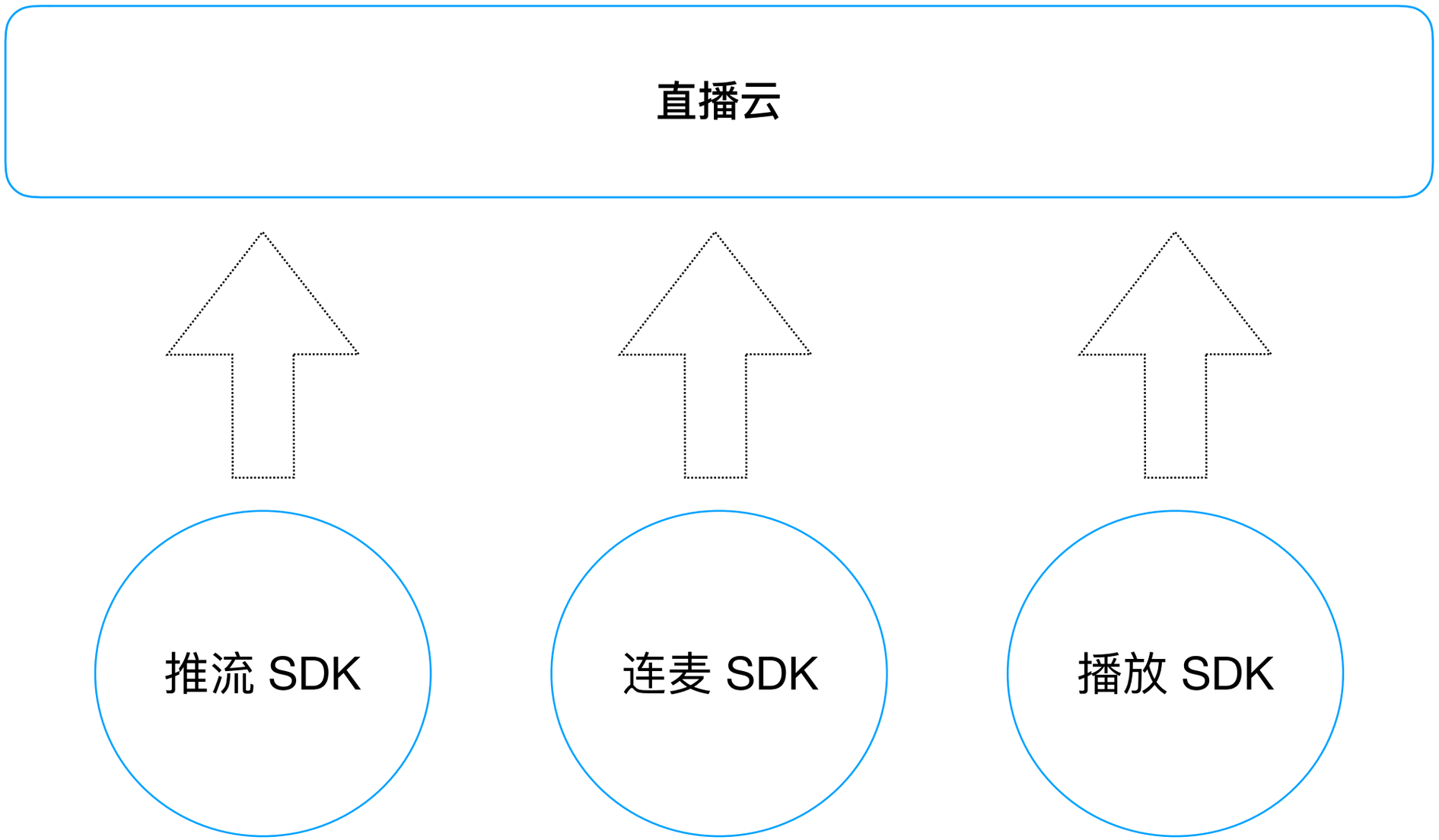
移动直播的关键技术优化

卢 俊 @ 七牛客户端负责人

lujun@qiniu.com

背景介绍



演讲大纲

1. 直播的痛点介绍

2. 技术层面的关键优化

- 秒开优化
- 卡顿优化
- 延时优化
- 清晰度优化
- 功耗优化
- 排障的优化

3. 直播部分高级功能技术剖析

- 连麦方案
- 合流原理

4. 总结

直播 - 痛点

首开慢

卡顿率高

延时大

马赛克

功耗大

排障不易

直播 - 痛点

首开慢

卡顿率高

延时大

马赛克

功耗大

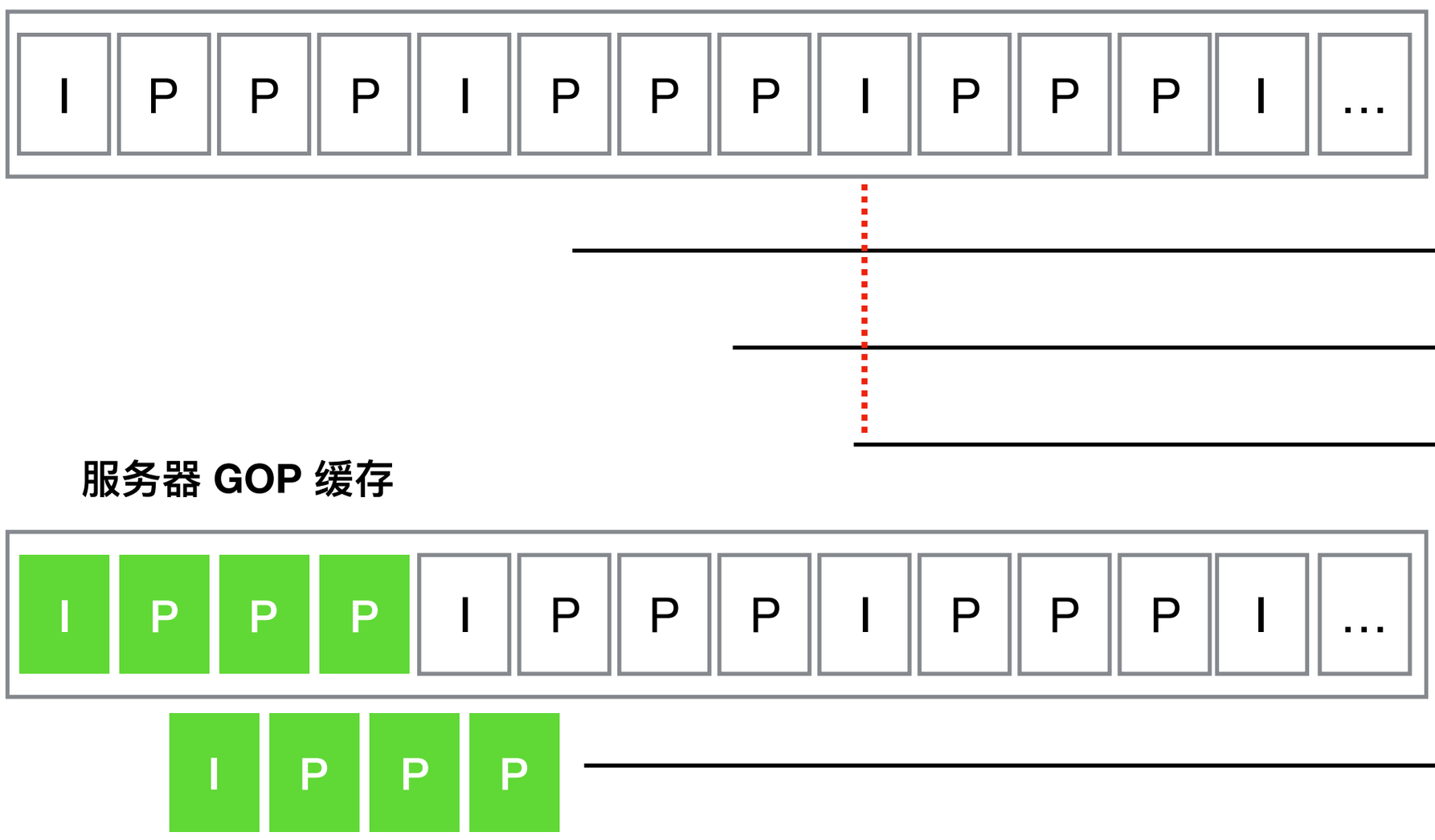
排障不易

直播 - 首开慢

首开速度和用户感受

首屏打开时间	用户感受
0 ~100 ms	很好，很快
100~300 ms	还好，一般
300~1000 ms	略慢，再等等看
> 1000 ms	这么慢，我要切应用了
> 10000 ms	嘛，服务器宕机了么

