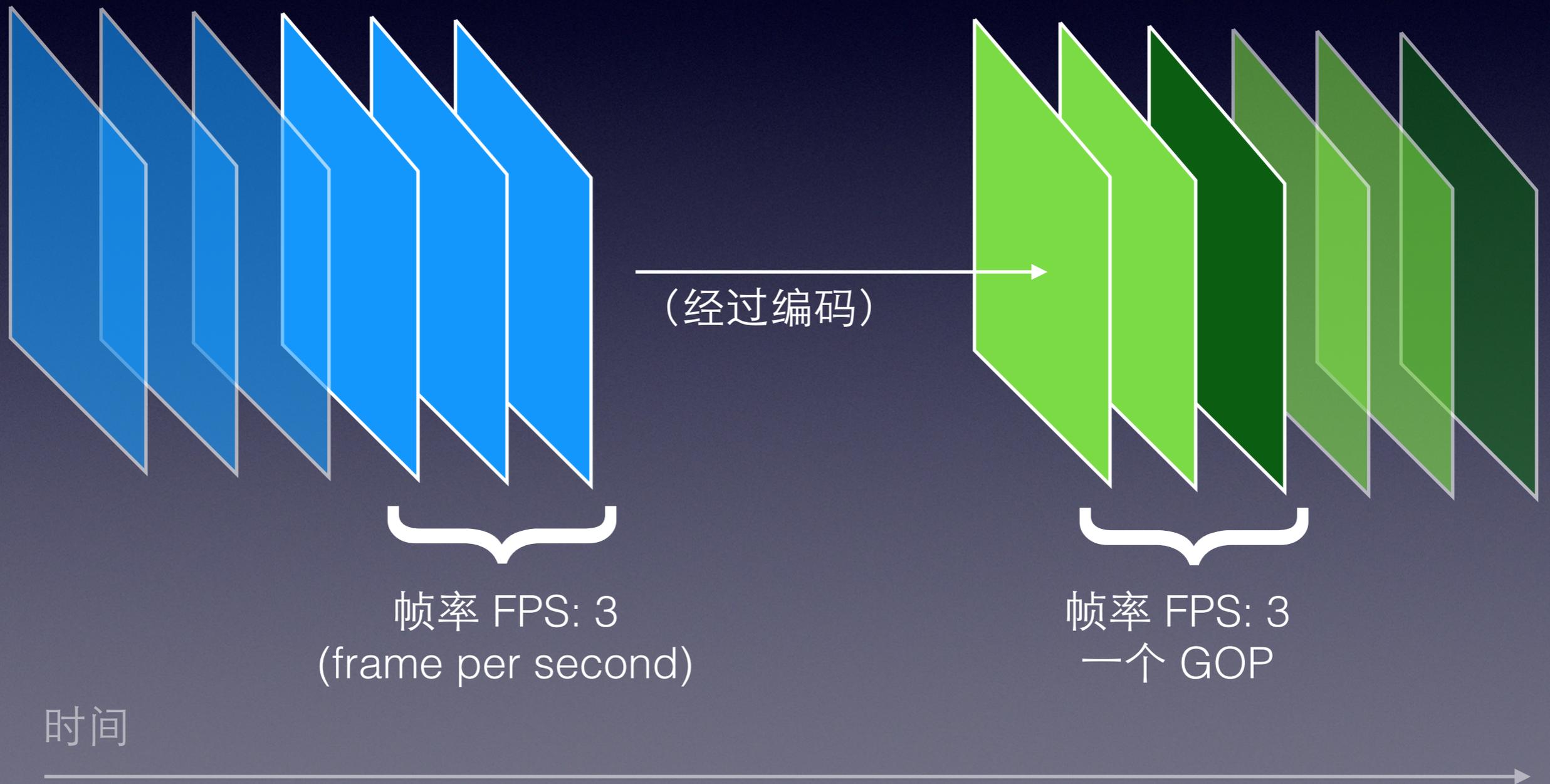


帧率 FPS: 3
(frame per second)

