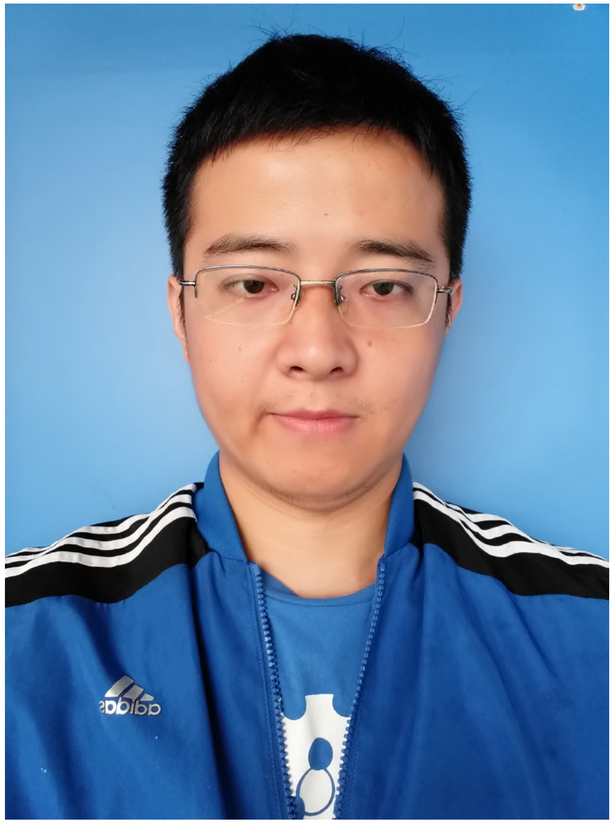
TLC 2017 EXCHANGE INNOVATION 交流 · 创新

全民K歌歌房技术实践

梅江霞(tigermei)

个人介绍

2013年加入QQ音乐团队，先后负责QQ音乐性能优化和微云pc端的技术管理。
最近1年负责全民K歌直播和歌房技术搭建工作，先后进行了全民K歌直播间性能优化，主导了K歌房核心技术设计和功能落地。



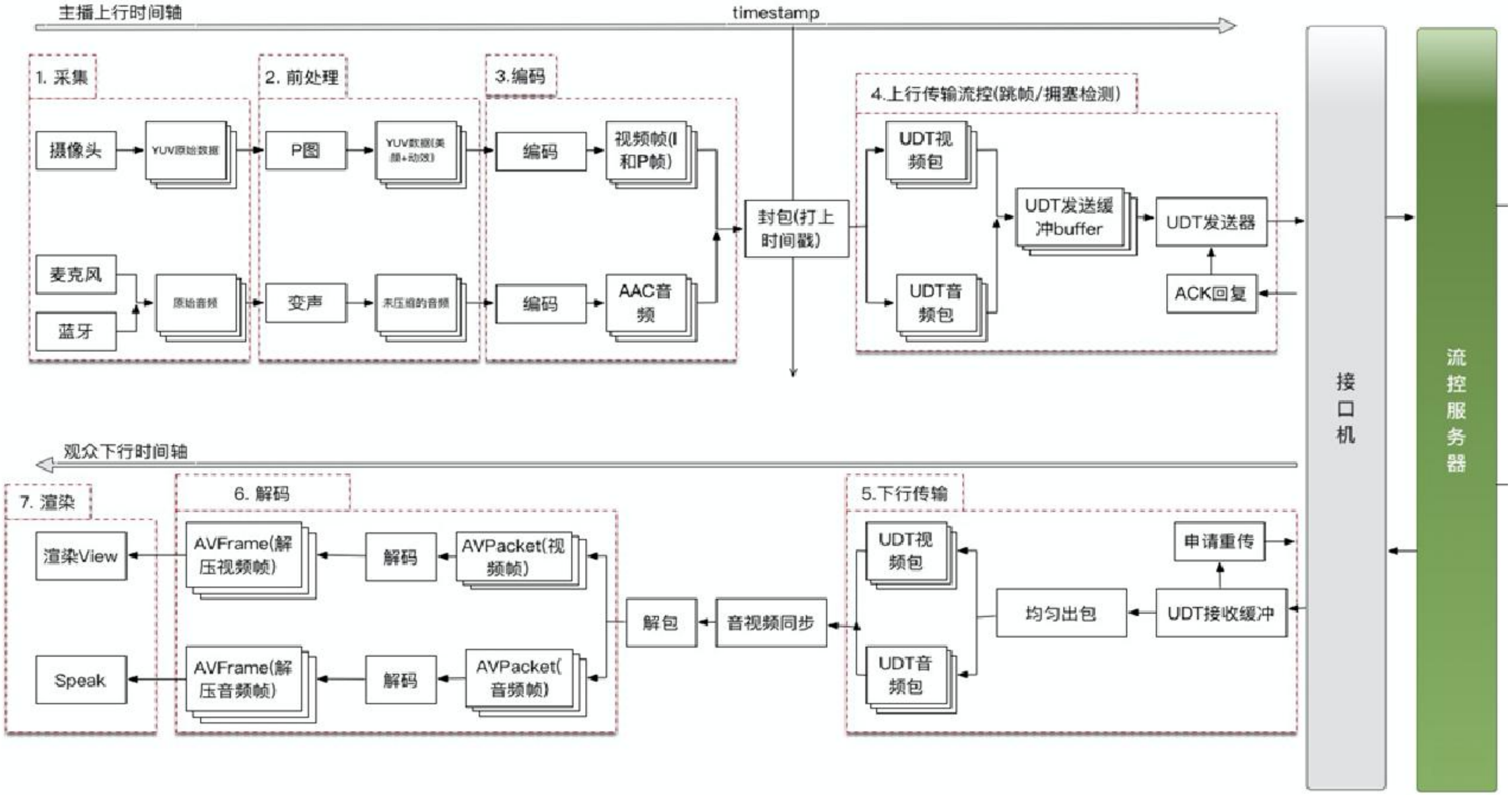
内容大纲

全民K歌歌房项目介绍

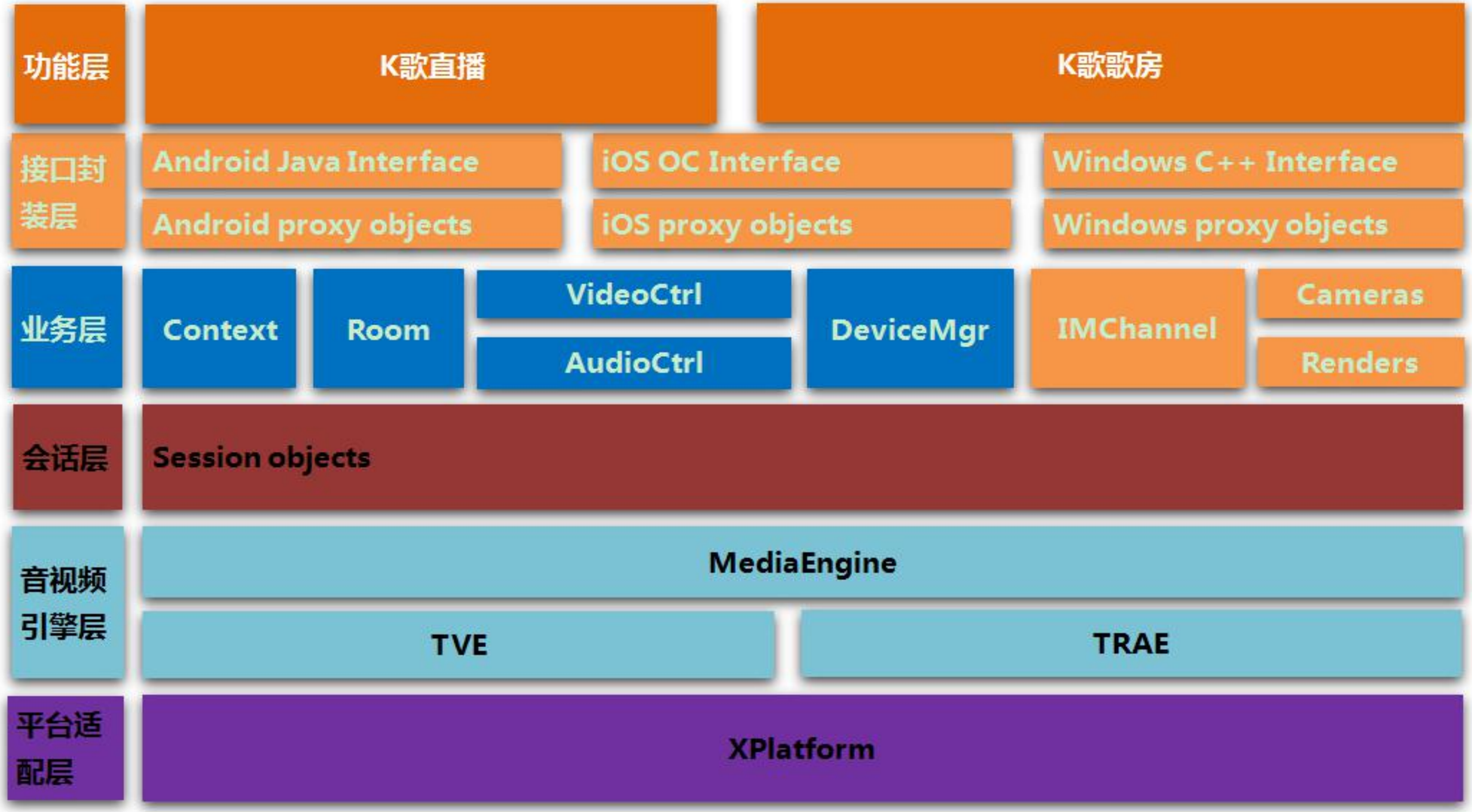
项目概述：

全民K歌在线KTV是将线下的KTV的体验搬到线上，除了竞品传统的独唱的玩法外，我们还首创性地推出在线实时合唱的玩法。

歌房框架——音视频处理流程



歌房框架——整体框架


功能层
K歌直播
K歌歌房
接口封装层
Android Java Interface
iOS OC Interface
Windows C++ Interface
Android proxy objects
iOS proxy objects
Windows proxy objects
业务层
Context
Room
VideoCtrl
AudioCtrl
DeviceMgr
IMChannel
Cameras
Renders
会话层
Session objects
音视频引擎层
MediaEngine
TVE
TRAE
平台适配层
XPlatform

