



# 蘑菇街直播实践

蘑菇街-花荣

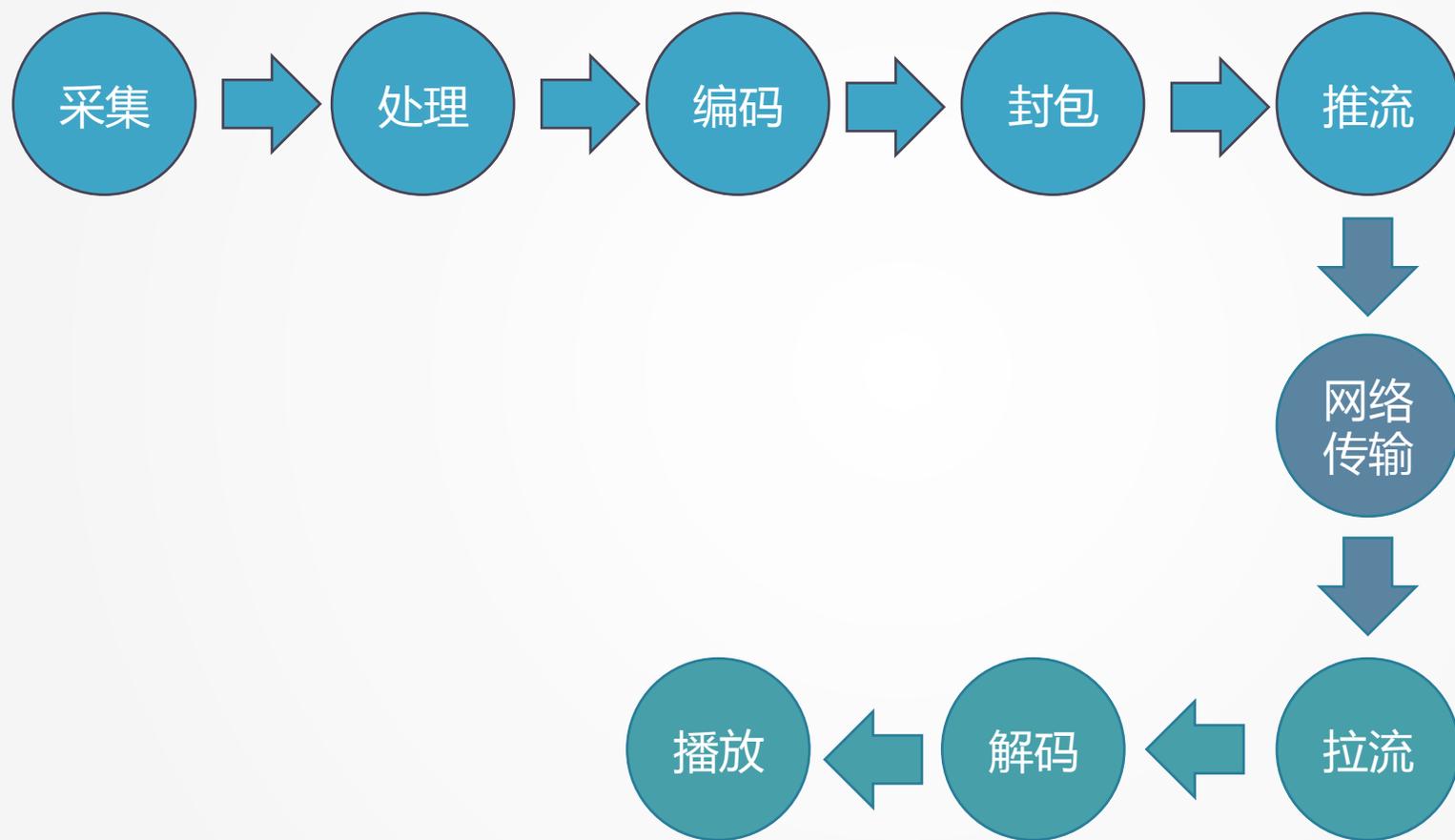
## 直播简介



## 什么是直播

通俗点讲就是将主播端的视频信息以较低的延时传输到观众端。直播和人们常见的点播在线视频的区别，在于低延时地将主播端的视频信息传输到观众端。

## 直播简介 – 主要环节



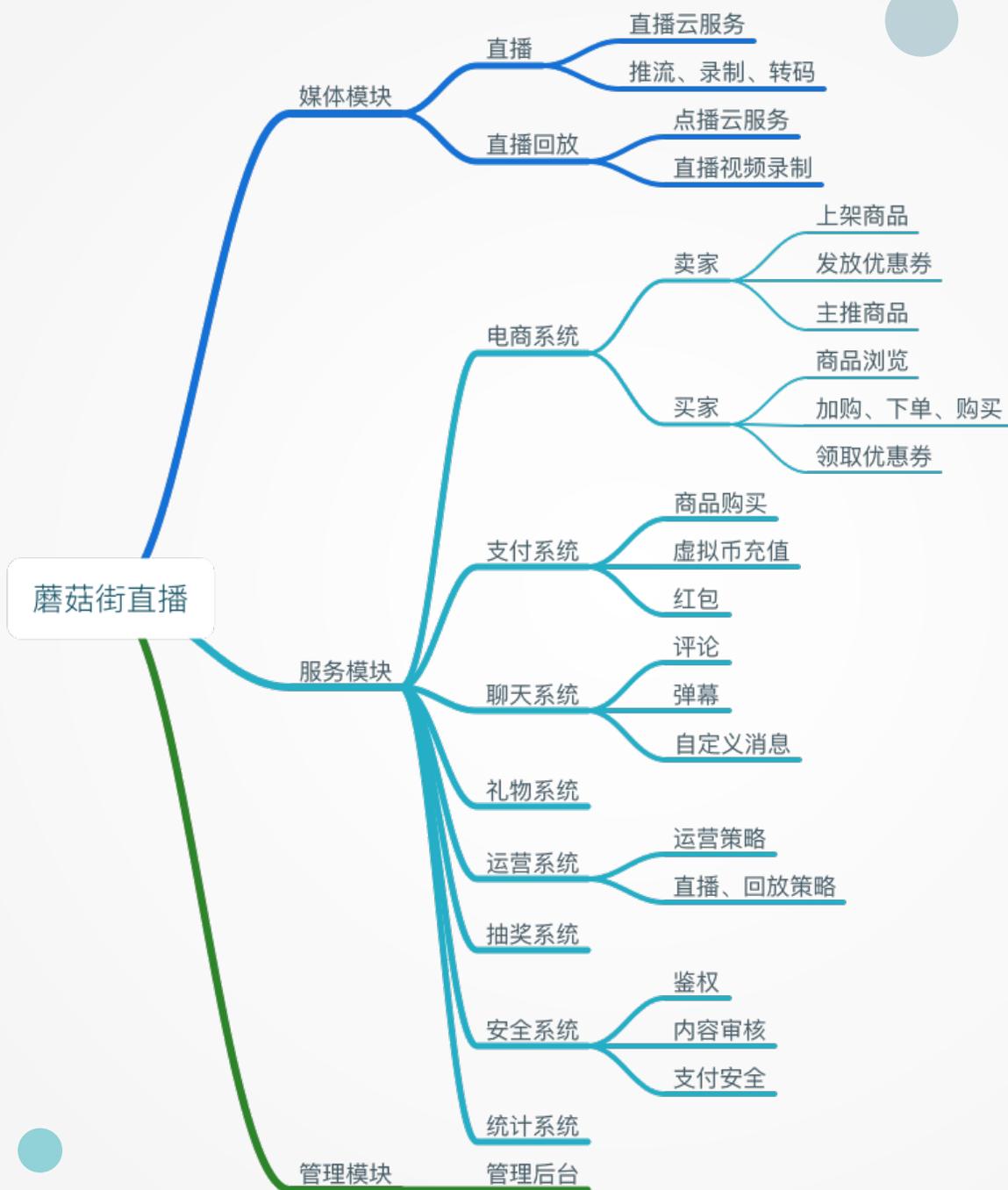
## 移动端直播形态

- 移动全民直播（映客、花椒）
- 社交软件直播（QQ空间直播，微博一直播、陌陌）
- 电商直播（蘑菇街、淘宝、聚美优品）
- 手游直播（斗鱼TV、触手TV）



# 蘑菇街直播形态

# 蘑菇街直播的组成

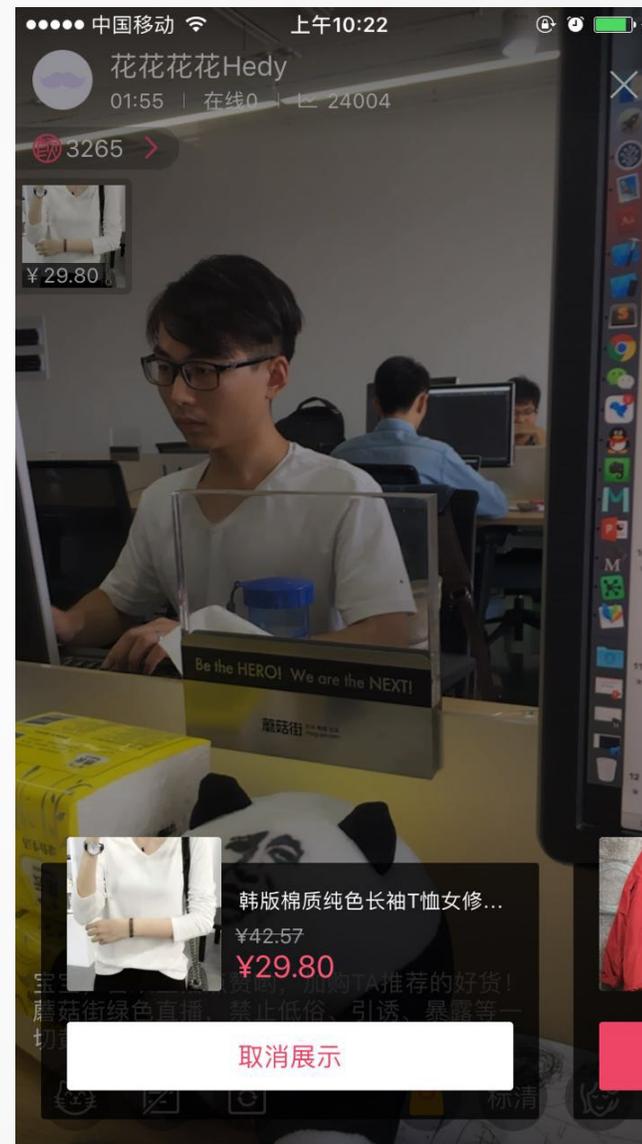


# 社交





# 电商 - 主播



# 电商 - 观众

中国移动 上午9:59

叫我吕太太 在线17056 [+关注](#) 蘑菇直播

570

¥49.90

宝宝，喜欢主播点赞哟，加购TA推荐的好货！  
蘑菇街绿色直播，禁止低俗、引诱、暴露等一切黄赌毒内容，警察叔叔24小时巡查哦~！

时光像个旧美人92 等一群小可爱来了

围观群众入场啦

大傻小傻一起吧 等一群小可爱来了

m\_jaymi7lyqdu57 身高160体重120可以穿吗

可爱点点Ella 等一群小可爱来了

加入购物车 立即购买

中国移动 上午9:55

许小丫 iamy 在线11869 [+关注](#) 蘑菇直播

3505

¥39.90

【小丫推荐】半永久撕拉...  
¥39.90

款式

1号深棕 2号自然棕

3号灰黑色

尺码

6g

数量

加入购物车 立即购买

宝宝，喜欢主播点赞哟，加购TA推荐的好货！  
蘑菇街绿色直播，禁止低俗、引诱、暴露等一切黄赌毒内容，警察叔叔24小时巡查哦~！

中国移动 上午9:49

odbo CC Brightening cc cushion