直播 - 首开慢 - 优化



直播 - 首开慢 - 优化

Step1 播放域名 -> 服务器 IP 地址 → 提前完成解析，直接传入 IP

Step2 连接服务器，发送播放请求

Step3 服务器 -> 码流的媒体信息

Step4 播放器接收媒体信息 -> parser ->init demuxer & decoder

Step5 服务器 -> 音视频数据

Step6 播放器接收数据 -> demux -> decode -> display

优化速度

首帧立即解码渲染，不缓冲

直播 - 首开慢 - 优化

 # 提前完成域名 -> IP 解析

URL：rtmp://live.hkstv.hk.lxdns.com/live/hks

——→ rtmp://221.230.141.131/live/hks

```
Jhuster:~ lujun$ ffplay rtmp://221.230.141.131/live/hks
ffplay version N-80238-g9887cfd Copyright (c) 2003-2016 the FFmpeg developers
  built with Apple LLVM version 8.1.0 (clang-802.0.42)
  configuration:
  libavutil      55. 24.100 / 55. 24.100
  libavcodec     57. 44.101 / 57. 44.101
  libavformat    57. 37.101 / 57. 37.101
  libavdevice    57.  0.101 / 57.  0.101
  libavfilter     6. 46.101 /  6. 46.101
  libswscale      4.  1.100 /  4.  1.100
  libswresample   2.  0.101 /  2.  0.101
[rtmp @ 0x7f929661e140] Server error: Failed to play hks; stream not found
rtmp://221.230.141.131/live/hks: Operation not permitted
```

 # 坑在哪？

CDN 服务商需要知道是 “谁” 在申请这个流 -> 计费，判断方式：domain

 # 怎么解决？

RTMP 协议中，有一个参数：tcUrl，记录的是完整的播放地址

——→ 修改播放器底层 RTMP 代码模块，把含有 domain 的 URL 填写到 tcUrl 字段即可

直播 - 首开慢 - 优化

优化媒体信息解析速度

1. 基于 ffmpeg 的播放内核

- probesize
- analyzeduration

减小这两个值，可以明显加快首开，但是 probesize 过小，可能导致媒体信息解析错误

2. 自研播放内核

- 应用场景固定，只支持和解析特定的协议和格式，缩小解析范围

直播 - 痛点

首开慢

卡顿率高

延时大

马赛克

功耗大

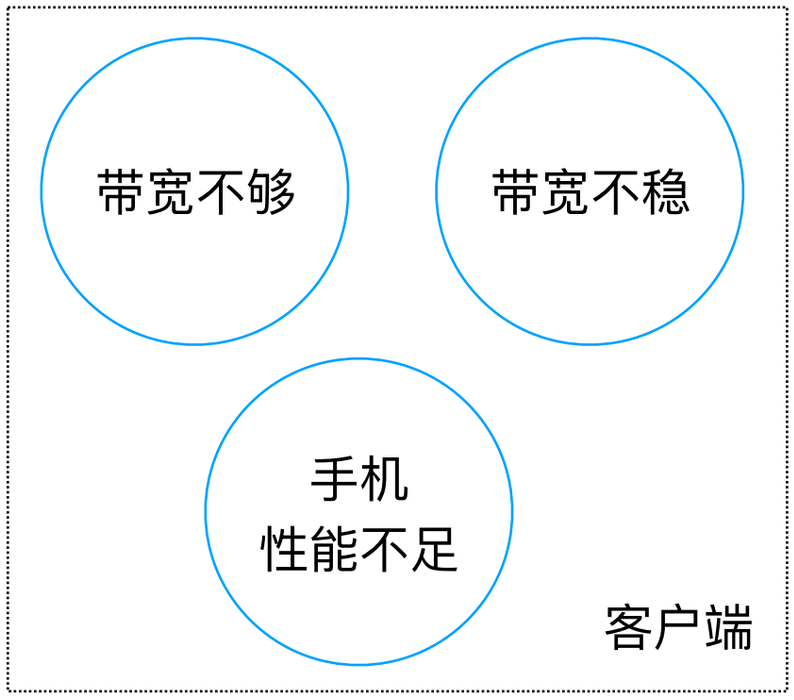
排障不易

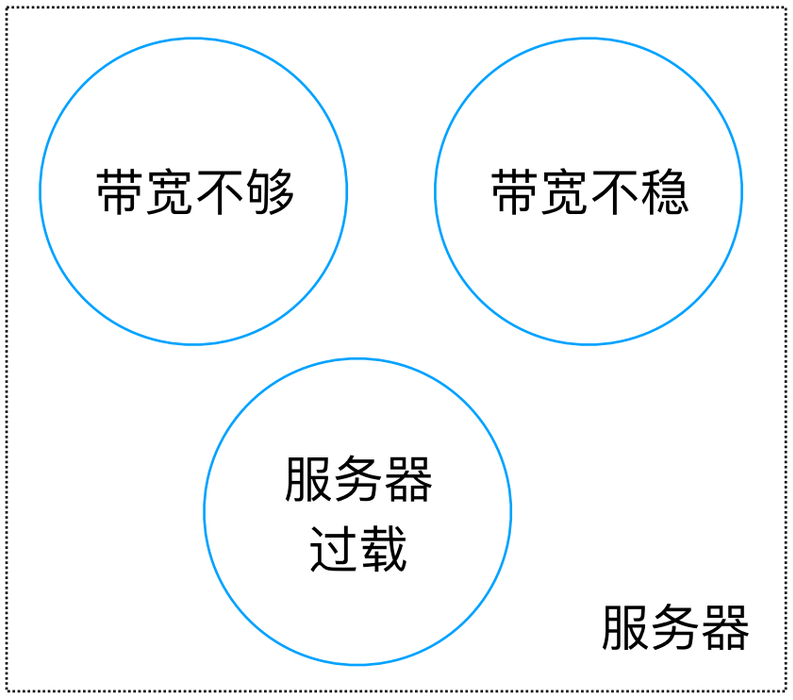
直播 - 卡顿

Q：为什么会卡顿？

由于人的视觉保持现象，要使观看者看到平滑的运动，帧率至少要达到约 **8 帧/秒**，要想让观察者完全感觉不到闪烁，则至少在 **24帧/秒** 以上。

A：播放的 “帧率太低” 或者 “帧率不稳定”