歌房核心功能实践

化繁为简——打造高稳定性的歌房服务

- 完善、多维度的监控平台
- 稳定、容错容灾能力强的麦序服务

精细分析——解决双人合唱人声/伴奏/歌词同步问题

- 唱歌角色的人声和伴奏的偏移在20ms以内
- 歌词进度逐字对齐

云端调控——增强歌房对网络的抗性

化繁为简——打造高稳定性的歌房服务

稳定的歌房服务——直播与歌房对比

1.直播与歌房功能对比

直播	歌房
以主播为中心	没有中心，房间任何成员都可能成为中心
主播1人在麦上	多人轮流上麦
主播进退房间上下麦	多人频繁上下麦
观众请求主播音视频	观众轮流请求不同麦上人的音视频
无合唱	有合唱
无语音席	有语音席位

2.直播与歌房功能复杂度对比

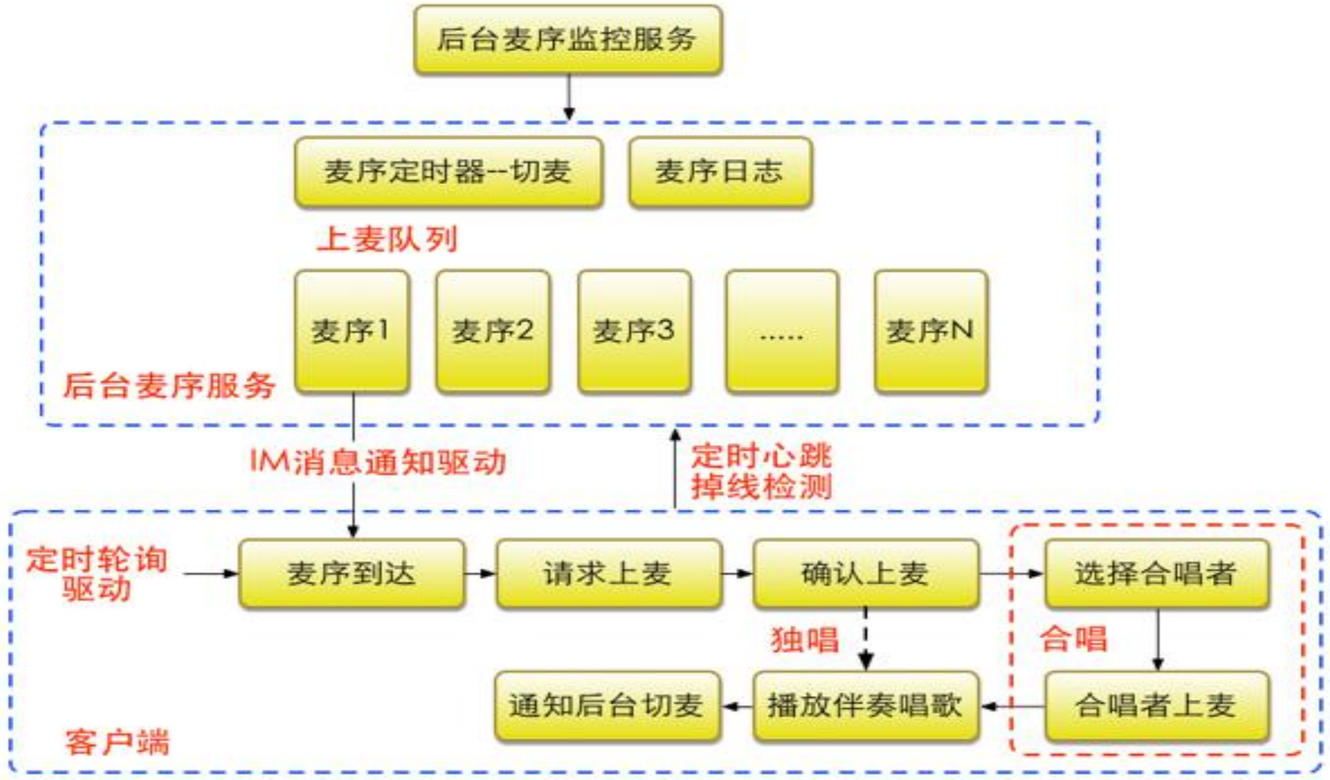
直播	歌房
2种角色	5种角色
一次上麦最多4步操作	一次上麦最多15步操作
3种互斥操作	7种互斥操作
角色最多2种状态	角色最多5种状态