odbo

请大家 新品气垫带替换装

¥159.00

销量 67 默认快递

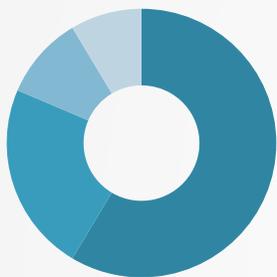
请选择 款式 尺码

私聊 收藏 小店 加入购物车 立即购买

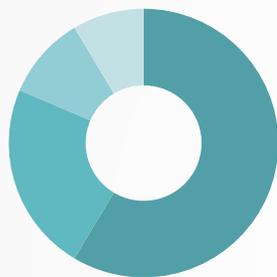


# 直播遇到的问题

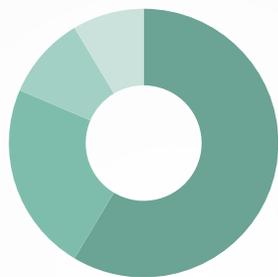
# 蘑菇街快速接入直播遇到的问题：



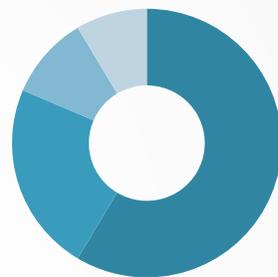
迭代



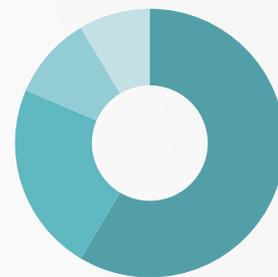
云服务



代码质量

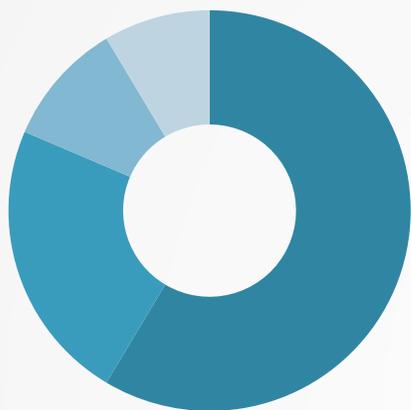


稳定性



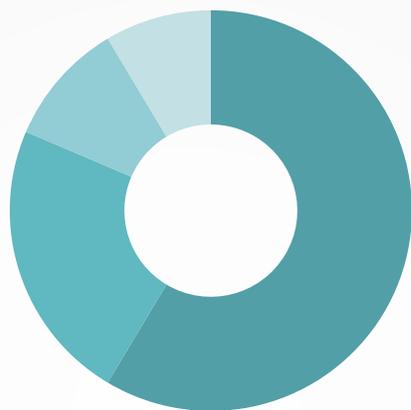
性能

# 迭代过快



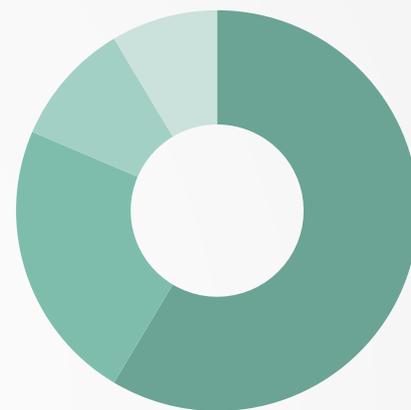
需求调研不充分

直播从立项到上线差不多三个星期



新需求

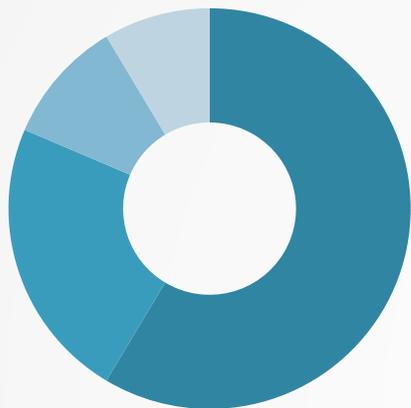
直播保持着两周一个版本上线的班车计划，实际开发周期一周



新老bug

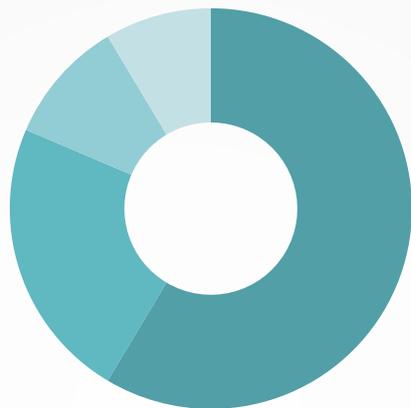
快速迭代上线的直播的之后几个版本，均是新老bug并存的局面

# 三方服务不稳定



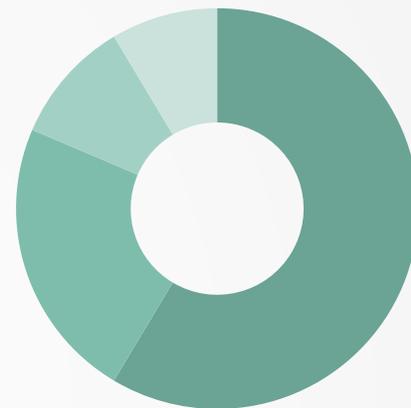
## 文档不全

接入文档不全或存在疑点，导致接入后因调用方式或者线程时序带来了不少问题



## 客户端SDK不稳定

SDK内部存在较多异常和crash问题



## 云服务不稳定

云服务不稳定，时常出现服务不可用情况



# 稳定性优化

稳定性存在的问题：

无法正常观看直播，闪退