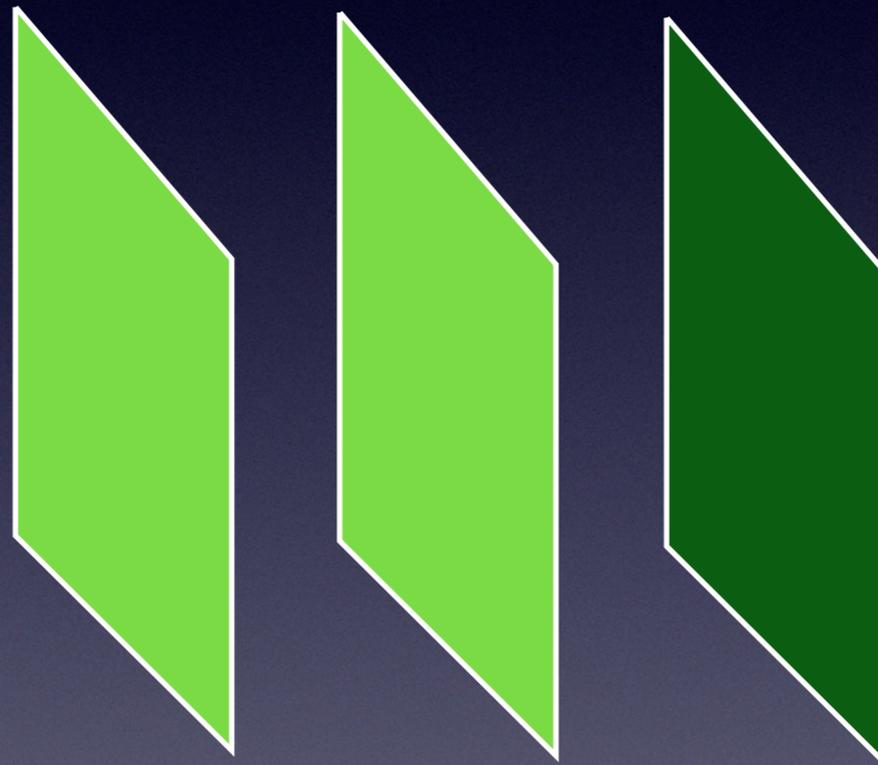
时间



编码时发生了什么



编码时发生了什么



参考帧
(B/P 帧)

关键帧
(I 帧)

关键帧 vs. 参考帧

帧类型	是否可以独立解码	依赖谁解码	大吗?
I 帧	√	NA	大
P 帧	X	前	小
B 帧	X	前与后	小

编码器

编码方式	资源消耗	兼容性	当前偏好
软编	CPU	好	√
硬编	GPU	Android 设备存在差异	X

推流

推流协议	封包	发送
RTMP	FLV Tag	RTMP Chunk

资源消耗

编码方式	计算资源消耗	I/O 消耗
编码	高	低
封包	低	低
发送	低	高

美颜怎么做？

iOS PLStreamingKit

特性	发生时机	用法	扩展
美颜	编码前	GPUImage	可自定义美颜算法

Android PLDroidCameraStreaming

特性	发生时机	用法	扩展
美颜	编码前	SurfaceTextureCallback StreamingPreviewCallback	可自定义美颜 算法

SDK 异常懵逼处理

- 网络异常了

SDK 异常懵逼处理

- 网络异常了
- 重连，网络又异常了
- 重连，网络又异常了
- 重连，网络又异常了
- 重连.....