复杂度是连麦直播的4-8倍， ***歌房比直播复杂很多***

稳定的歌房服务——两个核心交互原则

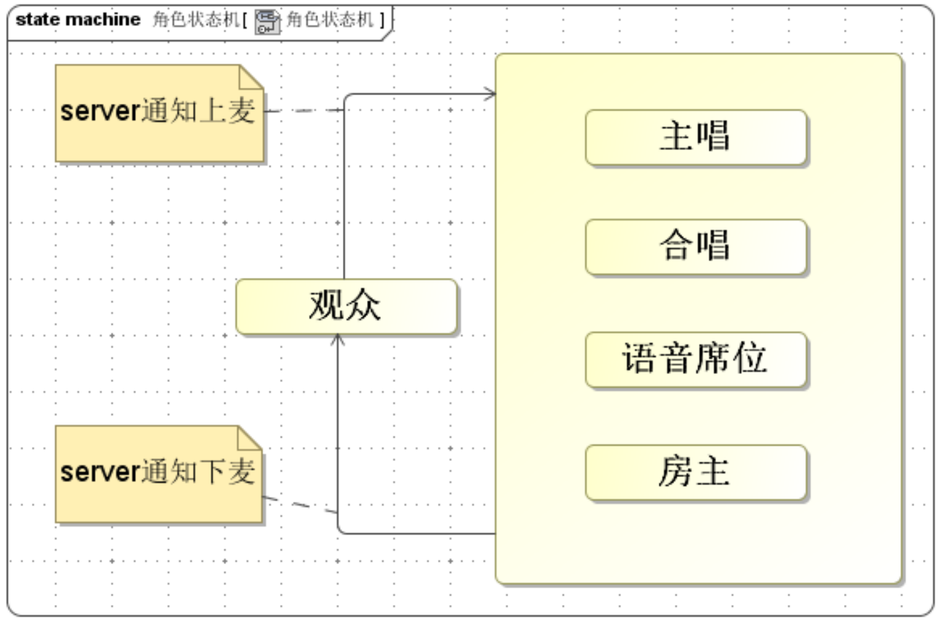
核心前后台交互原则：

- ✧ *后台保持当前状态，客户端通过推拉两种方式结合来驱动客户端状态变换*
- ✧ *客户端通过完善的过滤机制和状态机来规避服务器推送的各种异常状态*



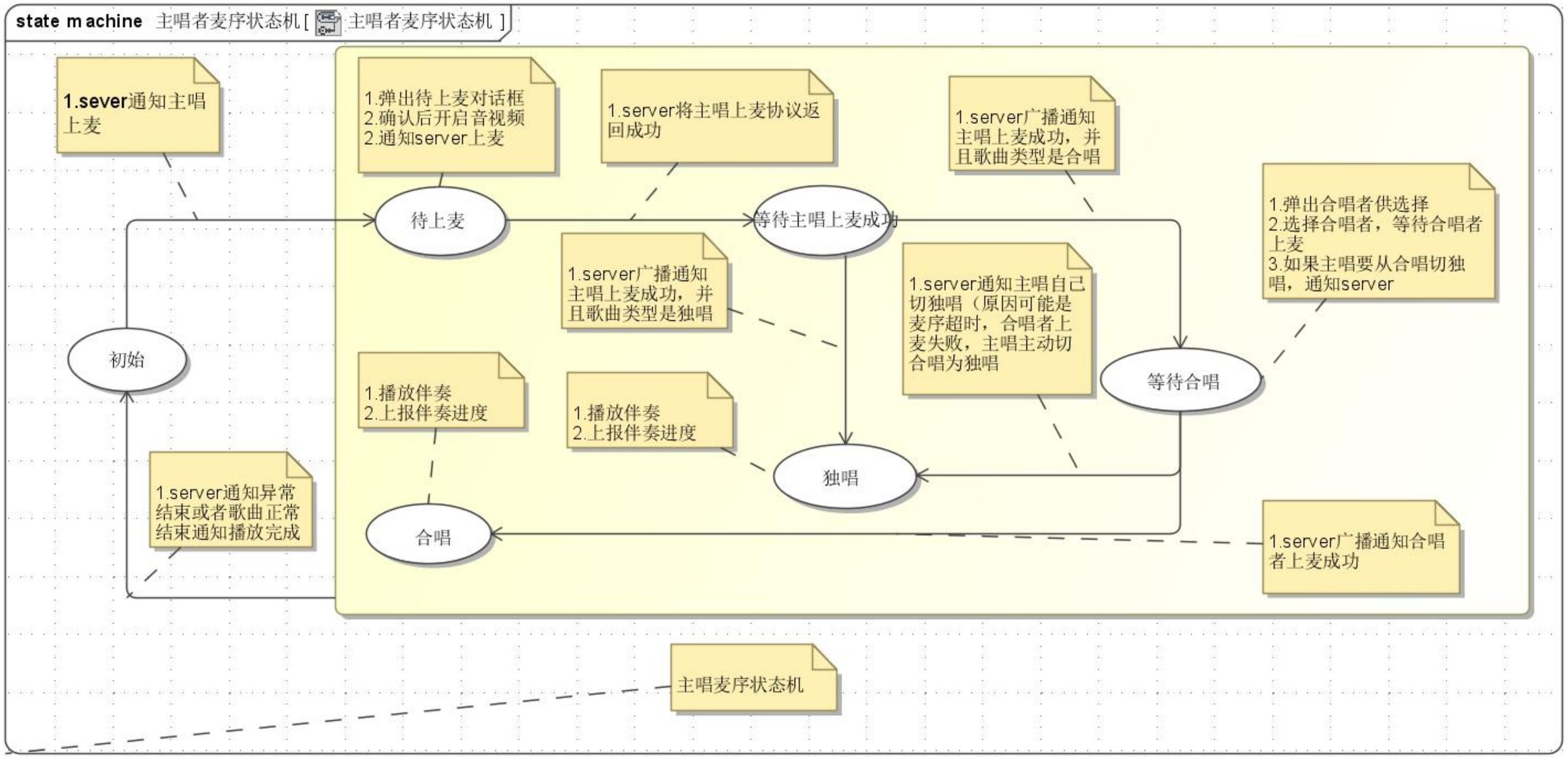
稳定的歌房服务——统一的过滤机制

稳定的歌房服务——状态机

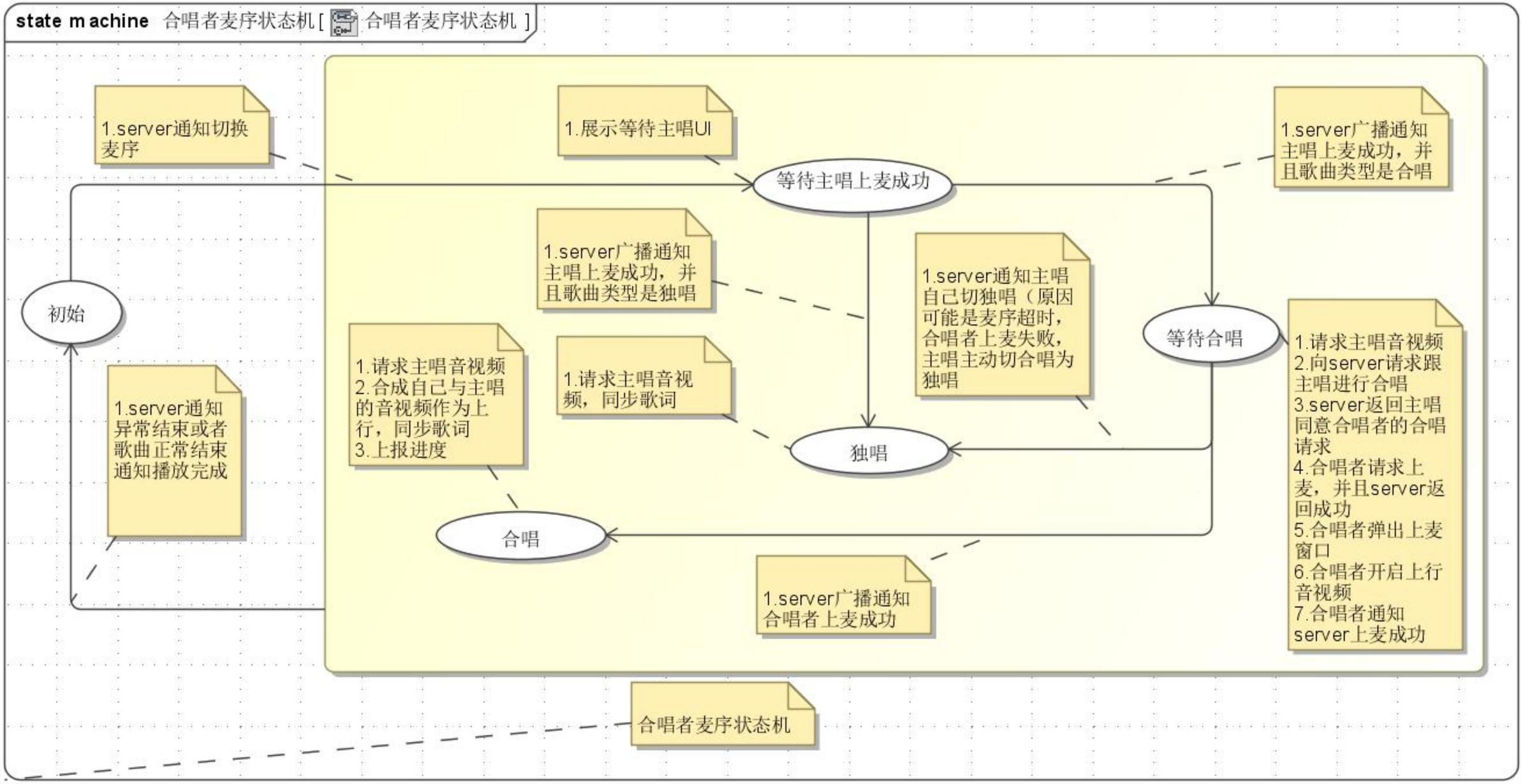


1. *客户端角色切换和状态的变化是通过server通知*
2. *Server保持当前状态，Server状态通过客户端的行为来更新。*
3. *Server状态更新通过推拉两种方式通知客户端*

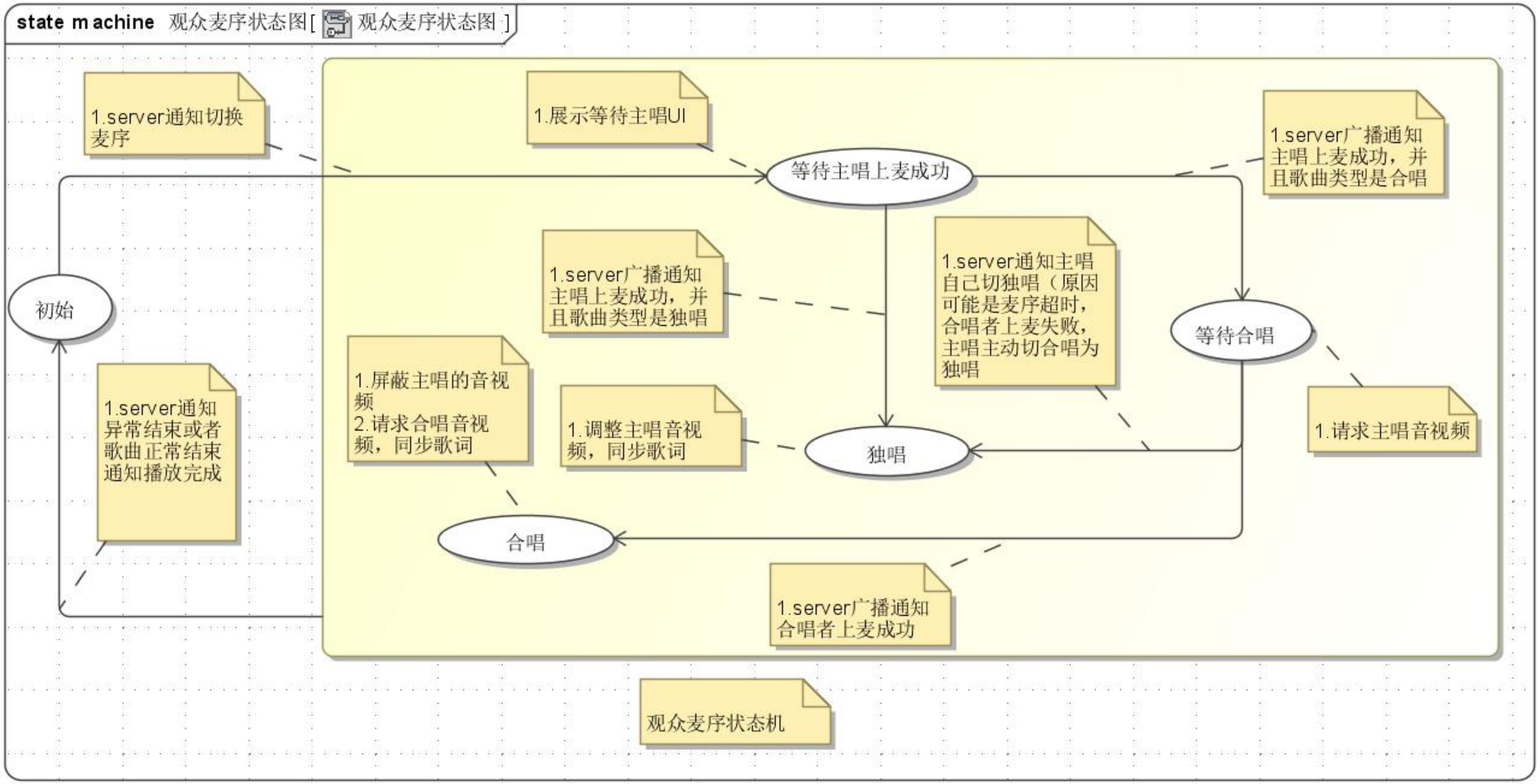
稳定的歌房服务——状态机


state machine 主唱者麦序状态机 [主唱者麦序状态机]
1.sever通知主唱上麦
1.弹出待上麦对话框
2.确认后开启音视频
2.通知server上麦
1.server将主唱上麦协议返回成功
1.server广播通知主唱上麦成功，并且歌曲类型是合唱
1.弹出合唱者供选择
2.选择合唱者，等待合唱者上麦
3.如果主唱要从合唱切独唱，通知server
待上麦
等待主唱上麦成功
初始
1.server广播通知主唱上麦成功，并且歌曲类型是独唱
1.server通知主唱自己切独唱（原因可能是麦序超时，合唱者上麦失败，主唱主动切合唱为独唱
等待合唱
1.播放伴奏
2.上报伴奏进度
1.播放伴奏
2.上报伴奏进度
独唱
1.server通知异常结束或者歌曲正常结束通知播放完成
合唱
1.server广播通知合唱者上麦成功
主唱麦序状态机

稳定的歌房服务——状态机


state machine 合唱者麦序状态机 [合唱者麦序状态机]
1.server通知切换麦序
1.展示等待主唱UI
1.server广播通知主唱上麦成功，并且歌曲类型是合唱
等待主唱上麦成功
1.server广播通知主唱上麦成功，并且歌曲类型是独唱
1.server通知主唱自己切独唱（原因可能是麦序超时，合唱者上麦失败，主唱主动切合唱为独唱
初始
等待合唱
1.请求主唱音视频
2.向server请求跟主唱进行合唱
3.server返回主唱同意合唱者的合唱请求
4.合唱者请求上麦，并且server返回成功
5.合唱者弹出上麦窗口
6.合唱者开启上行音视频
7.合唱者通知server上麦成功
1.请求主唱音视频
2.合成自己与主唱的音视频作为上行，同步歌词
3.上报进度
1.请求主唱音视频，同步歌词
独唱
1.server通知异常结束或者歌曲正常结束通知播放完成
合唱
1.server广播通知合唱者上麦成功
合唱者麦序状态机