- 内存泄漏
- 客户端SDK不稳定
- 存在硬件兼容性问题
- 多线程情况复杂

# 稳定性优化

Code Review & 代码规范  
优化代码质量  
统一编码风格

静态分析 & 内存泄漏检测

接入静态分析  
Debug模式下接入  
MLeaksFinder

日志查看上报

CocoaLumberJack主流程打点  
日志查看、发送  
日志定向上报分析



## 稳定性优化 – 案例：block嵌套优化方案

修改为**单一职责**的方法调用，新增了代码的可阅读性和可维护性，同时不易造成回调丢失，内存泄漏等问题

```
/**
 * 登录IM
 */
- (void)loginIMWithSignature:(NSString *)signature
{
    /**
     * 建房流程2
     */
    self.createRoomStep = 2;
    LiveSDKLogInfo(@"建房流程2 loginIM");

    MGJLVWeakSelf;
    [MGJLVLoginManager login:self.userId userSign:signature succ:^(id data) {
        MGJLVStrongSelf
        [self startVideoContext];
    } fail:^(NSError *error) {
        MGJLVStrongSelf;
        NSError *err = createMGJLVRoomErrorMessageFrom(MGJLVRoom_Error_Create_Room_Fail, error);
        [self onCreateRoomFailedHandler:err];
        LiveSDKLogError(@"建房流程2 error:%@", err);
    }];
}

/**
 * 开启视频上下文
 */
- (void)startVideoContext
{
    /**
     * 建房流程3
     */
    self.createRoomStep = 3;
    LiveSDKLogInfo(@"建房流程3 startVideoContext");

    MGJLVWeakSelf
    [[MGJLVVideoContextManager sharedInstance] startVideoContext:self.userId complete:^(QAVResult result) {
        MGJLVStrongSelf
        if (result != QAV_OK) {
            NSError *error = createMGJLVVideoError(MGJLVVideo_Error_Start_Context_Fail, result, @"开启视频上下文失败");
            [self onCreateRoomFailedHandler:error];
            LiveSDKLogError(@"建房流程3 error:%@", error);
        }
        else {
            [self onLoginSuccessHandler];
        }
    }];
}

- (void)onLoginSuccessHandler
{
    LiveSDKLogInfo(@"建房流程onLoginSuccessHandler");

    [self createVideoRoom];
    [self createChatRoom];
}
}
```

## 稳定性优化 – 内存泄漏

- Instruments进行内存泄漏的检测
- iOS客户端使用MLeaksFinder针对直播组件进行了Debug下的内存泄漏检测，将内存泄漏扼杀在开发阶段。