SDK 异常懵逼处理

情况	谁来监控	怎么处理
App 网络状态监控	开发者自己	停止/恢复推流
推流异常断开	SDK	重连

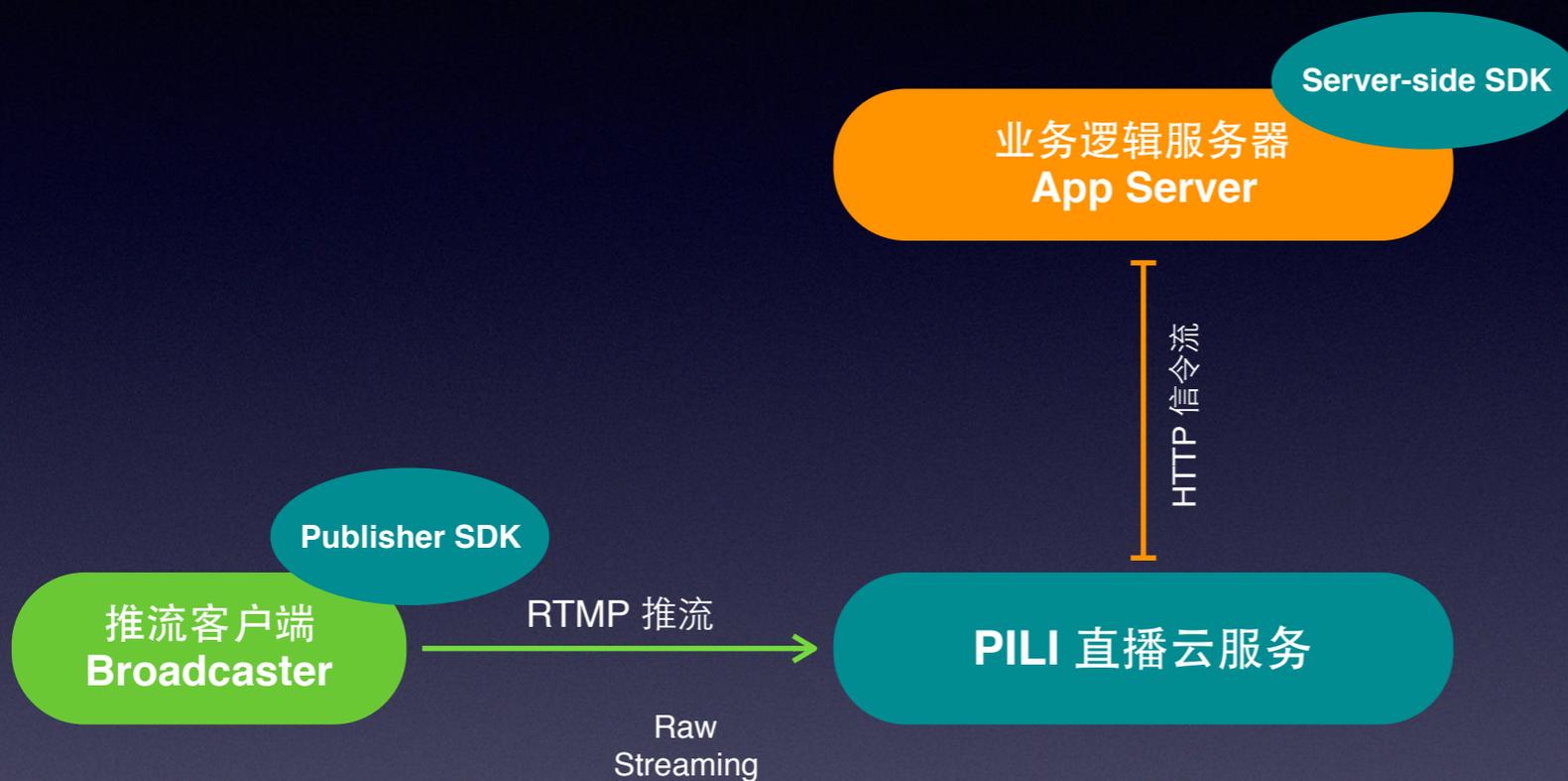
问题还在，怎么办？

审视:产品/架构/代码