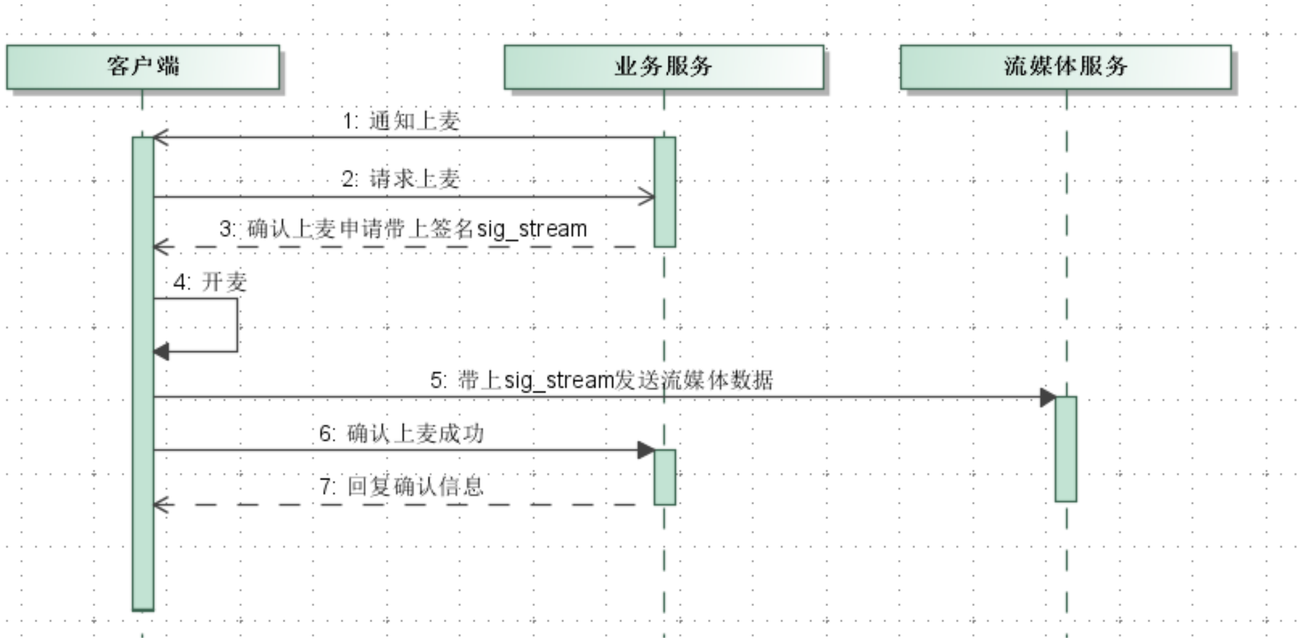
稳定的歌房服务——状态机


state machine 观众麦序状态图[观众麦序状态图]
1.server通知切换麦序
1.展示等待主唱UI
等待主唱上麦成功
1.server广播通知主唱上麦成功，并且歌曲类型是合唱
初始
1.server广播通知主唱上麦成功，并且歌曲类型是独唱
1.server通知主唱自己切独唱（原因可能是麦序超时，合唱者上麦失败，主唱主动切合唱为独唱
等待合唱
1.server通知异常结束或者歌曲正常结束通知播放完成
1.屏蔽主唱的音视频
2.请求合唱音视频，同步歌词
1.调整主唱音视频，同步歌词
独唱
1.请求主唱音视频
合唱
1.server广播通知合唱者上麦成功
观众麦序状态机

稳定的歌房服务——安全地上麦

异常上麦	解决方案
server通知上麦信息丢失	客户端定时轮询
server通知上麦信息重复	状态机逻辑去重
客户端状态异常	server保持状态恢复
协议延迟或者失败	客户端定时轮询
不合理的时机上麦	XXXXXX
异常上麦到其他房间	XXXXXX
被第三方破解协议	XXXXXX
后端异常推送上麦	XXXXXX
观众上麦后一直不下麦	XXXXXX

稳定的歌房服务——安全地上麦



7步上麦	解决的上麦异常
1. 业务server通知客户端上麦	可以排除无关麦序和当前房间的人异常上麦
2. 客户端向业务server请求上麦	可以排除掉消息延迟或者错乱导致的异常上麦
3. 业务server成功返回，并且带上签名sig_stream	
4. 开麦	可以排除客户端硬件异常的上麦
5. 带上sig_stream发送流媒体数据	可以排除掉非业务允许的异常上麦，破解掉app的异常上麦可以在这一步被阻止
6. 向业务server确认上麦成功	可以排除绕过各种条件后异常上麦后，观众会请求非法的音视频。也可以排除掉上一个上麦观众不响应业务server通知，迟迟不下麦的场景
7. 业务server返回确认信息	可以排除绕过各种条件后异常上麦后，非法观众还持续地上行音视频

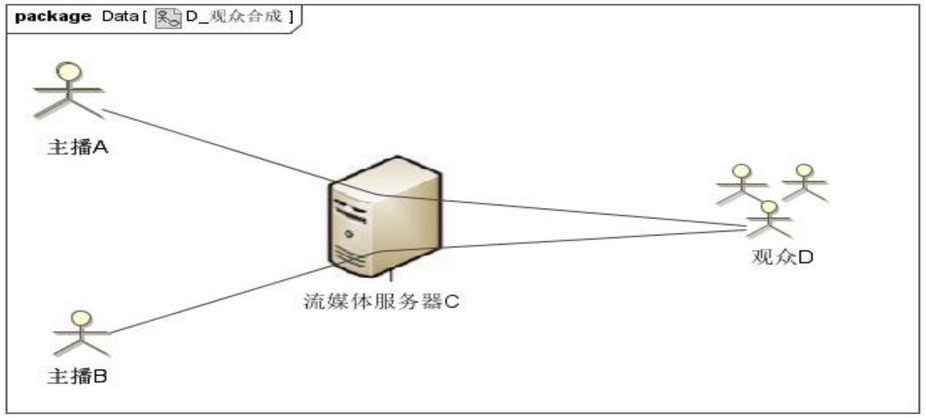
稳定的歌房服务——安全地上麦

异常上麦	解决方案
server通知上麦信息丢失	客户端定时轮询
server通知上麦信息重复	状态机逻辑去重
协议延迟或者失败	客户端定时轮询
不合理的时机上麦	第2步和第5步
异常上麦到其他房间	第5步
被第三方破解协议	1. 观众端只接收合理的音视频数据 2. 第5步
后端异常推送上麦	第2步和第5步
观众上麦后一直不下麦	第6步

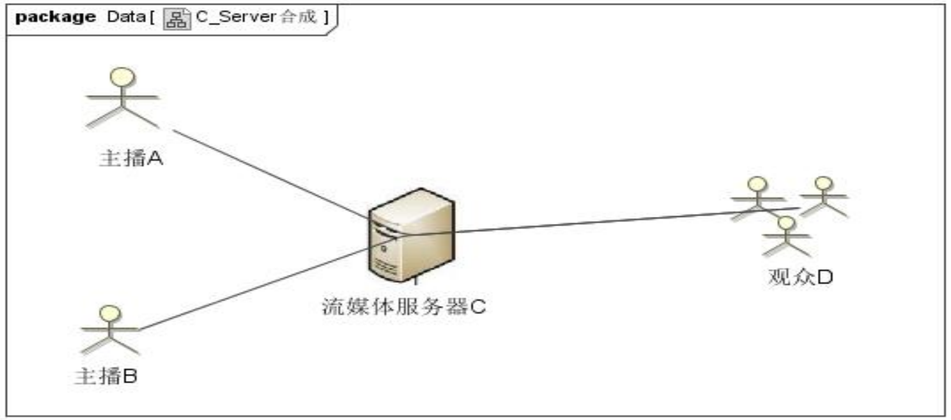
除了上面提到的异常用例外，我们跟测试一起总共构造了260+个类似的异常用例，我们的歌房都能够顺利通过验证。