# 稳定性优化 – 内存泄漏MLeaksFinder

```
#pragma mark - MLeaksFinder
#ifdef ENABLE_LEAK_FINDER
- (BOOL)willDealloc {
    if (![super willDealloc]) {
        return NO;
    }

    // components
    MLCheck(self.viewerRoomInfo);
    MLCheck(self.closeComponent);
    MLCheck(self.containerComponent);
    MLCheck(self.hostInfoComponent);
    MLCheck(self.toolbarComponent);
    MLCheck(self.praiseComponent);
    MLCheck(self.giftPlayerComponent);
    MLCheck(self.debugComponent);
    MLCheck(self.giftPickerComponent);
    MLCheck(self.barrageComponent);
    MLCheck(self.inputBarComponent);
    MLCheck(self.biglegUserComponent);
    MLCheck(self.shareComponent);
    MLCheck(self.leftContainerComponent);
    MLCheck(self.goodsComponent);
    MLCheck(self.commentTableComponent);
    MLCheck(self.luckBagComponent);
    MLCheck(self.goodsCouponRushComponent);
    MLCheck(self.hostBusyComponent);
    MLCheck(self.networkAlertComponent);

    // services
    MLCheck(self.paymentService);

    return YES;
}
#endif
```

## Mute Assertion

If your class is designed as singleton or for some reason objects of your class should not be deallocated, override `-(BOOL)willDealloc` in your class by returning NO.

```
- (BOOL)willDealloc {
    return NO;
}
```

## Find Leaks in Other Objects

MLeaksFinder finds leaks in UIView and UIViewController objects by default. However, you can extend it to find leaks in the whole object graph rooted at a UIViewController object.

```
- (BOOL)willDealloc {
    if (![super willDealloc]) {
        return NO;
    }

    MLCheck(self.viewModel);
    return YES;
}
```

## 稳定性优化 – 流程打点

日志内部使用CocoaLumberJack实现，自定义了log等级，log文件大小，回滚频率，数量分别为2MB，24h，10个。

```
typedef NS_OPTIONS(NSUInteger, MGJLiveLogFlag) {
    MGJLiveLogFlagError      = (1 << 5), // 0...0000100000
    MGJLiveLogFlagWarning    = (1 << 6), // 0...0001000000
    MGJLiveLogFlagInfo       = (1 << 7), // 0...0010000000
    MGJLiveLogFlagDebug      = (1 << 8), // 0...0100000000
    MGJLiveLogFlagVerbose    = (1 << 9), // 0...1000000000
};

typedef NS_ENUM(NSUInteger, MGJLiveLogLevel) {
    MGJLiveLogLevelOff      = 0,
    MGJLiveLogLevelError    = (MGJLiveLogFlagError),
    MGJLiveLogLevelWarning  = (MGJLiveLogLevelError | MGJLiveLogFlagWarning),
    MGJLiveLogLevelInfo     = (MGJLiveLogLevelWarning | MGJLiveLogFlagInfo),
    MGJLiveLogLevelDebug    = (MGJLiveLogLevelInfo | MGJLiveLogFlagDebug),
    MGJLiveLogLevelVerbose  = (MGJLiveLogLevelDebug | MGJLiveLogFlagVerbose),
    MGJLiveLogLevelAll      = MGJLiveLogLevelVerbose //只是LiveLog中的所有Level
};

typedef NS_ENUM(NSUInteger, MGJLiveLogContext) {
    MGJLiveLogContextDefault = 10,
    MGJLiveLogContextCoreSDK = 11,
    MGJLiveLogContextLiveSDK = 12,
    MGJLiveLogContextMWP     = 13,
};
```

## 稳定性优化 – 流程打点

使用相应的log等级，在需要的地方打印log即可

```
/**
 * 登录IM
 */
- (void)loginIMwithSignature:(NSString *)signature
{
    /**
     * 建房流程2
     */
    self.createRoomStep = 2;
    LiveSDKLogInfo(@"建房流程2 loginIM");

    MGJLVWeakSelf;
    [MGJLVLoginManager login:self.userId userSign:signature succ:^(id data) {
        MGJLVStrongSelf
        [self startVideoContext];
    } fail:^(NSError *error) {
        MGJLVStrongSelf;
        NSError *err = createMGJLVRoomErrorMessageFrom(MGJLVRoom_Error_Create_Room_Fail, error);
        [self onCreateRoomFailedHandler:err];
        LiveSDKLogError(@"建房流程2 error:%@", err);
    }];
}
```