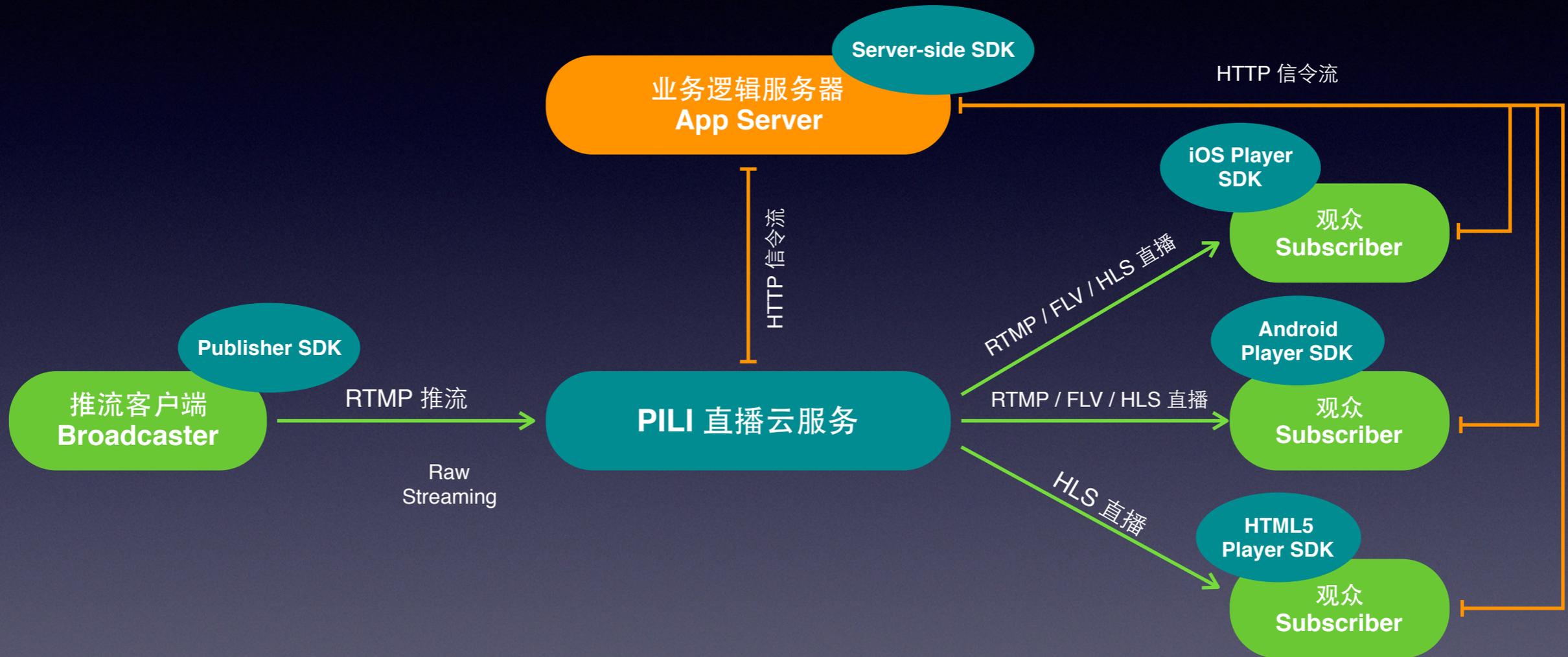
- 对网络很差的主播给予界面提示
- 业务判断主动断掉网络很差的主播
- 避免一个流做抢麦逻辑
- 直播的流 (socket) 与业务的流 (直播间) 不可混为一谈