精细分析——解决双人合唱人声/伴奏/歌词同步问题

歌房合唱——实时在线双人合唱方案对比

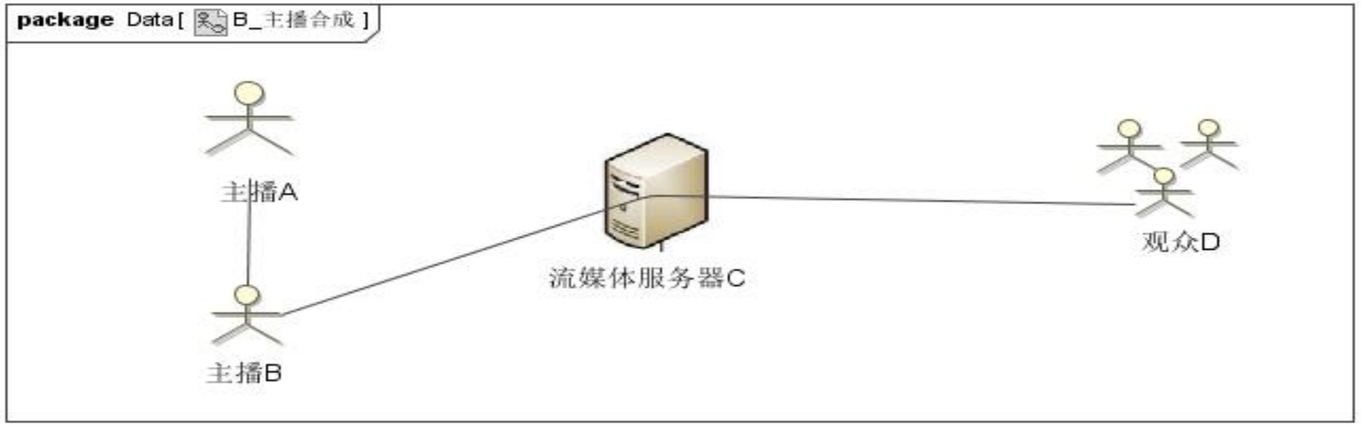


- *主播A和主播B都没有合唱体验*
- *人声伴奏对齐会非常困难*
- *带宽高，观众资源占用高*
- *h5/录制/旁路会支持不到。*



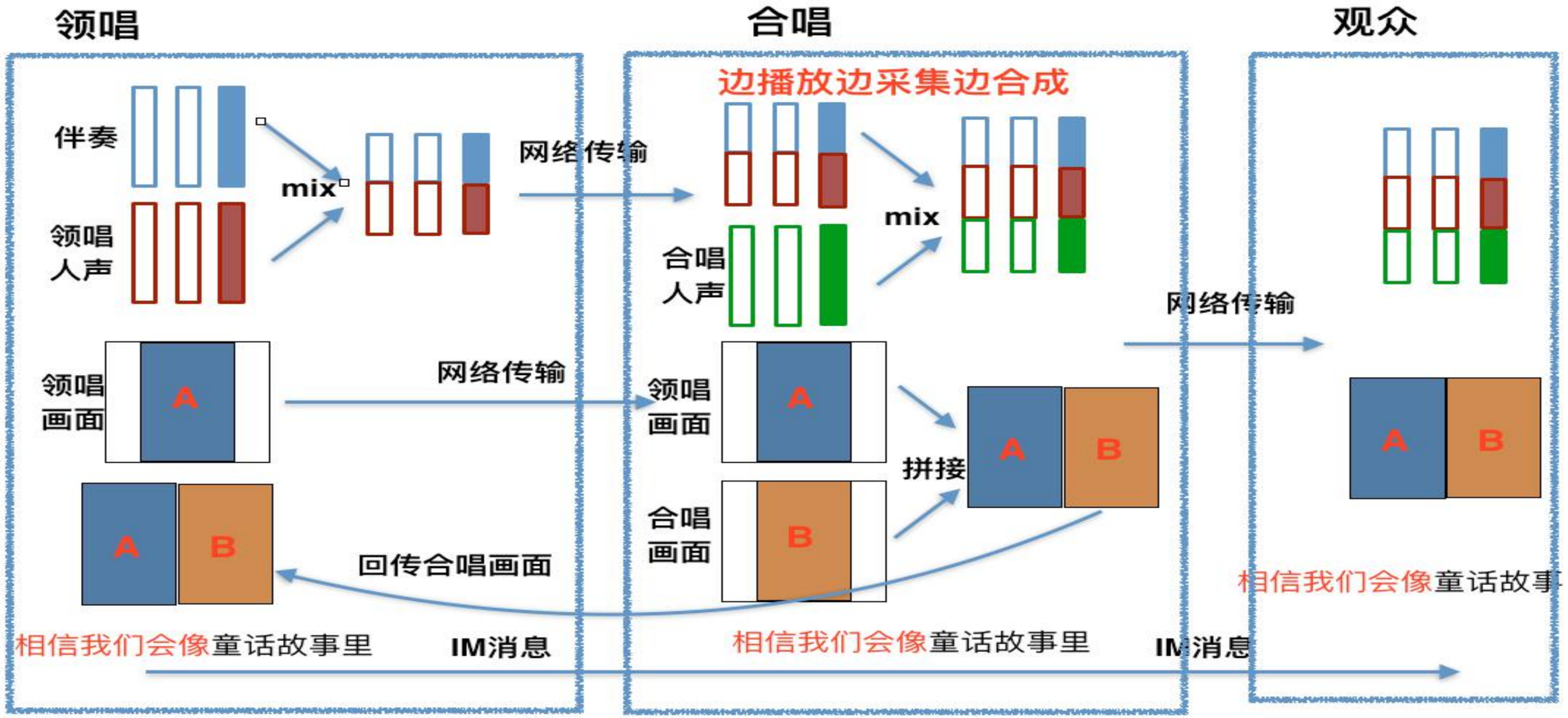
- *主播A和主播B都没有合唱体验*
- *服务器资源消耗大，延迟高，*
- *人声A/伴奏/人声B对齐非常困难，网络抖动、音视频的快进慢放让这里的复杂度陡增*

歌房合唱——实时在线双人合唱方案对比



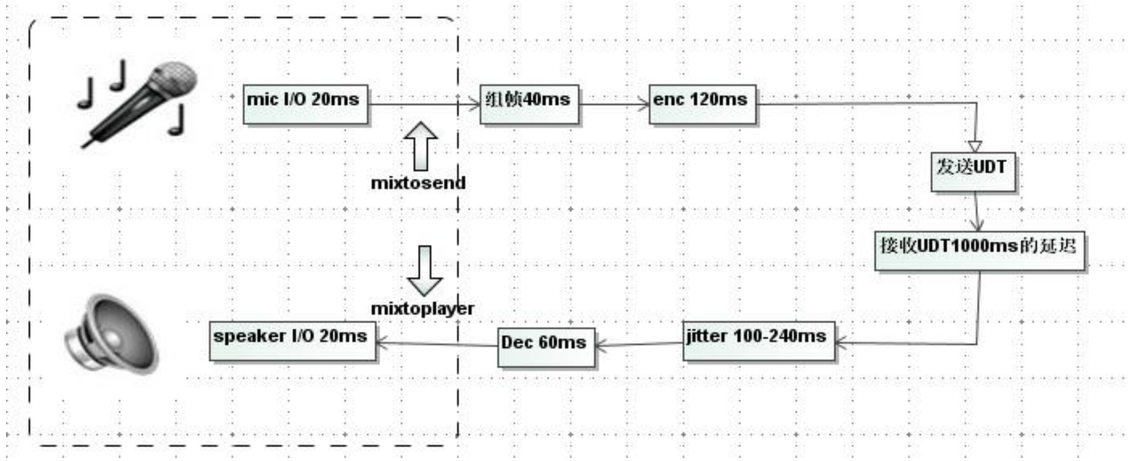
- *主播B都有合唱体验*
- *服务器跟正常直播消耗相当*
- *h5/录制/旁路可以很容易支持到。*
- *理论上可以严格对齐人声伴奏*