## 稳定性优化 – 流程打点

### 解决发生异常或者crash 无法定位问题的痛点

```
●●● 中国移动 下午8:33
<Warning><MWP><com.apple.main-thread> 2016-09-17
20:25:22.415 MWPRemote+MGJLive:94 __104-
[MWPRemote(MGJLive)
mgjliveRequestWithMethod:api:version:params:completion:needSec
urity.bizConfigName:]_block_invoke
-> [GET][mwp.mogulive.livesService:2] -> error: Error
Domain=NSURLErrorDomain Code=-1001 "请求超时。"
UserInfo={NSUnderlyingError=0x1503b42a0 {Error
Domain=kCFErrorDomainCFNetwork Code=-1001 "请求超时。"
UserInfo={NSErrorFailingURLStringKey=http://
newapi.mogujie.com/gw/mwp.mogulive.livesService/2?
rnd=IT760SLE, NSErrorFailingURLKey=http://
newapi.mogujie.com/gw/mwp.mogulive.livesService/2?
rnd=IT760SLE, _kCFStreamErrorCodeKey=-2102,
_kCFStreamErrorDomainKey=4, NSLocalizedDescription=请求超
时。}}, NSErrorFailingURLStringKey=http://newapi.mogujie.com/
gw/mwp.mogulive.livesService/2?rnd=IT760SLE,
NSErrorFailingURLKey=http://newapi.mogujie.com/gw/
mwp.mogulive.livesService/2?rnd=IT760SLE,
_kCFStreamErrorDomainKey=4,
_kCFStreamErrorCodeKey=-2102, NSLocalizedDescription=请求
超时。}
<Warning><MWP><com.apple.main-thread> 2016-09-17
20:25:22.441 MWPRemote+MGJLive:94 __104-
[MWPRemote(MGJLive)
mgjliveRequestWithMethod:api:version:params:completion:needSec
urity.bizConfigName:]_block_invoke
-> [GET][mwp.mogulive.liveListService:4] -> error: Error
Domain=NSURLErrorDomain Code=-1001 "请求超时。"
UserInfo={NSUnderlyingError=0x1501b57f0 {Error
Domain=kCFErrorDomainCFNetwork Code=-1001 "请求超时。"
UserInfo={NSErrorFailingURLStringKey=http://
newapi.mogujie.com/gw/mwp.mogulive.liveListService/4?
rnd=IT760VQI, NSErrorFailingURLKey=http://
newapi.mogujie.com/gw/mwp.mogulive.liveListService/4?
rnd=IT760VQI, _kCFStreamErrorCodeKey=-2102,
_kCFStreamErrorDomainKey=4, NSLocalizedDescription=请求超
时。}}, NSErrorFailingURLStringKey=http://newapi.mogujie.com/
gw/mwp.mogulive.liveListService/4?rnd=IT760VQI,
NSErrorFailingURLKey=http://newapi.mogujie.com/gw/
mwp.mogulive.liveListService/4?rnd=IT760VQI,
_kCFStreamErrorDomainKey=4,
_kCFStreamErrorCodeKey=-2102, NSLocalizedDescription=请求
```