如何快速构建直播APP

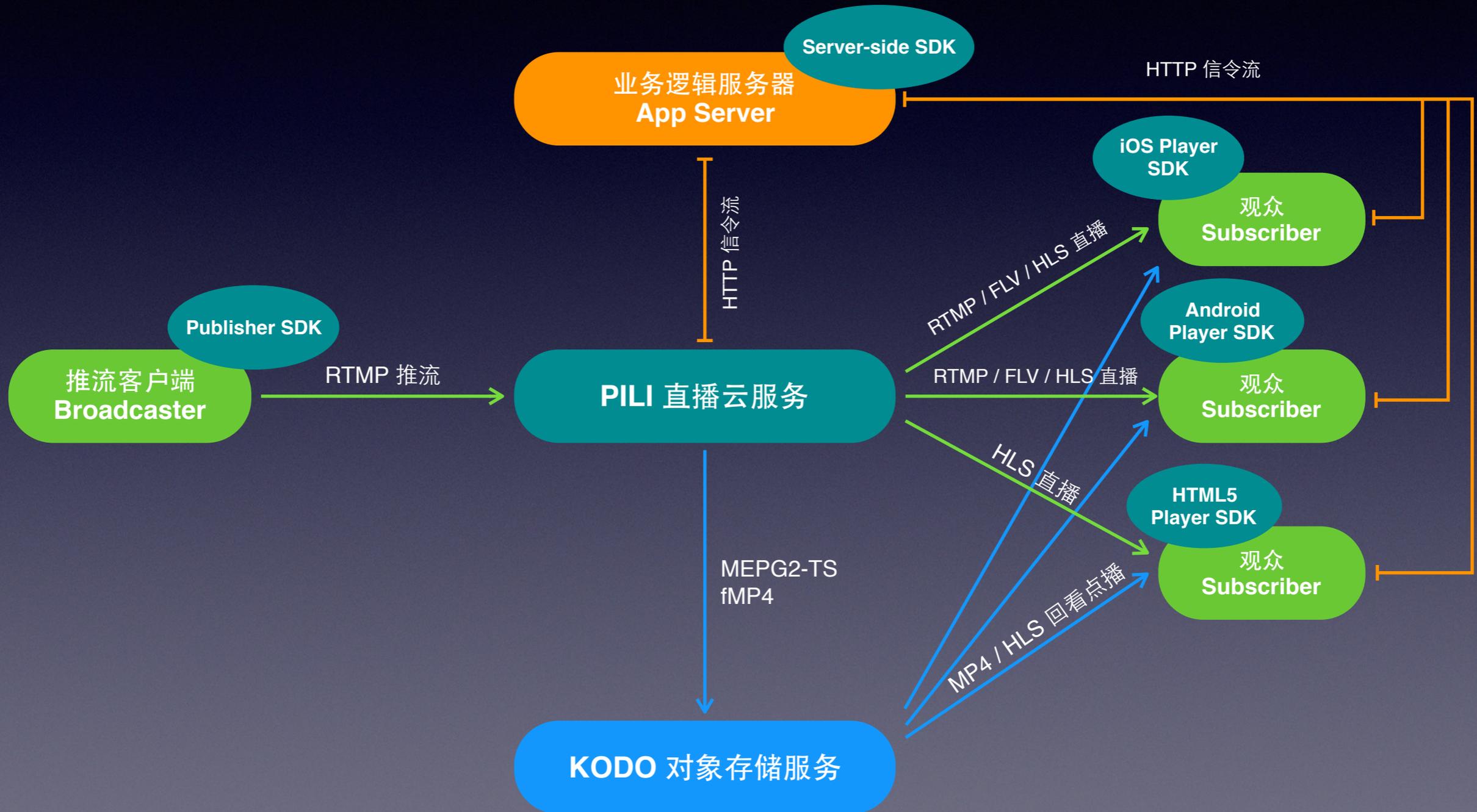
采集和推流



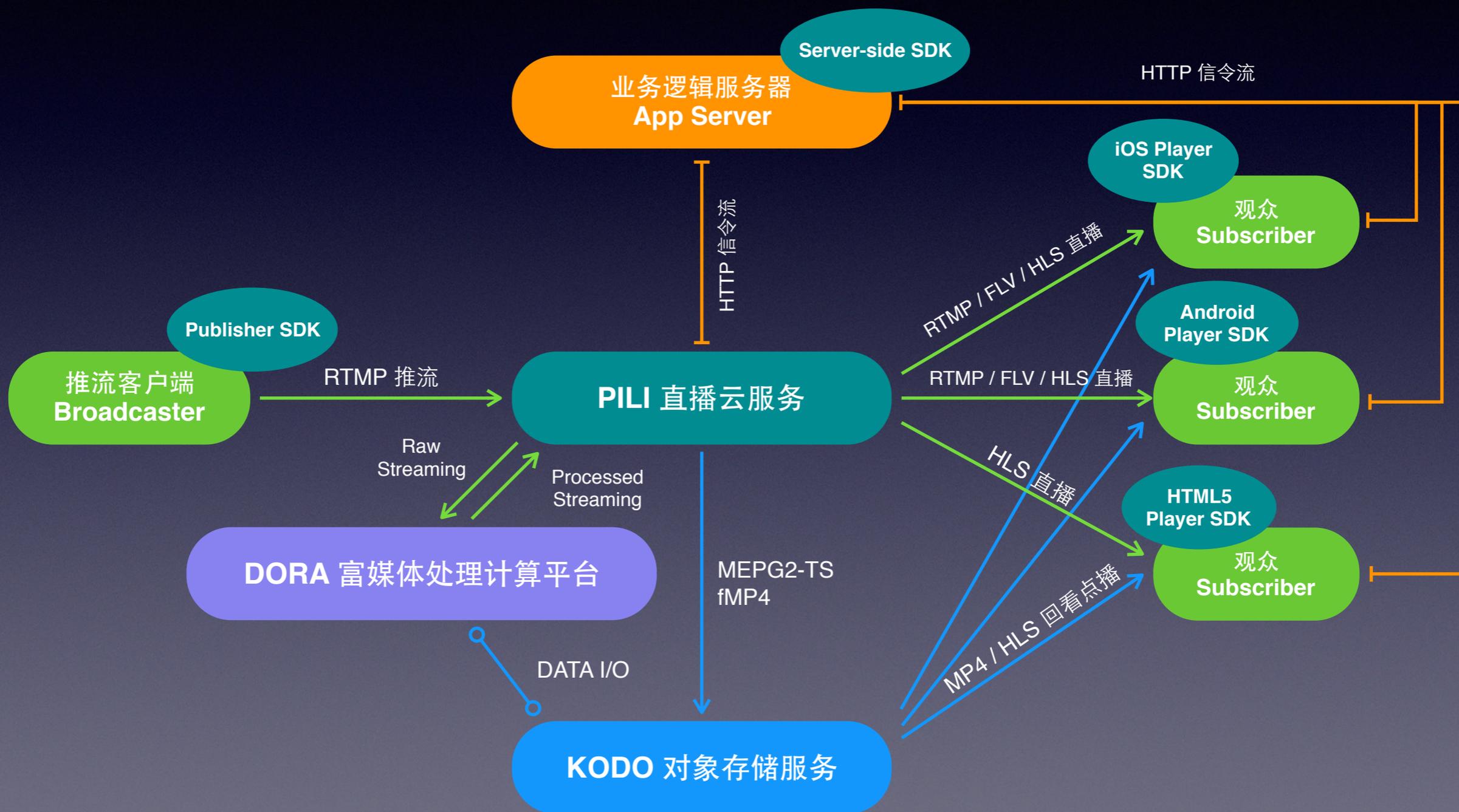
传输和分发



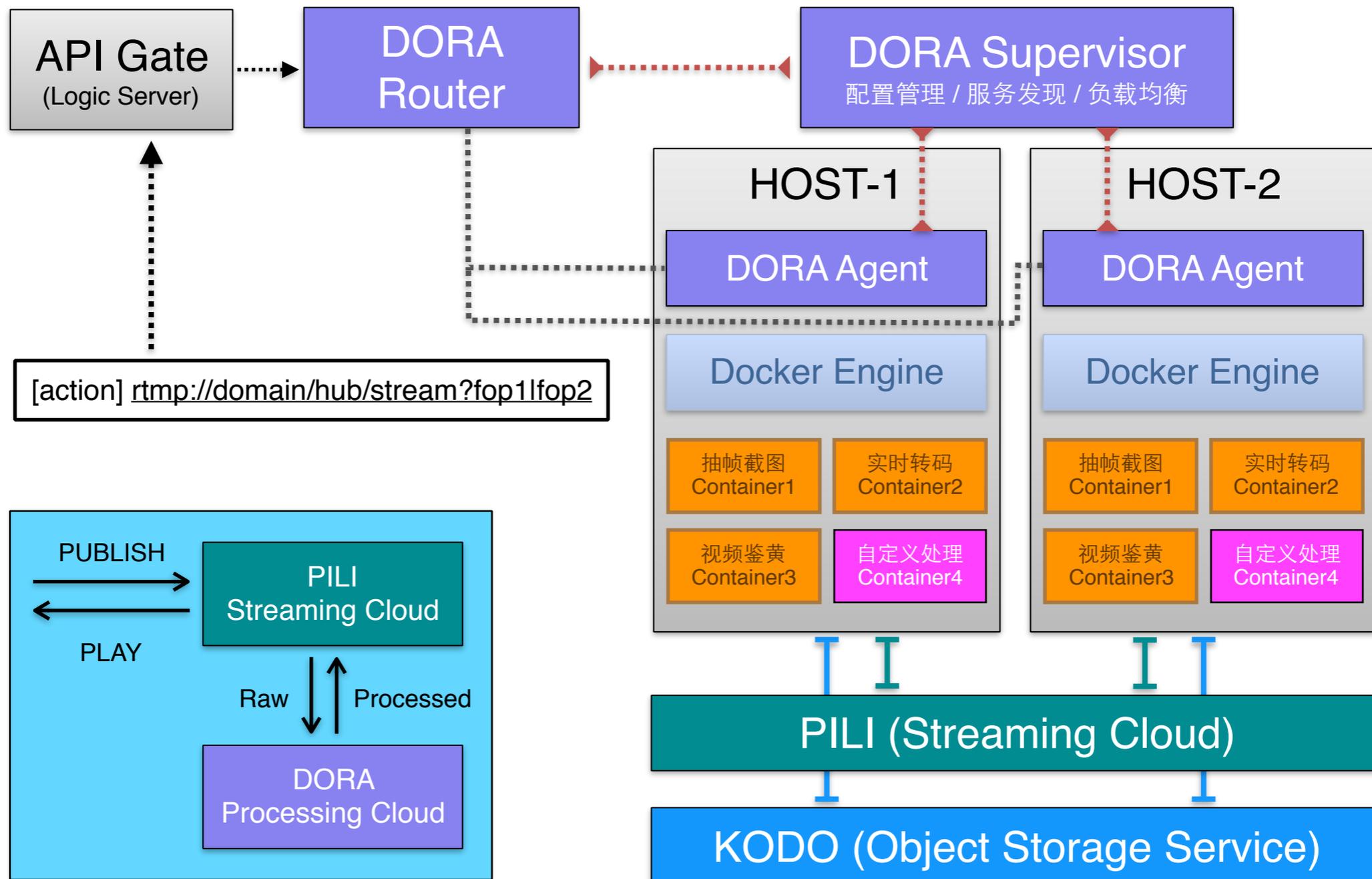