直播 - 卡顿 - 优化

客户端带宽不足

上行带宽 < 推流的码率

↓

全局卡顿

↓

动态码率

下行带宽 < 推流的码率

↓

个体卡顿

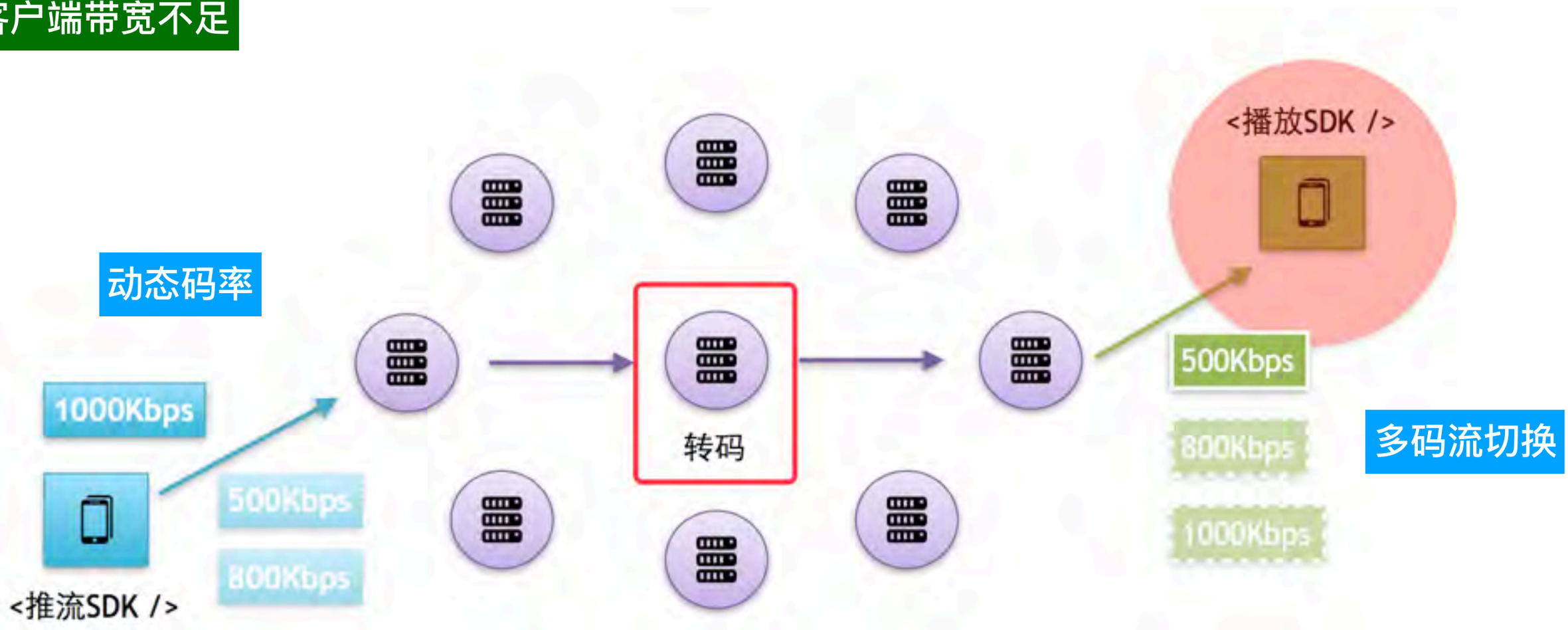
↓

多码流切换

直播 - 卡顿 - 优化

客户端带宽不足

动态码率



多码流切换

直播 - 卡顿 - 优化

客户端带宽不稳

上行带宽不稳

↓

全局卡顿

↓

发送缓冲区

下行带宽不稳

↓

个体卡顿

↓

播放缓冲区

直播 - 卡顿 - 优化

客户端性能不足

手机性能不足

↓

全局卡顿

↓

使用硬编 & 减小分辨率 & 降低帧率

手机性能不足

↓

个体卡顿

↓

使用硬解 & 多码率切换 & 主动丢帧

直播 - 卡顿 - 优化

服务端原因

带宽不足/不稳/服务过载

↓

全局卡顿

↓

质量监控 & 线路调整

直播 - 痛点

首开慢

卡顿率高

延时大

马赛克

功耗大

排障不易

直播 - 延时

Q：延时来自哪里？

A1：来自从 “主播—观众” 的网络物理传输耗时吗？

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

结论 1：物理传输延时是 ms 级别，不是直播延时的主要来源！

直播 - 延时

Q：延时来自哪里？

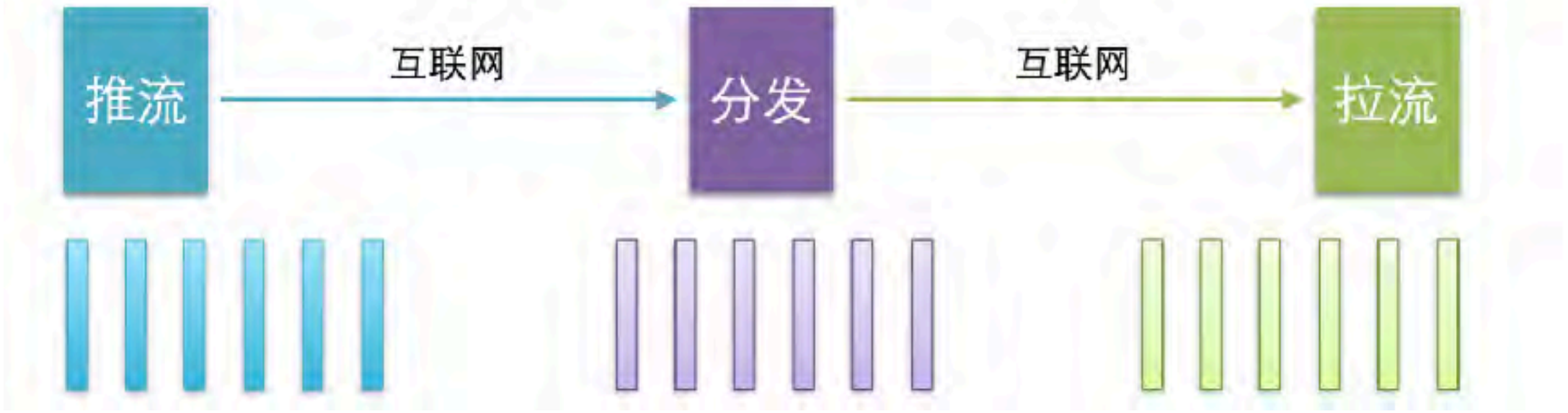
A2：来自视频采集 -> 美颜特效 -> 编码的耗时吗?