# 性能优化

## 问题五：性能

初期不支持硬件编解码

不支持硬件编解码导致CPU和内存占用过高，容易发烫、卡顿

评论列表刷新过于频繁

评论列表在高并发情况下依然保持收到消息就立即处理刷新，导致很多资源的损耗

点赞、礼物、弹幕渲染

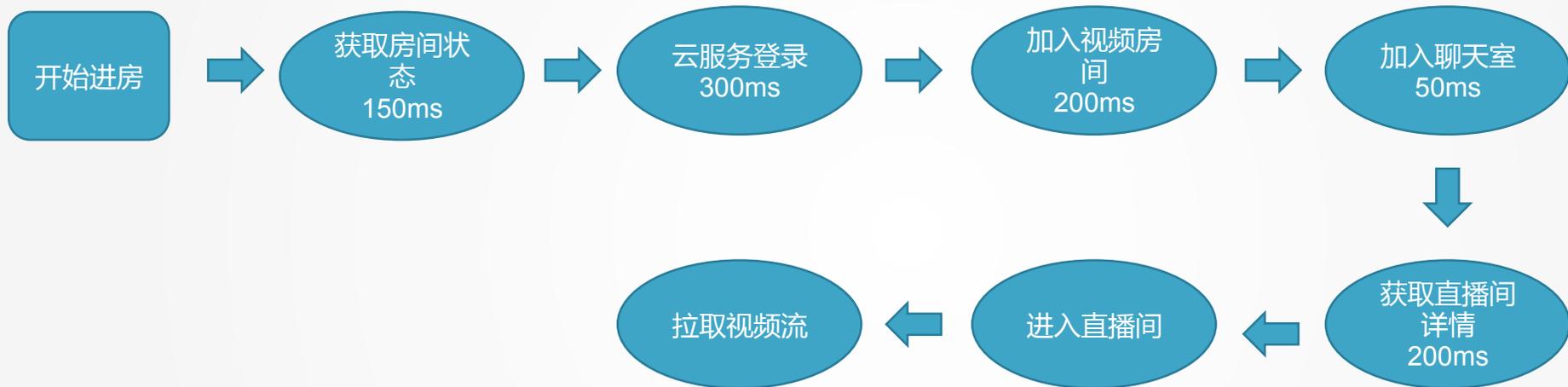
点赞、礼物、弹幕在高并发下存在无法复用、离屏渲染、图片缓存等问题

高并发下打点

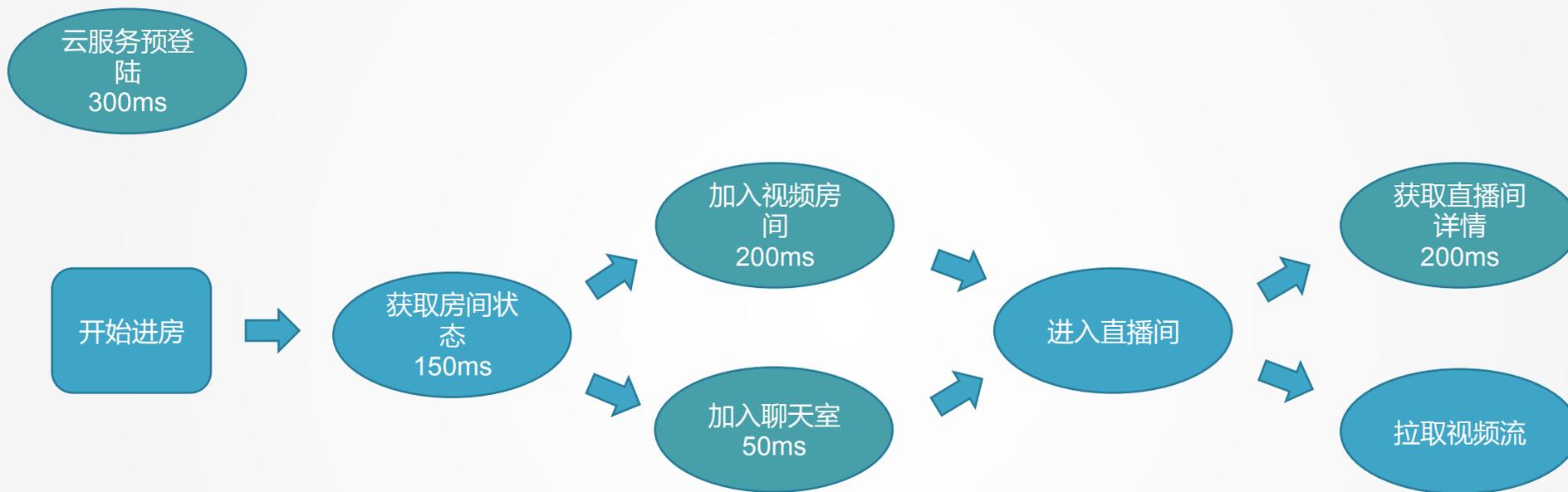
高并发下的基础库打点存在磁盘读写和网络请求频繁的问题，CPU负荷重

## 性能优化：进房速度慢

### 串行的进房流程



# 性能优化：进房速度慢优化方案

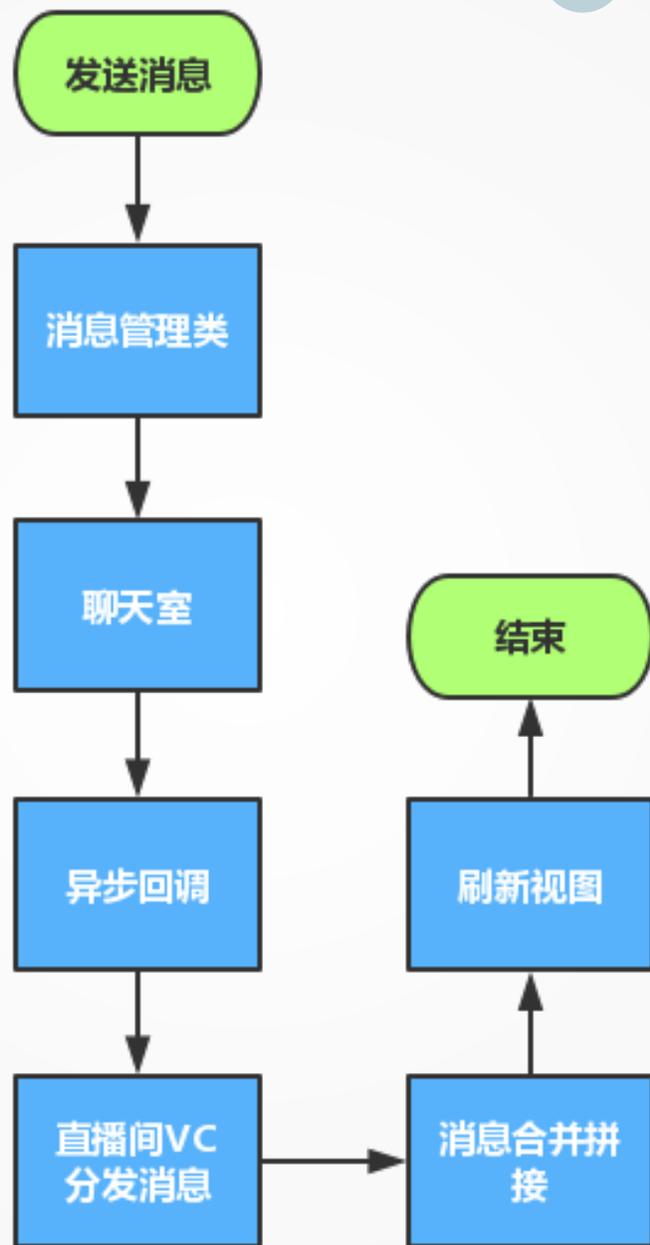