存储与点播



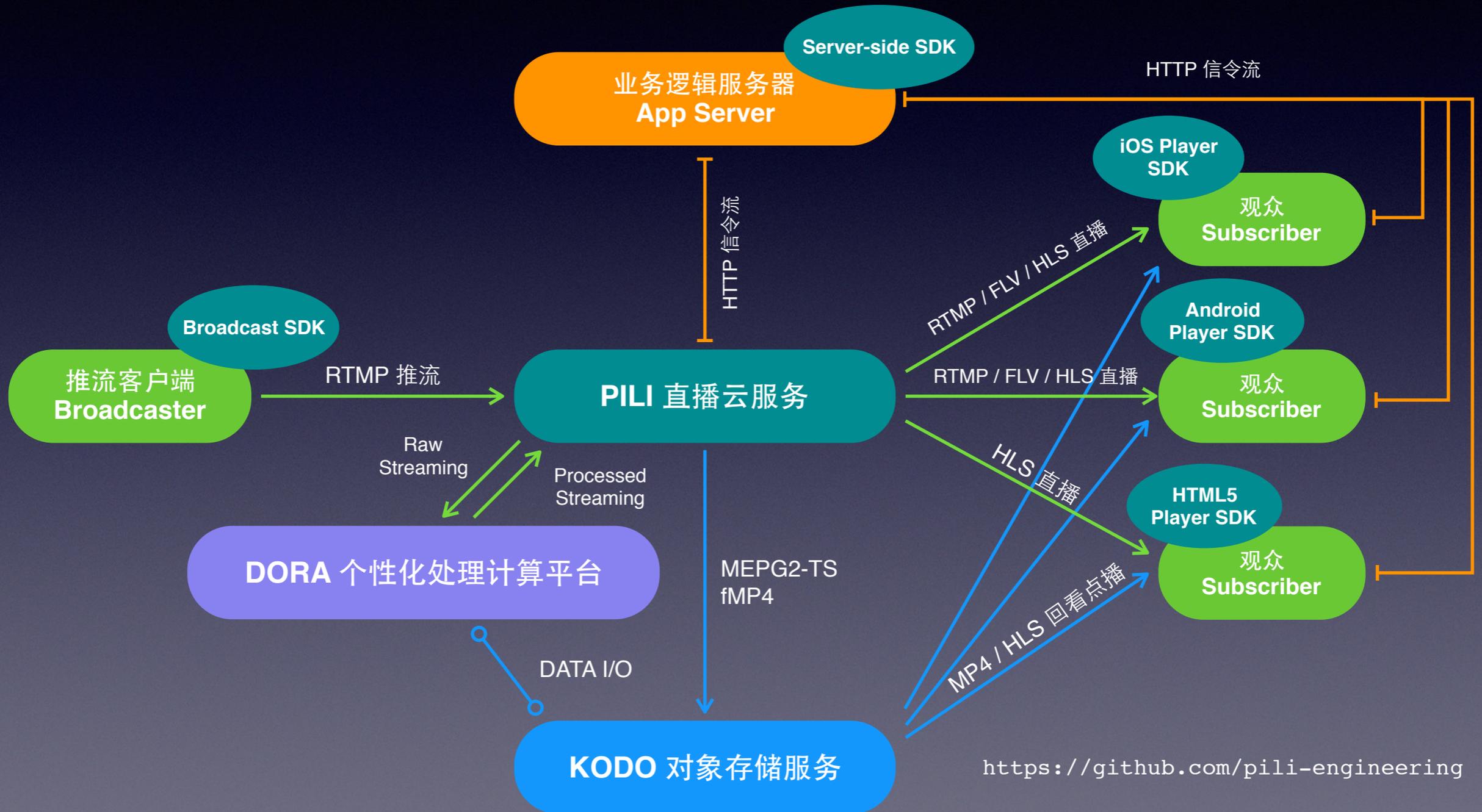
处理 (识别/转码/等等)



七牛云处理架构



七牛云直播架构



直播痛点诉求

- 云端存储：直播录制功能
- 在线点播：直播回放功能
- 处理需求：转码、抽帧、鉴黄 等功能
- 移动时代：海量 UGC 并发推流，不仅仅是分发

直播常用功能

功能特性	对应需求	是否支持
直播存档	云端录制、转存、下载	√
直播回放	延迟直播、回放点播	√
直播实时转码	多码率适应带宽、多分辨率多屏	√
视频抽帧	截图、快照	√

自建 vs. 上云

	传统直播技术	新型直播技术
方式	服务器+流媒体软件+CDN	Pili Streaming Cloud + SDK
优点	“自认为更可控”	“多快好省”
缺点	UGC并发瓶颈、功能个性化难	一次新的尝试成本

资料

<https://github.com/pili-engineering>

提问时间