假设推流帧率：20～30 fps ⟶ **每一帧的处理延时**：33ms~50ms

结论 2：视频处理延时是 ms 级别，也不是直播延时的主要来源！

直播 - 延时

Q：延时来自哪里？

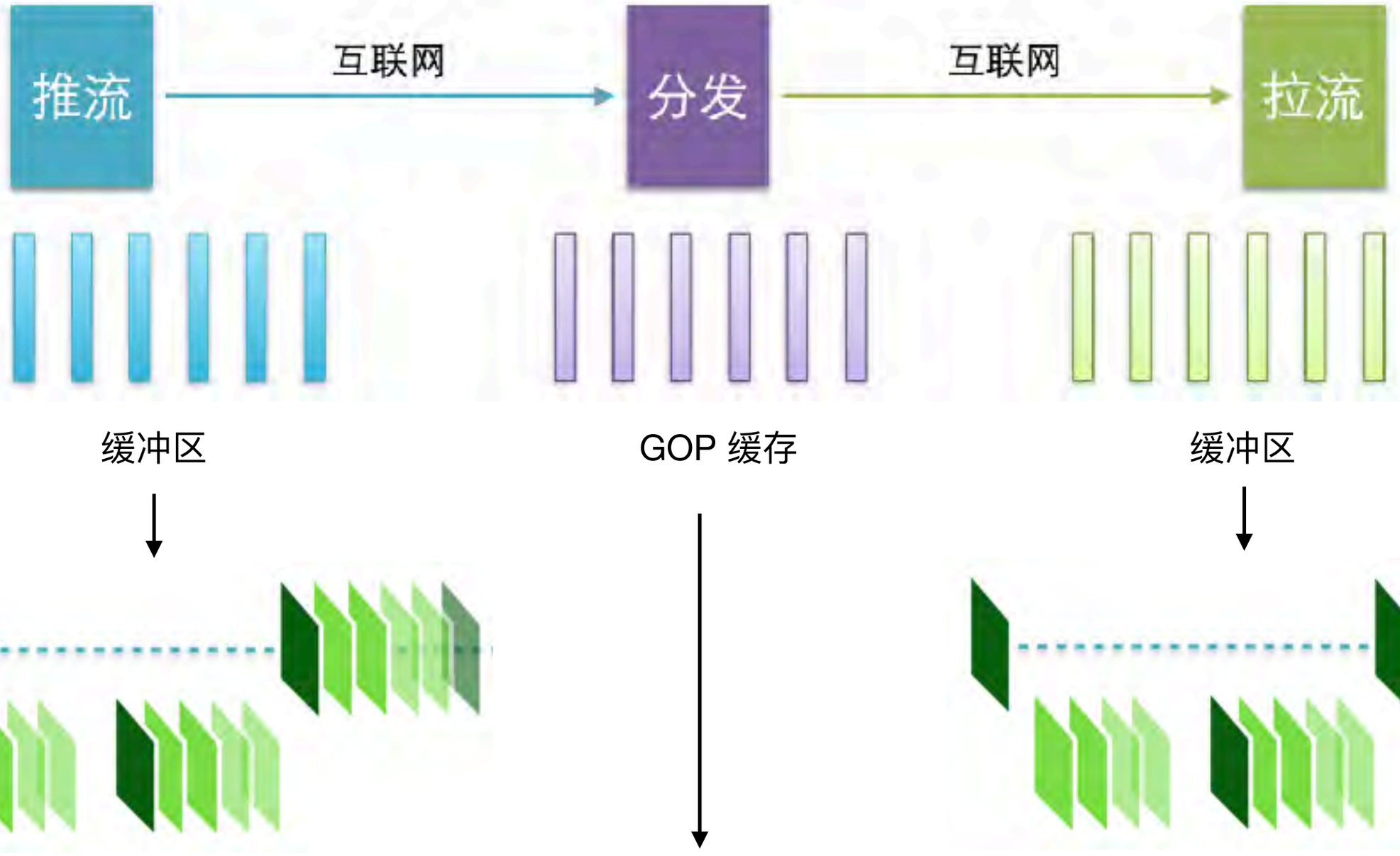


假设推流帧率：20～30 fps ⟶ 每缓冲 20~30 帧，延时增加：1s

直播的 GOP 通常设置的是 2～3s，因此服务端 GOP 缓存直接导致延时至少 2～3s！

结论：延时主要来自各业务代码中的缓冲区！

直播 - 延时 - 优化


推流
互联网
分发
互联网
拉流
缓冲区
GOP 缓存
缓冲区
编码前，主动丢帧，减少延时
减小 GOP 大小
解码后，主动丢帧，减少延时

直播 - 痛点

首开慢

卡顿率高

延时大

马赛克

功耗大

排障不易

直播 - 马赛克 - 优化

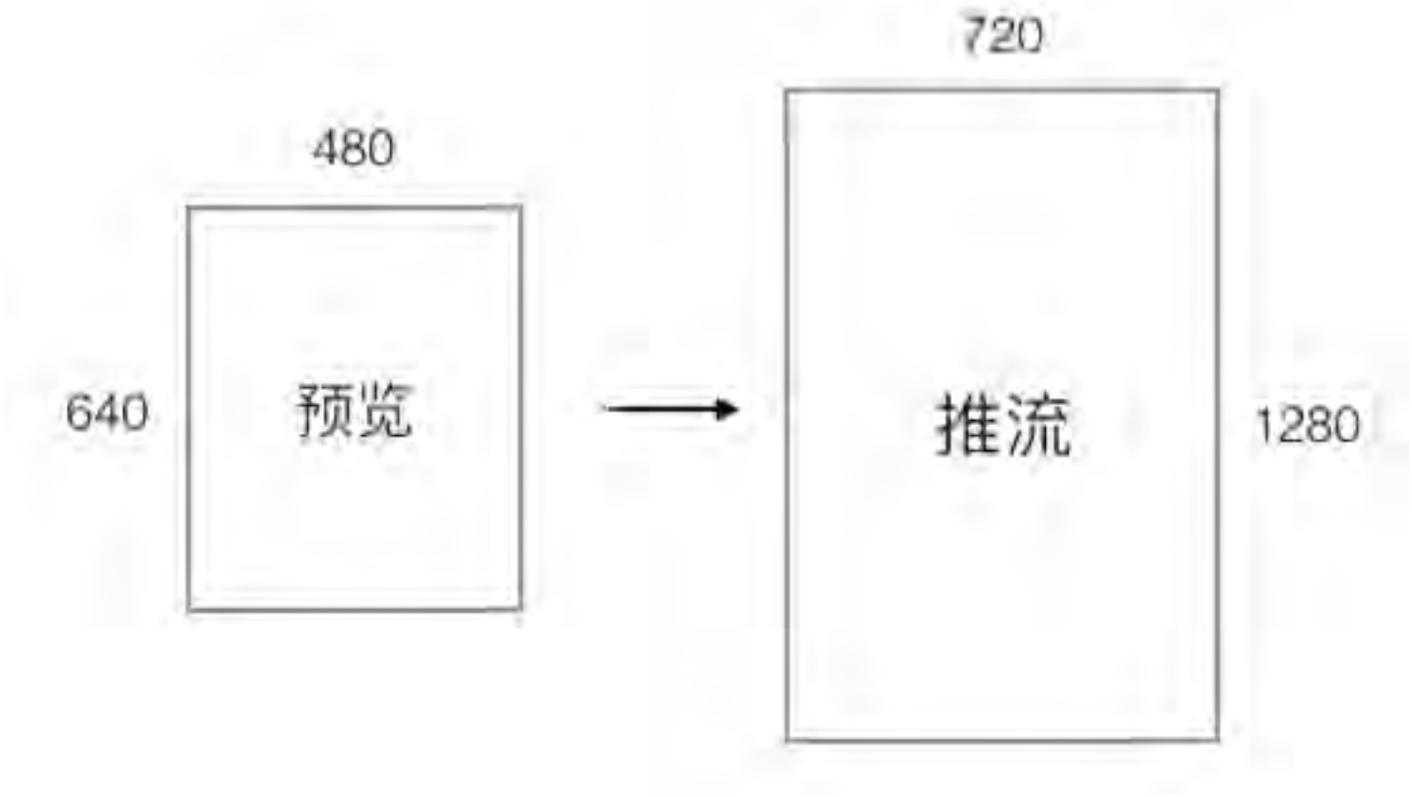
如何提高画质？

原则：减少送入编码器的数据量，降低压缩比，画面会更清晰

类 别	优 化 策 略	负 面 影 响
码率	提高码率	带宽压力变大
帧率	降低帧率	流畅度下降
GOP	增大 GOP	延时增大
画质级别	High > Main > Base Profile	功耗/延时增大
码控方式	VBR > CBR	码控不稳

直播 - 马赛克 - 优化

还有哪些优化手段？



1. 避免画面从小分辨率拉伸到大分辨率的情况出现

2. 保障良好的光照条件

直播 - 痛点

首开慢

卡顿率高

延时大

马赛克

功耗大

排障不易

直播 - 功耗大

如何降低功耗？

优化点	描 述
硬编	尽可能地使用系统硬编，功耗可以明显降低
帧率	降低帧率，可以减轻 “美颜”、“编码”、“传输” 的计算量
视频分辨率	拍摄分辨率不用太高，合适就行，比如 480P
画面剪裁	1. 尽可能保持 Camera 预览的分辨率与推流尺寸一致 2. 尽可能使用 GPU 对画面进行缩放和剪裁
格式转换	从 GPU 输出数据的时候，顺便完成格式转换，而不用再由 CPU 进行转换
OpenGL 高级特性	尽可能地使用 OpenGL 的高级特性提高效率，比如：VBO, VAO, FBO, PBO

直播 - 痛点

首开慢

卡顿率高

延时大

马赛克

功耗大

排障不易

直播 - 疑难杂症

黑屏

闪屏

花屏

马赛克

发烫

杂音

噪声

回声

线上崩溃

播放失败

音画不同步

拖动不准

直播 - 排障 - 优化

1. RTMP 的 Metadata

```
Input #0, flv, from 'rtmp://pili-publish.pilitest.qiniucdn.com/pilitest/pilitest018':
  Metadata:
    isp_name        : 中国联通
    os_version      : 10.2.1
    up_cdn_ip       : 122.193.203.197
    app_version     : 2.1.6
    gop_time        : 3000
    app_name        : com.qbox.PLMediaStreamingKitDemo
    video_encode_type: ios264
    audio_encode_type: iosaac
    network_type    : LTE
    os_platform     : iOS
    cdn_ip          : 121.15.220.3
    device_model    : iPhone 6s
    components_version: streamingkit-git-2016-11-18-5409374;librtmp-10004
    reqid           : pili1487687671
  Duration: 00:00:00.00, start: 50.193000, bitrate: N/A
    Stream #0:0: Subtitle: text
    Stream #0:1: Video: h264 (High), yuv420p, 352x640, 24 fps, 24 tbr, 1k tbn
    Stream #0:2: Audio: aac (LC), 48000 Hz, mono, fltp
```