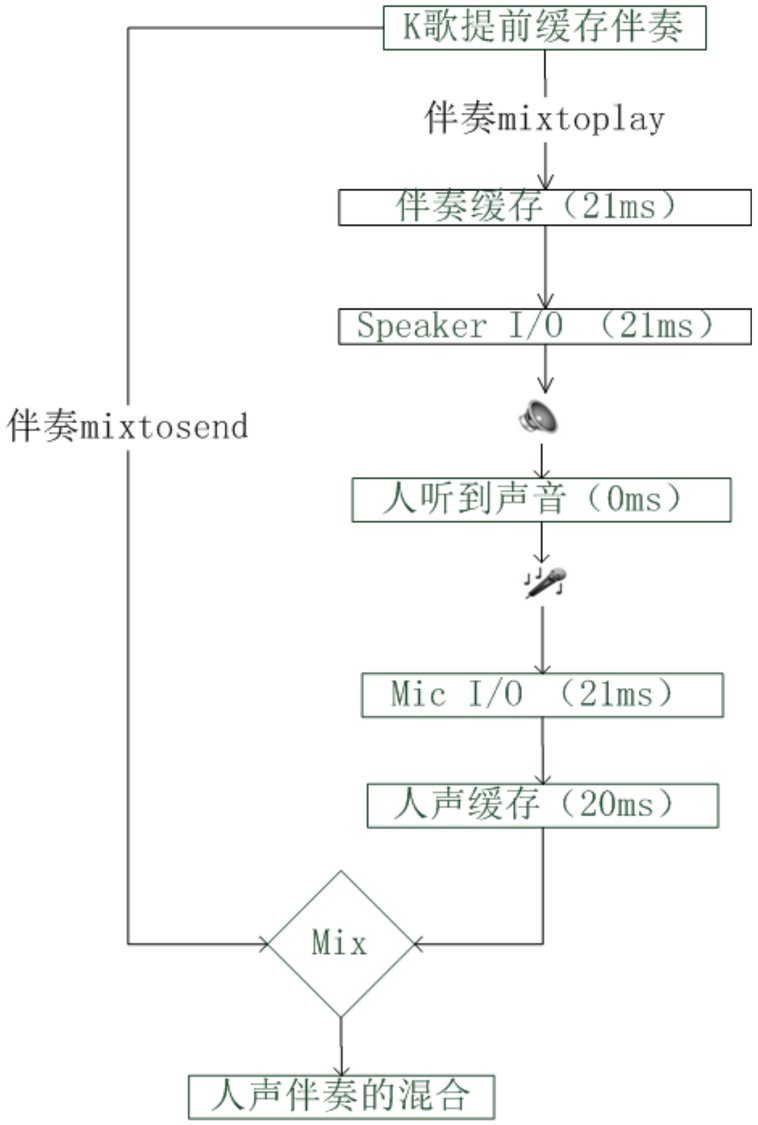
歌房合唱——实时在线双人音视频合唱方案



直播——人声/伴奏同步

人声伴奏的偏移80ms





需要人声伴奏同步的主要原因：录制和播放之间的时间差

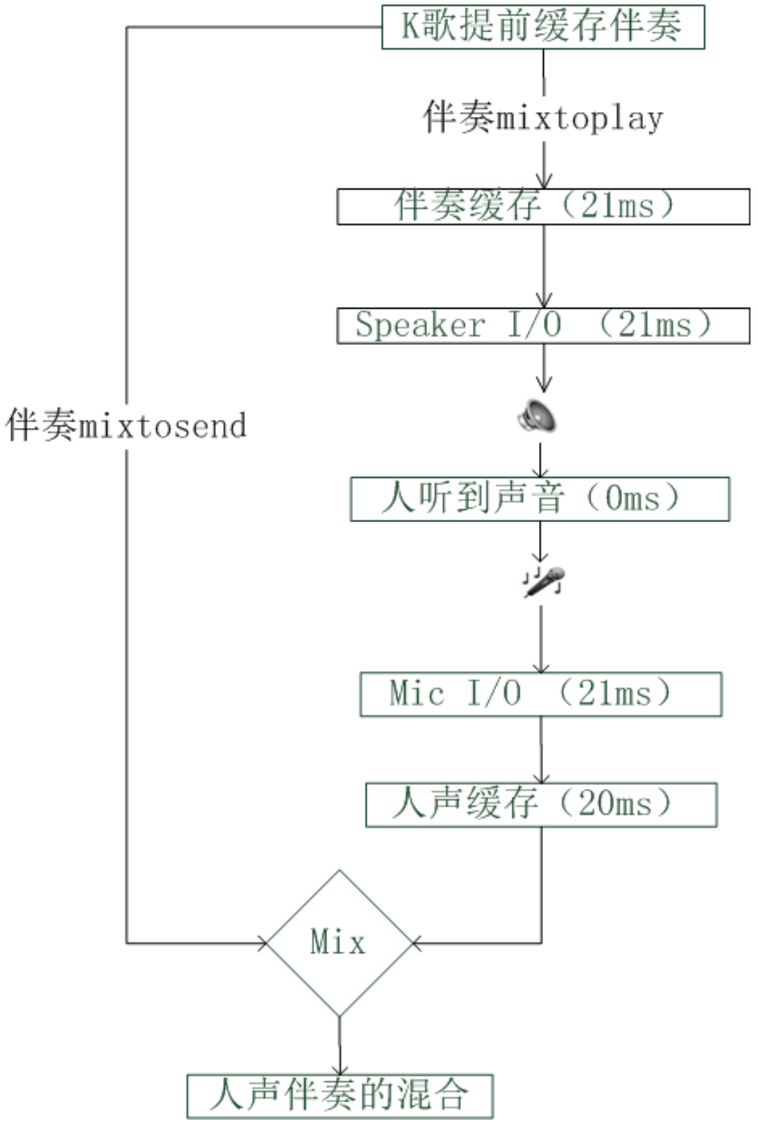
测试技巧：

1.左右声道分别播放人声和伴奏，保存mix后的波形对比

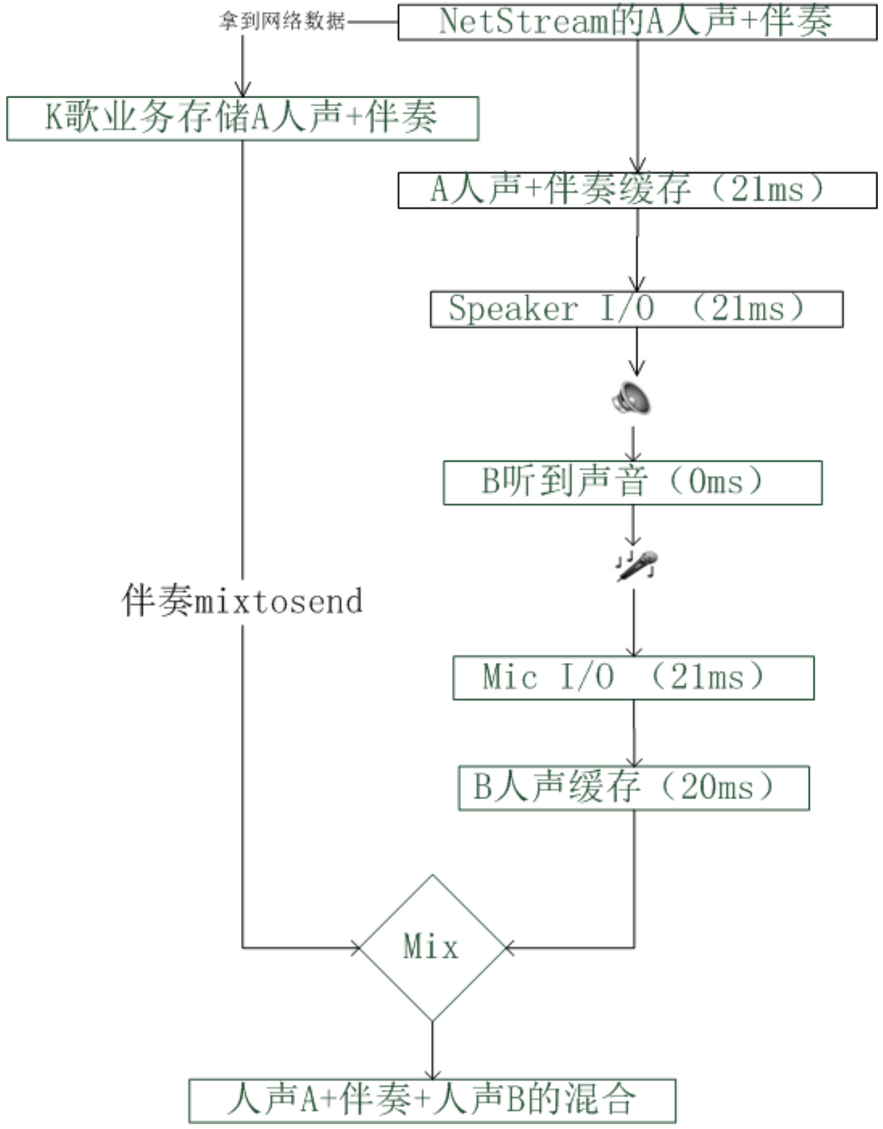
2.不做ec，播放一段伴奏从麦克风录入后对比波形

歌房合唱——人声A/伴奏/人声B的同步

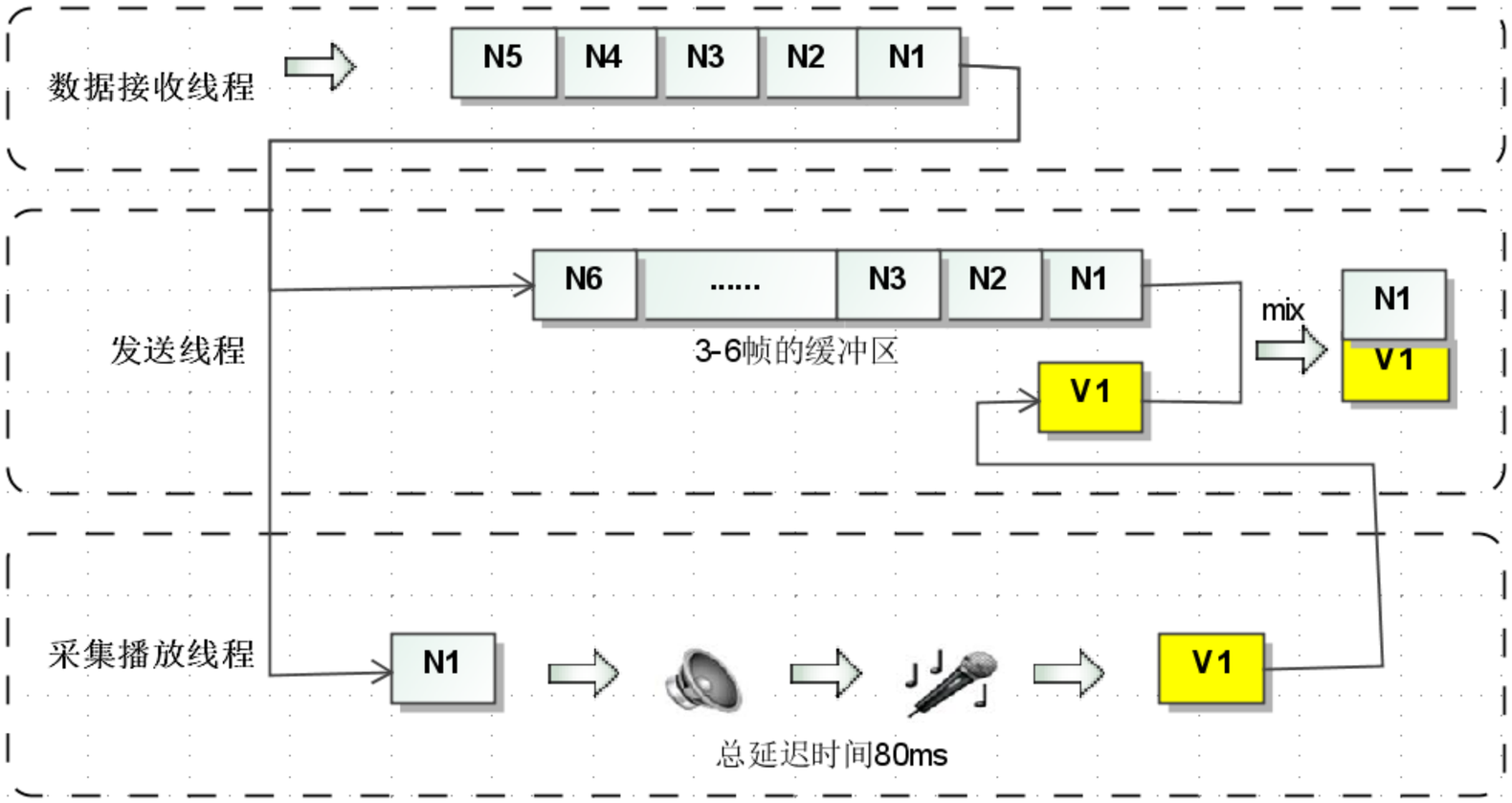
领唱



合唱

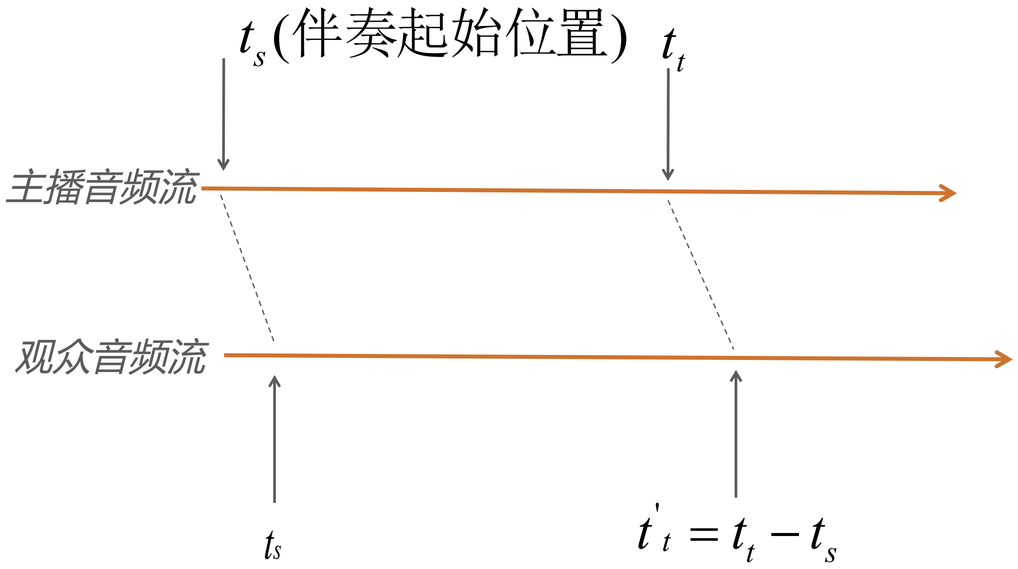


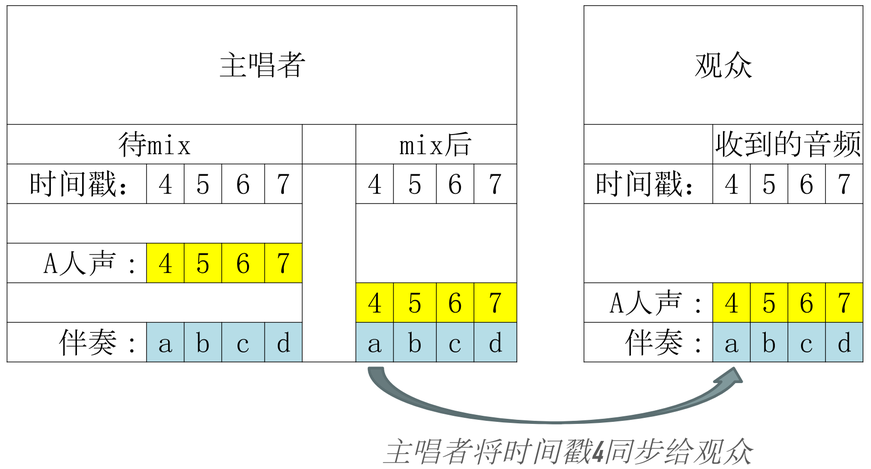
歌房合唱——人声A/伴奏/人声B的同步



合唱者人声A/伴奏/人声B的同步，延迟控制在20ms左右