每一个流都可以直接通过播放器拿到主播推流的环境信息，包括：

- 系统信息：设备型号、系统版本、App 版本号等
- 编码信息：软编硬编、帧率配置、GOP 参数配置等
- 网络信息：手机 ip 地址，服务器 ip 地址、WiFi 强度、运营商等

直播 - 排障 - 优化

2. 秒级服务端流数据统计

从服务端的帧率、码率变化曲线，可以准确判断卡顿问题，是否是主播网络不好导致的

- 码率 **稳定**，帧率 **降低** -> 带宽充足，帧率低可能是手机发热或者内存不足导致丢帧
- 码率 **降低**，帧率 **稳定** -> 带宽不足，动态码率生效了
- 码率 **降低**，帧率 **降低** -> 网络抖动厉害，卡顿是由于网络原因导致的

直播 - 排障 - 优化

3. 线上 Crash 收集上报 + 日志落存储

如果只有 Crash 堆栈，却没有上下文日志，线上问题排查和修复起来是比较困难的

直播 - 痛点 - 优化

首开慢

卡顿率高

延时大

马赛克

功耗大

排障不易

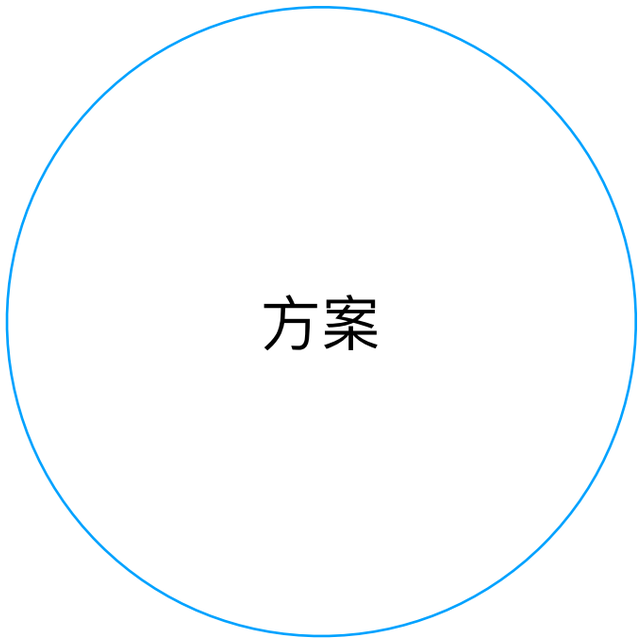
直播 - 高级功能 - 连麦

进入直播间,找到连线按钮