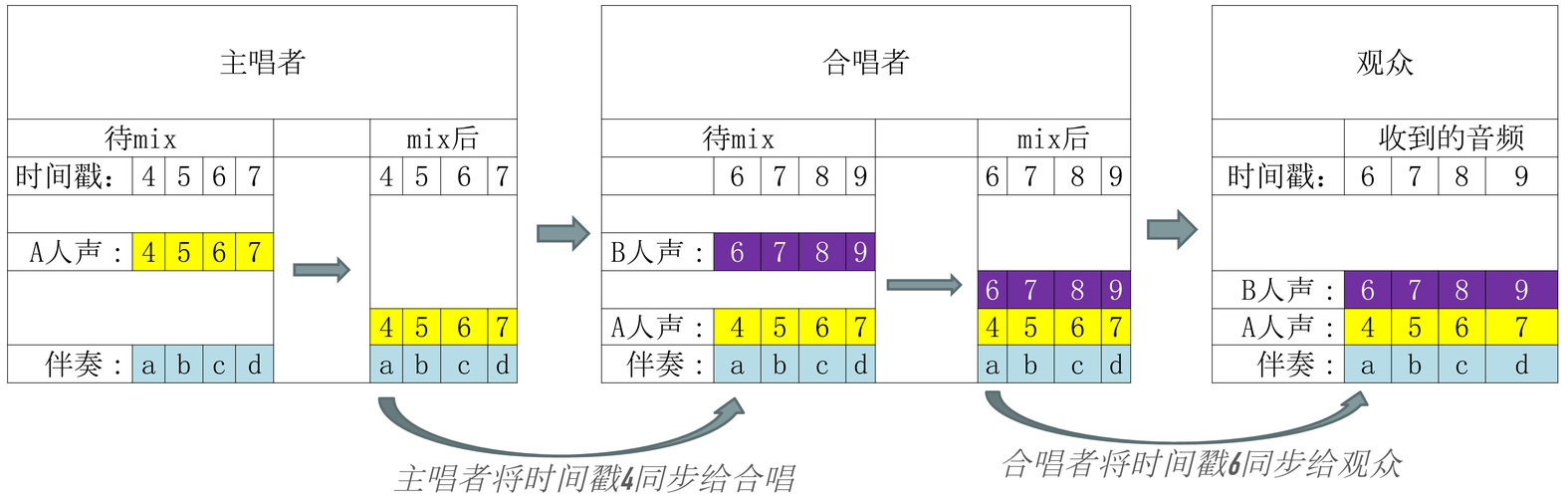
直播——歌词同步



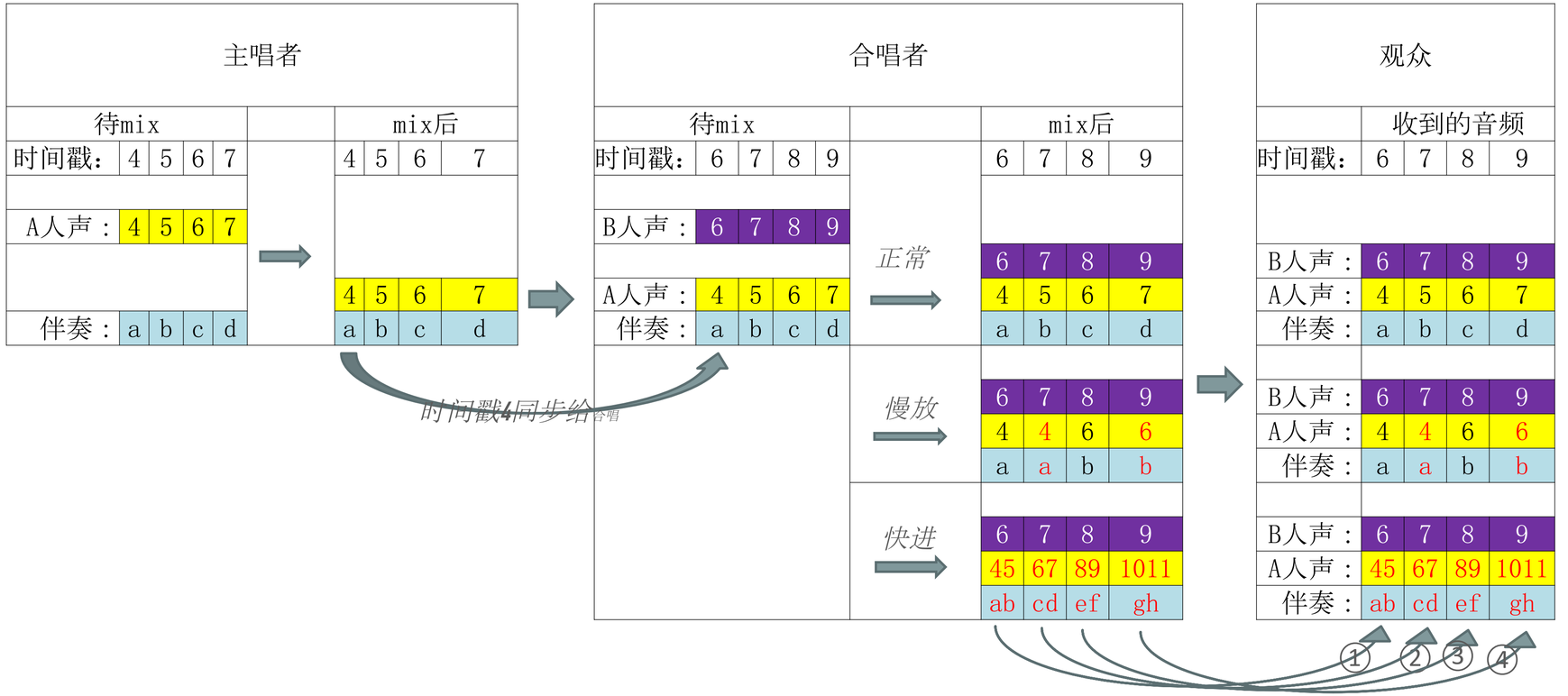


主播将 t_s 同步给观众，观众直接计算歌词位置 $t'_t = t_t - t_s$

歌房合唱——人声/伴奏/歌词同步



歌房合唱——人声/伴奏/歌词同步



①②③④分别将(6,a) (7,c) (8,e) (9,g)从合唱者同步给观众

问题	优化
1. B端快进慢放策略导致歌词偏差	1. 每20秒同步一次信令，定时修正误差 2. 将伴奏进度写入音频帧内，完美解决歌词同步问题（进行中）

云端调控——增强歌房对网络的抗性

云端调控——全方位云端控制

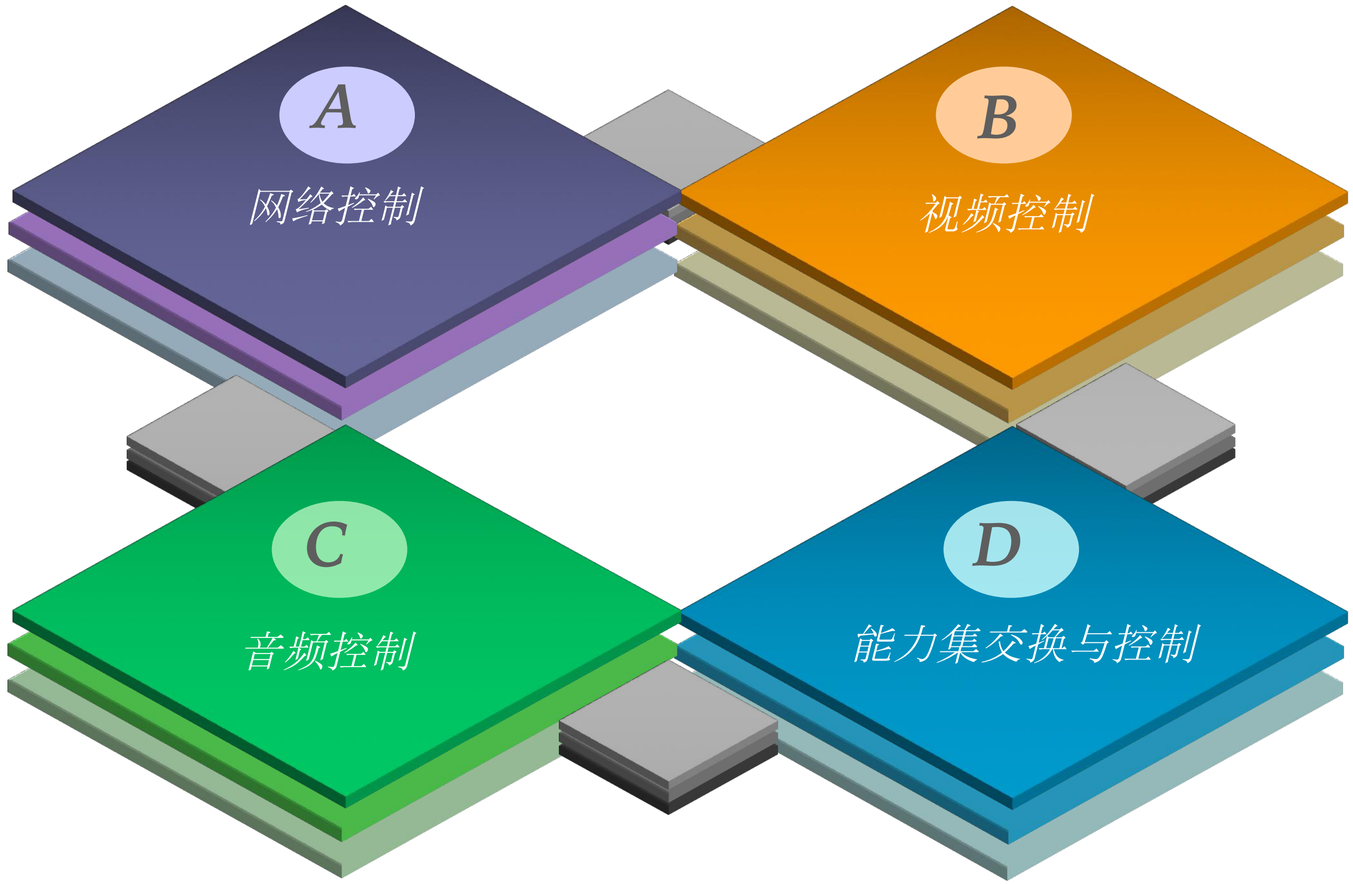
带宽估计

平滑发送

去抖，去重

FEC冗余控制

ARQ重传机制



编码器选择

编码序列选择

帧率、分辨率控制

3a选择

Fec控制

码率控制

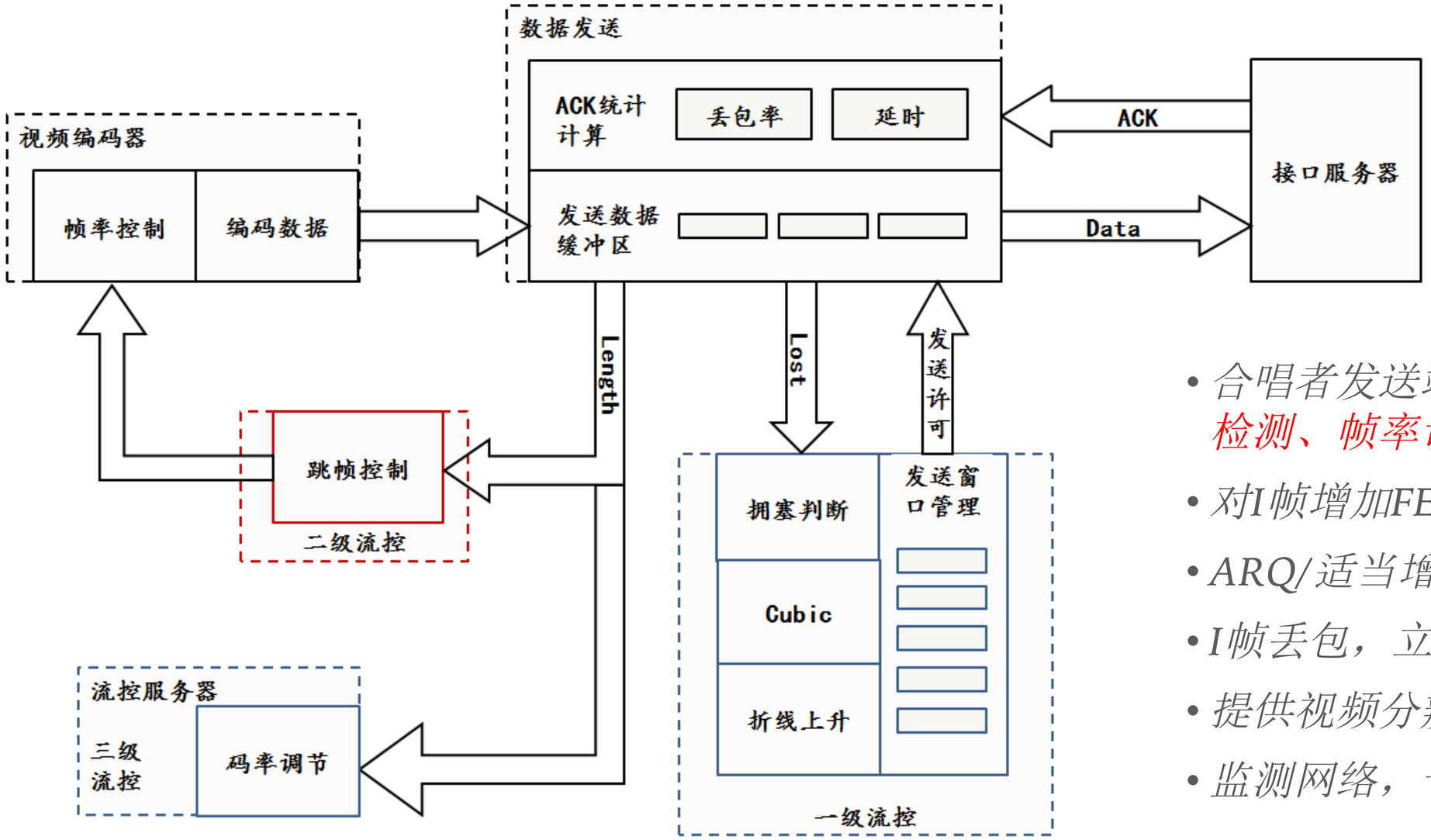
能力集交换

安卓机型适配