主播将收到连线请求，
选择连线对象，点击“连线”按钮

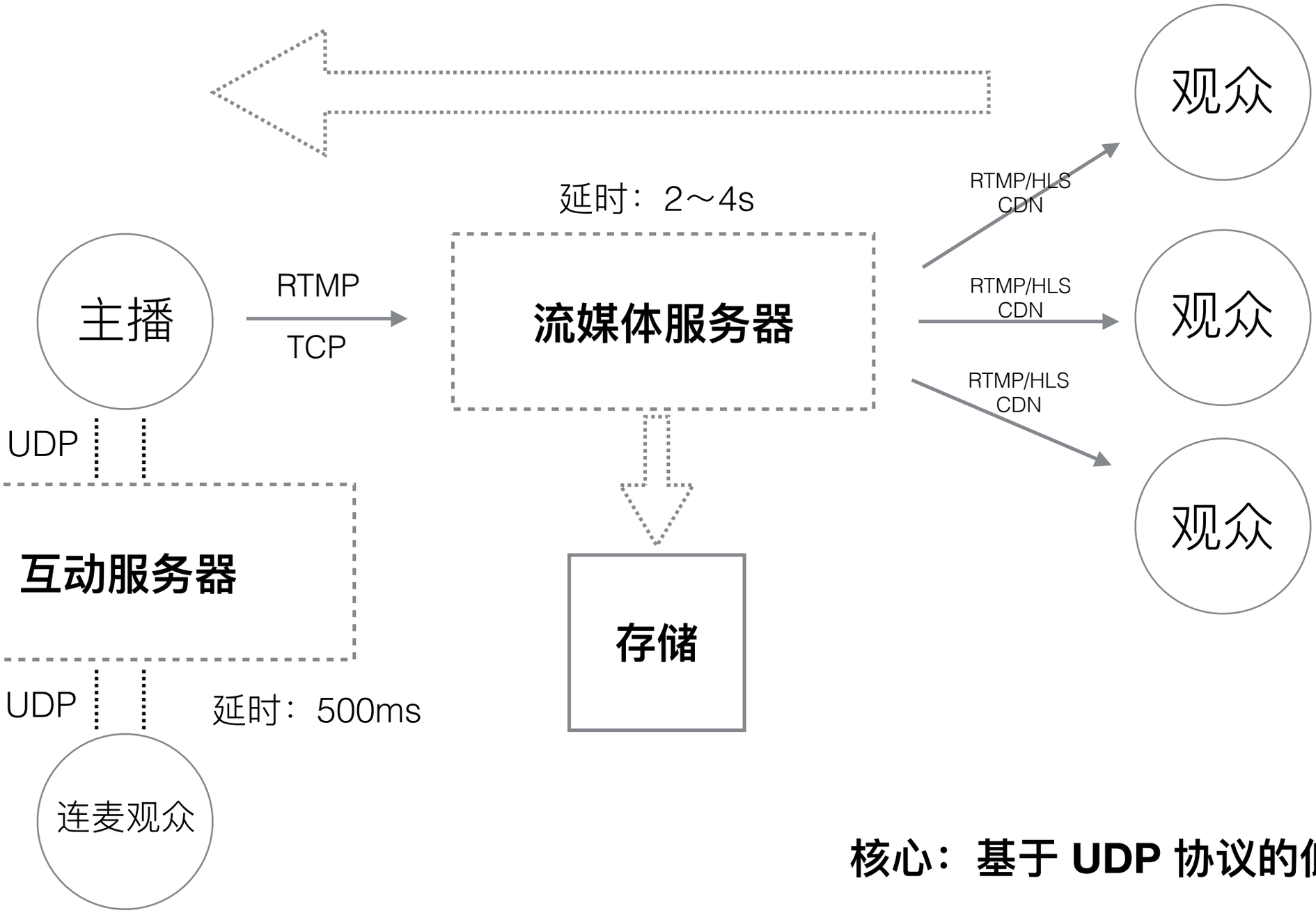
观众接受连线,连线成功

直播 - 高级功能 - 连麦



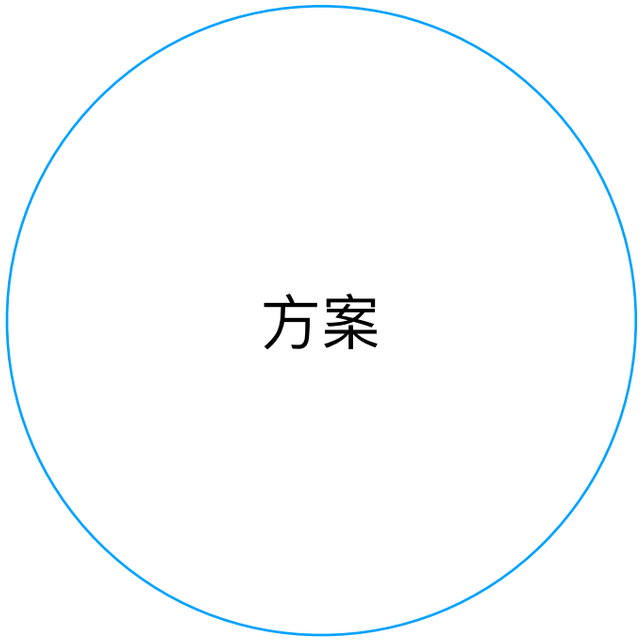
直播 - 高级功能 - 连麦

直播 - 高级功能 - 连麦

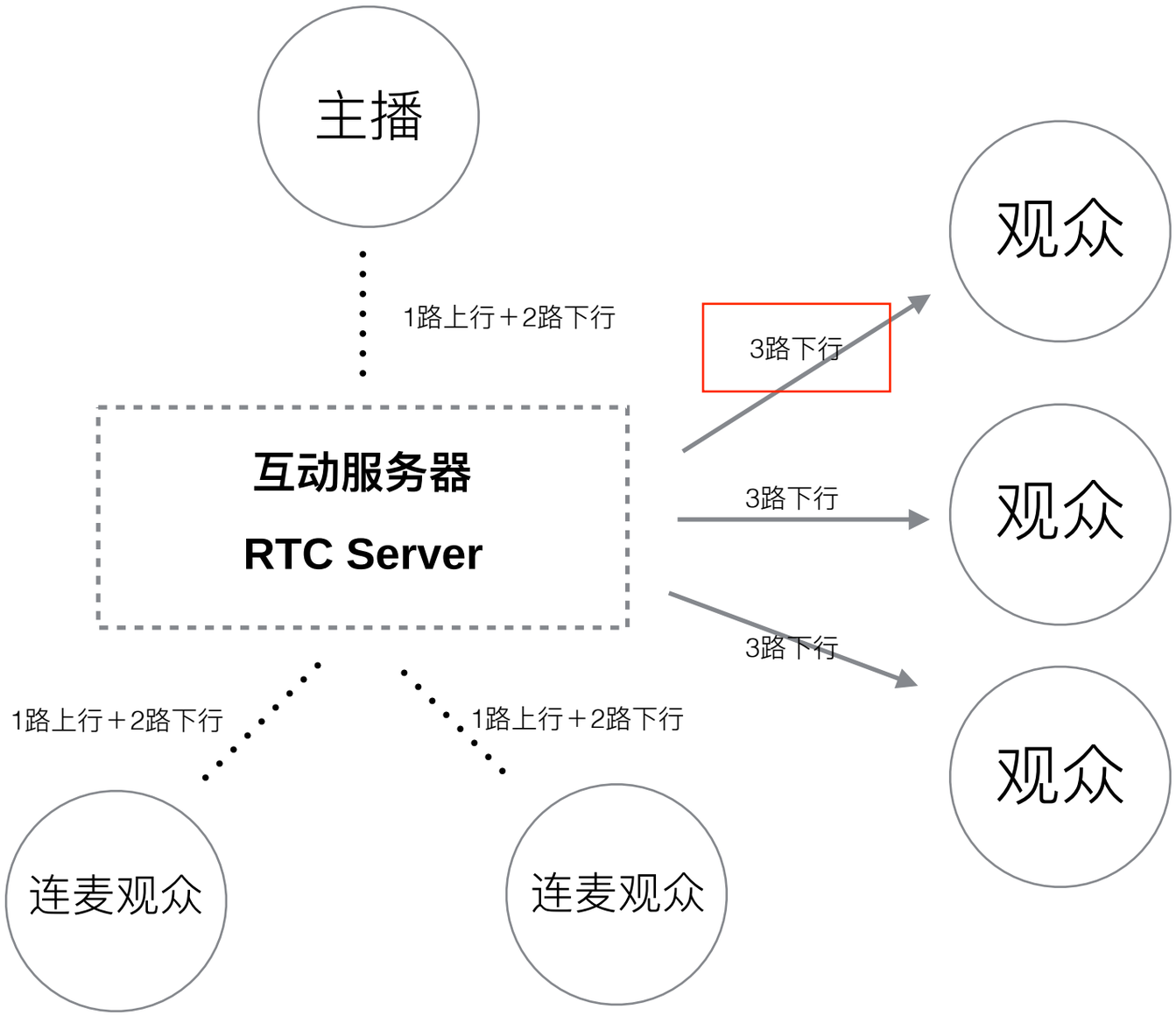


核心：基于 UDP 协议的低延时方案

直播 - 高级功能 - 连麦



连麦 - 无合流方案



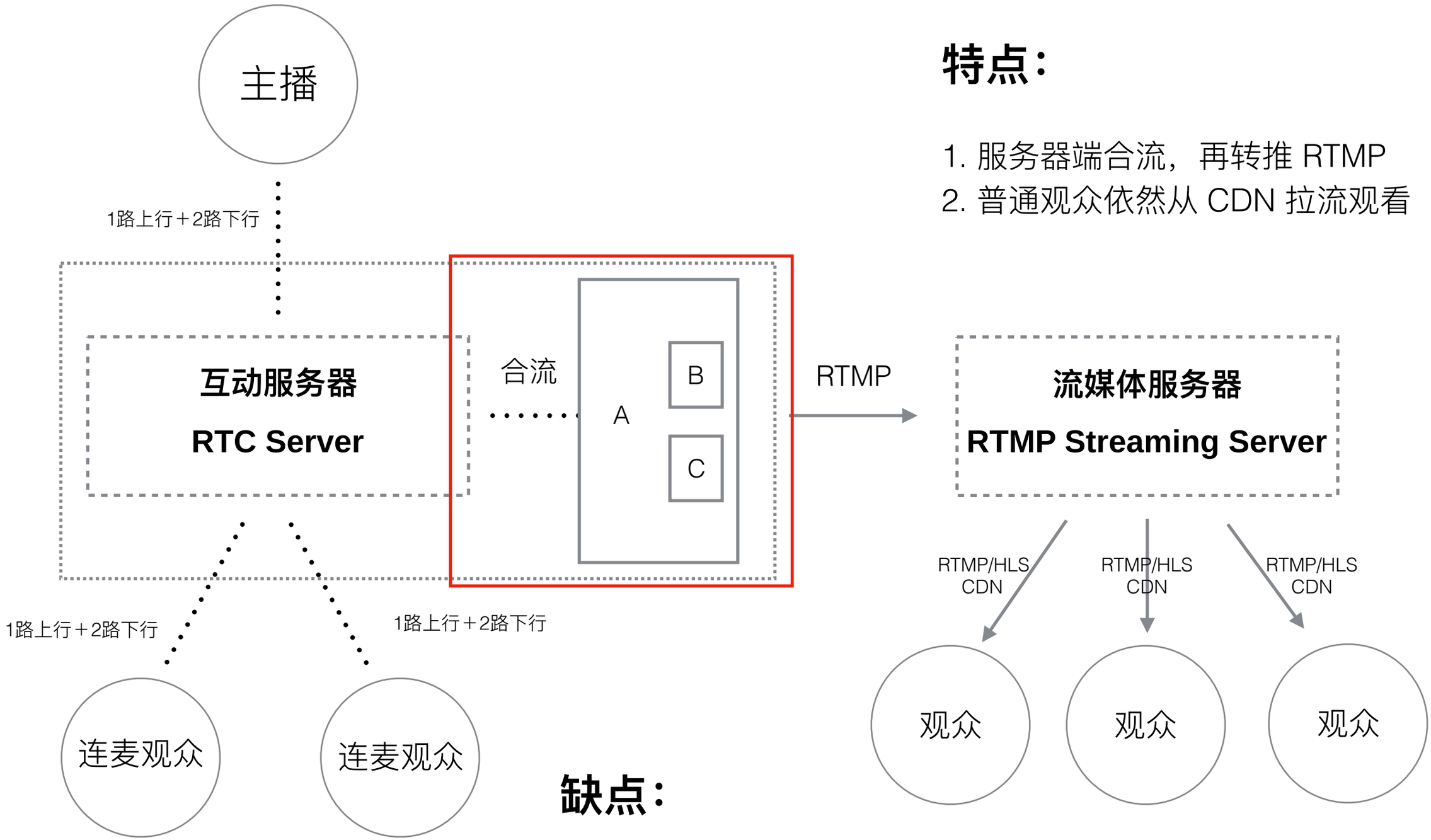
特点：

1. 所有的人都是视频会议的一部分
2. 抛弃传统的 RTMP 协议，只使用互动协议

缺点：

1. 抛弃了成熟的内容分发网络(CDN)
2. 无合流，连麦画面无法存储回放
3. 无合流，普通观众端带宽压力大，需要同时拉取所有跟主播连麦的其它观众的画面

连麦 - 服务端合流方案



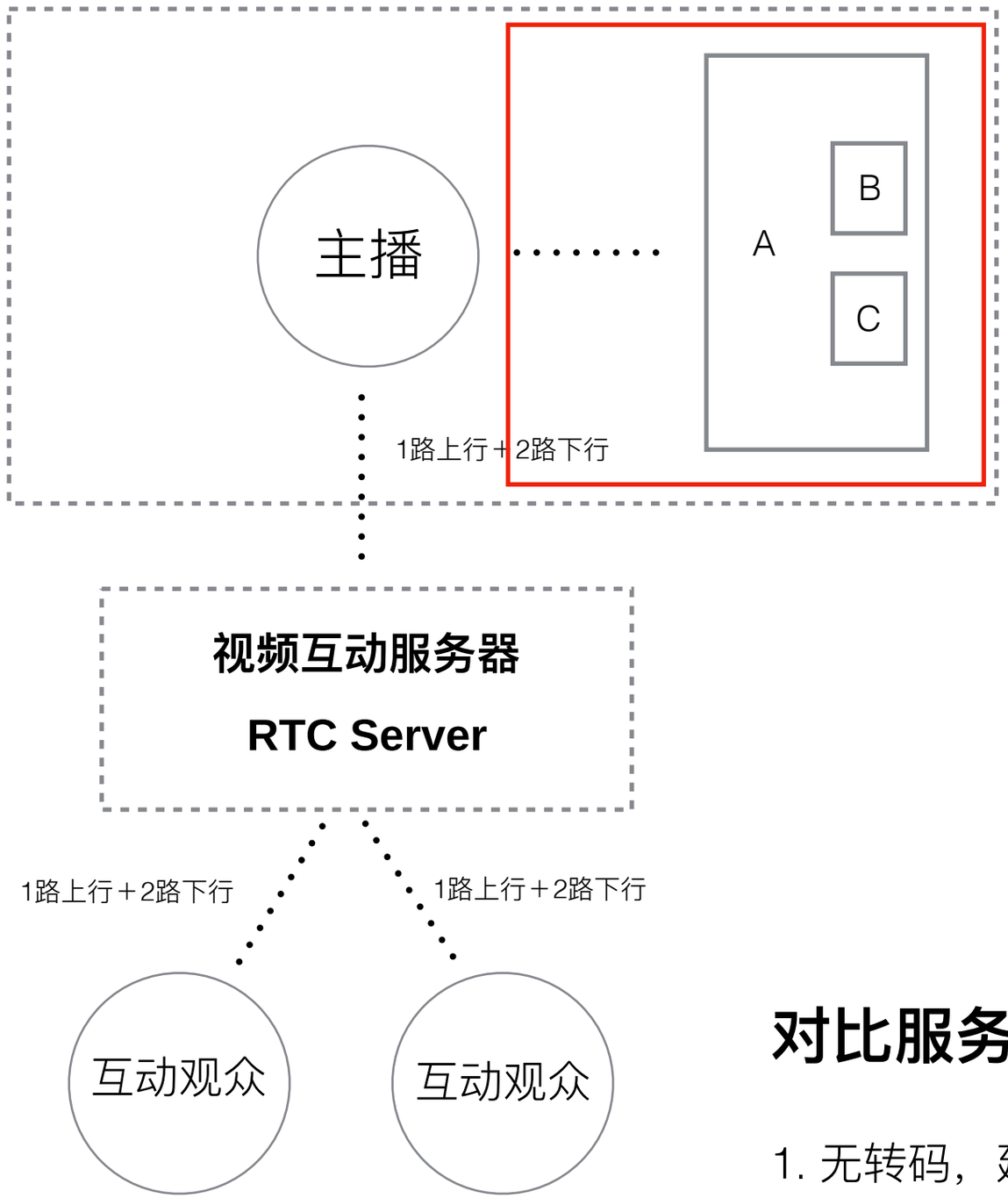
特点：

1. 服务器端合流，再转推 RTMP
2. 普通观众依然从 CDN 拉流观看