动态CPU适配

摄像头能力评估

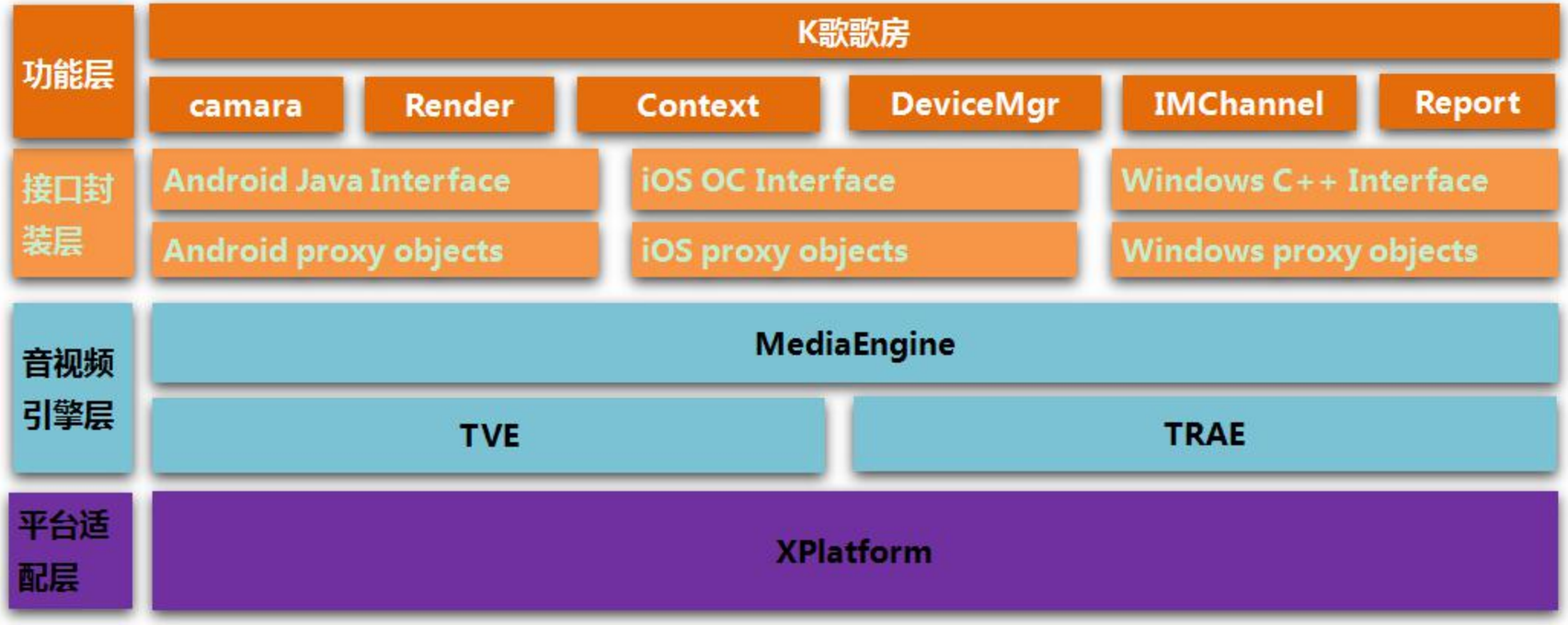
云端调控——合唱者网络抗性提升



- *合唱者发送端3级流控策略：拥塞检测、帧率调整、档位调节*
- *对I帧增加FEC*
- *ARQ/适当增加缓冲Buffer大小*
- *I帧丢包，立刻申请重传*
- *提供视频分辨率切换逻辑*
- *监测网络，卡顿提醒*

歌房整体框架改进

旧框架难支持的功能		调整框架后解决实现
不支持观众端/合唱端降低声音		播放在业务层，自己调节
人声伴奏不同步		业务来mix完成
不同的观众上行下行音视频会有内存泄漏		底层支持多路音视频的发送接收（发送和接收分别只一个通道）
无法评估出各个角色的质量状况		业务自己上报
不支持正方形的分辨率		新框架引擎只作为数据通道，业务支持
升级时间长，而且容易出问题		引擎很稳定，升级容易
业务形态不同导致上报复杂		业务根据需要上报



音视频引擎是流数据管道，负责音视频的编解码，流数据的发送，与业务完全解耦

END

感谢您的聆听.

-梅江霞（tigermei）

Q&A