缺点：

1. 服务器转码，延时比较大，6～10s 左右
2. 服务器压力大，性能要求极高，扩展成本也高

连麦 - 客户端合流方案



对比服务器合流方案：

1. 无转码，延时小
2. 无缝合流，服务端无感知

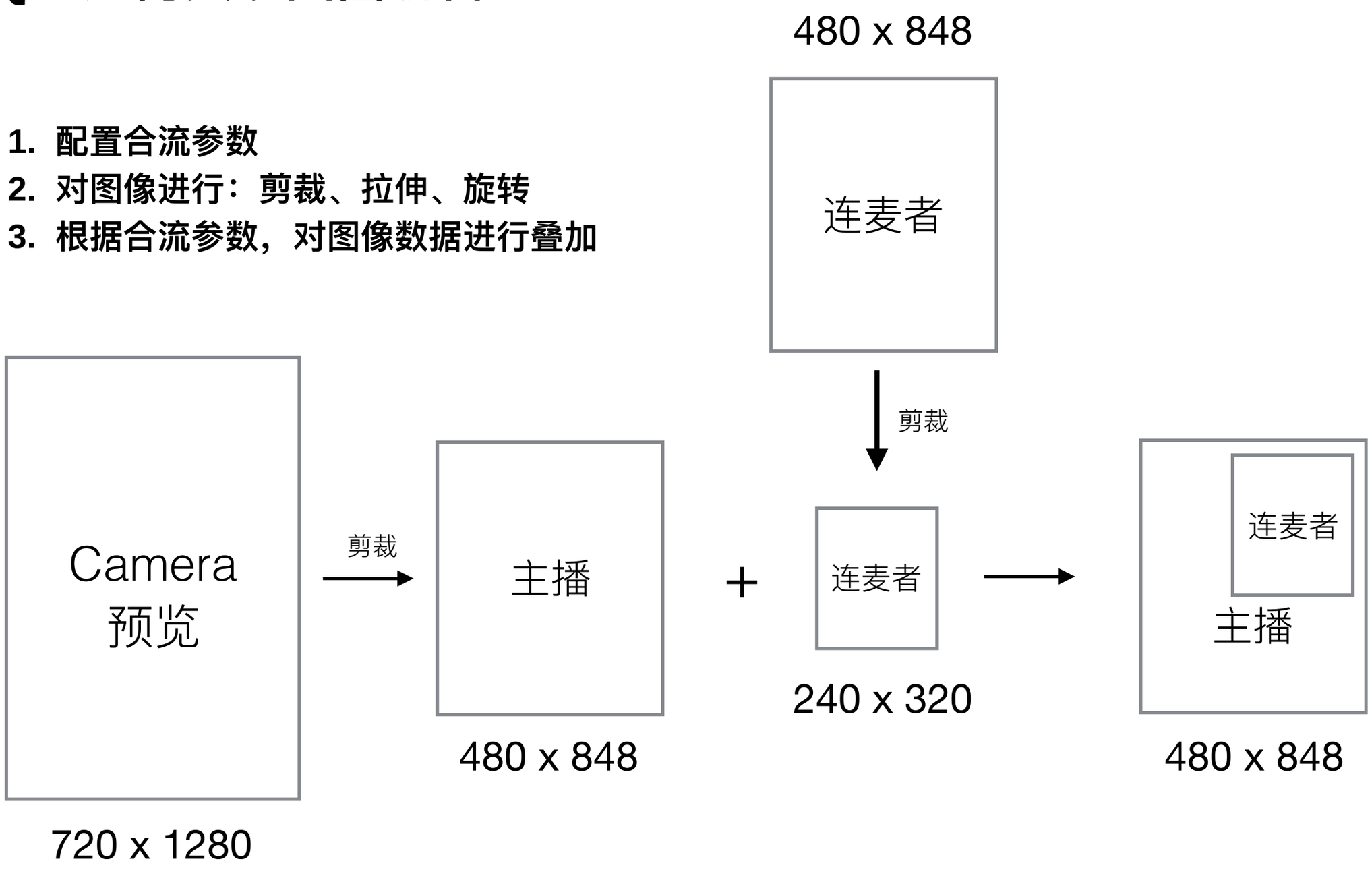
缺点：

1. 主播端带宽和功耗压力非常大

连麦 - 合流原理 - 视频

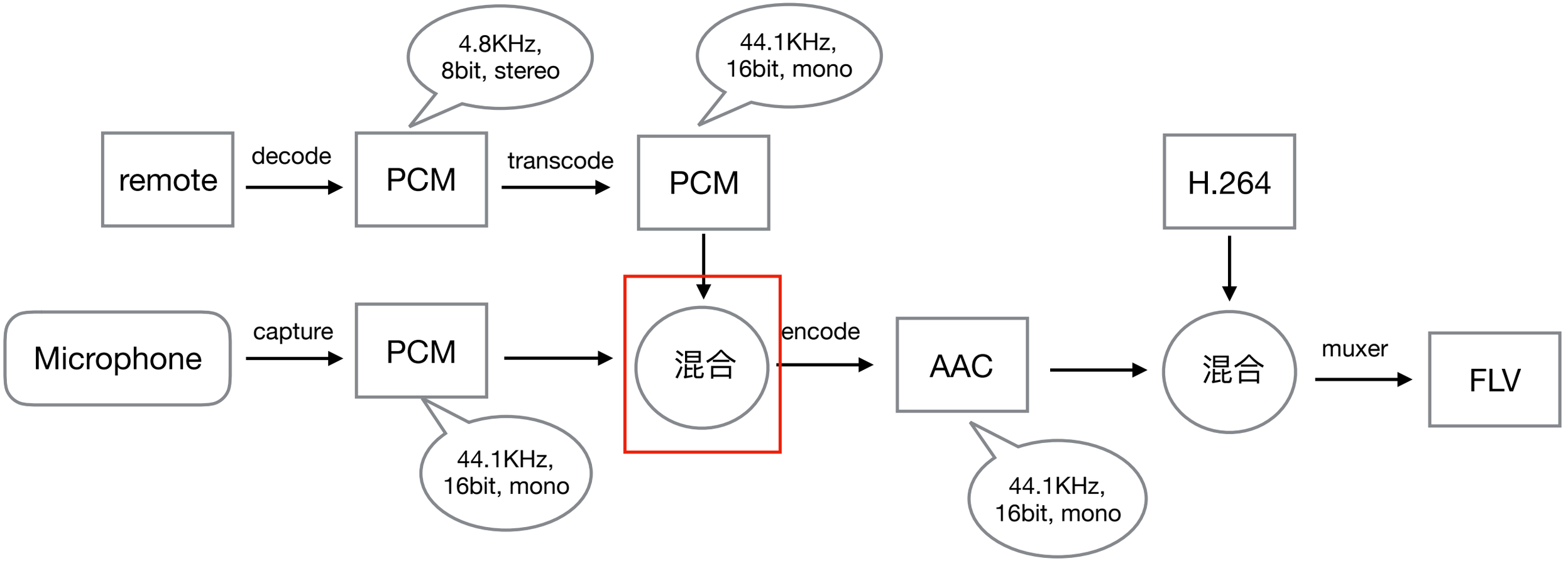
Q：如何实现画面混合？

1. 配置合流参数
2. 对图像进行：剪裁、拉伸、旋转
3. 根据合流参数，对图像数据进行叠加

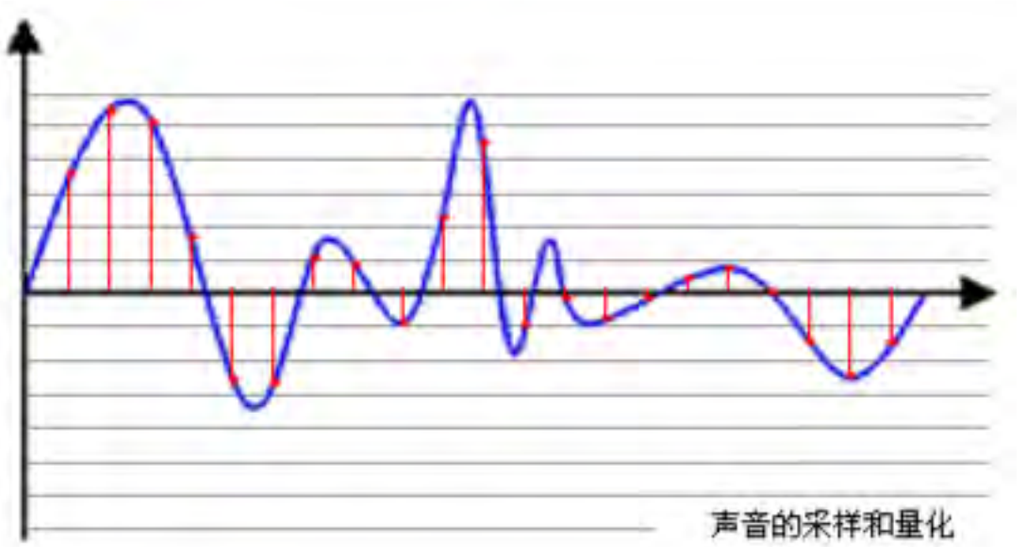


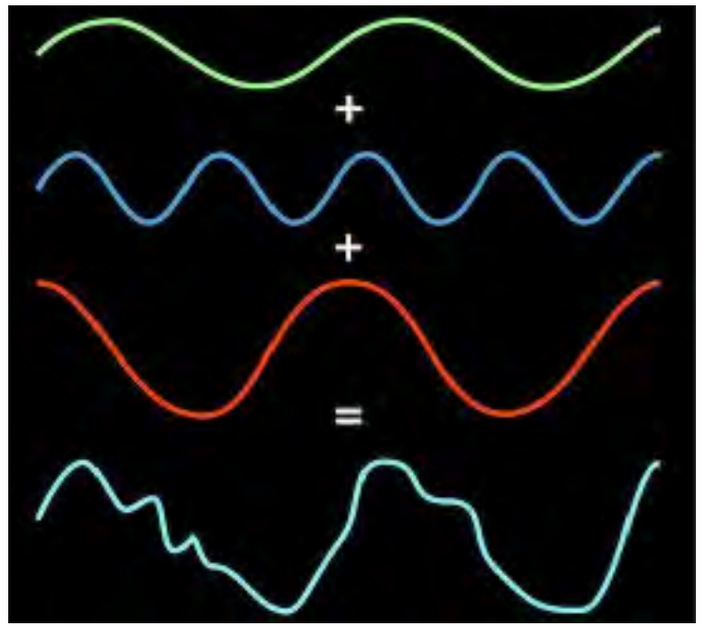
连麦 - 合流原理 - 音频

Q：如何实现混音功能？



连麦 - 合流原理 - 音频





1. 重采样（resample）
2. 对位叠加
3. 溢出处理

```
static void audio_mixer_mix(short *mix_frame, short *frame, int frame_size)
{
    for (int i = 0; i < frame_size; i++) {
        int mixed = mix_frame[i] + frame[i];
        if (mixed > 32767) {
            mixed = 32767;
        } else if (mixed < -32768) {
            mixed =  -32768;
        }
        mix_frame[i] = (short) mixed;
    }
}
```

演讲大纲

1. 直播的痛点介绍

2. 技术层面的关键优化

- 秒开优化
- 卡顿优化
- 延时优化
- 清晰度优化
- 功耗优化
- 排障的优化

3. 直播部分高级功能技术剖析

- 连麦方案
- 合流原理

4. 总结

移动直播的关键技术优化

“哪有什么岁月静好 只不过是有人替你负重前行。”

联系方式

主页：http://www.jhuster.com

邮箱：lujun.hust@gmail.com

微信：weixin_lujun

公众号：@Jhuster