## 性能优化：进房速度慢优化结果

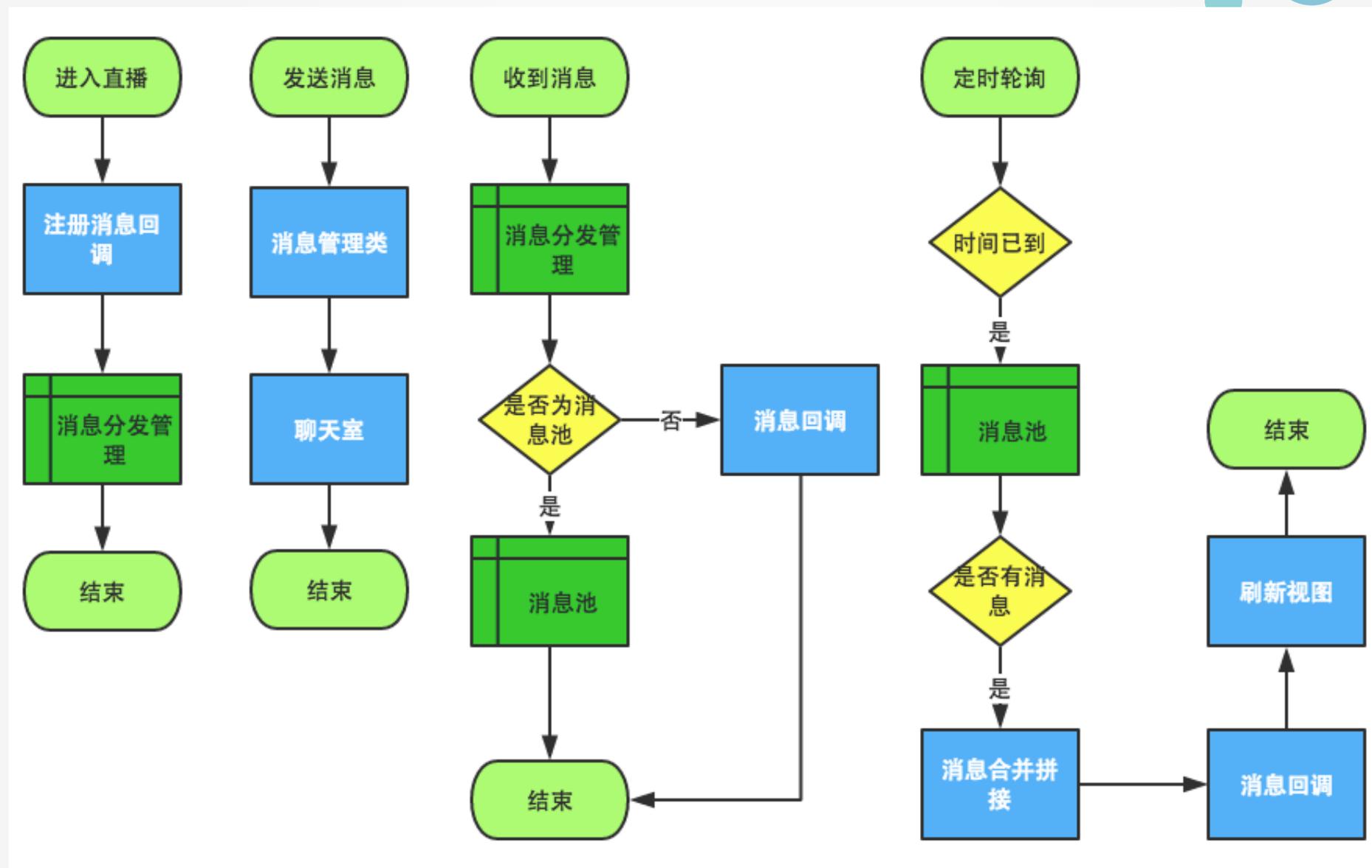


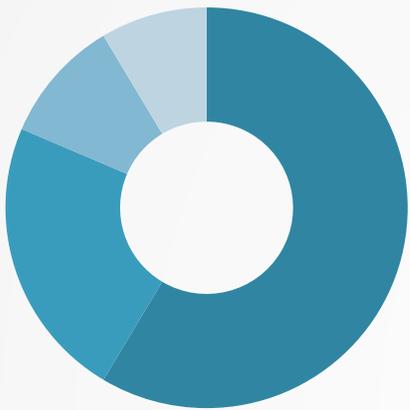
## 存在的三个问题

- 触发时机过于频繁
- 消息缺少缓存池
- 没有缓存列表的高度计算



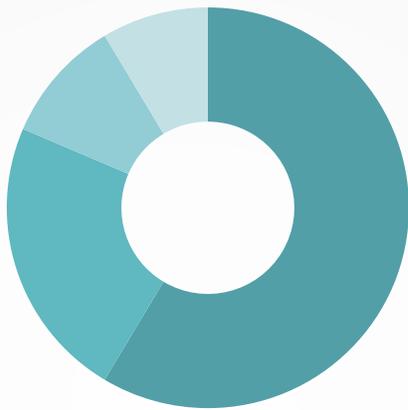
# 性能优化 – 消息优化方案





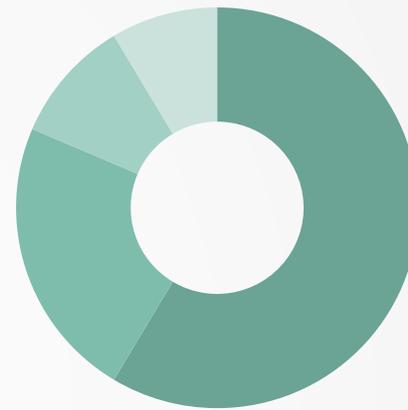
### 动画控件复用

优化方案：复用动画控件，减少内存的分配以及防止内存爆发式增长



### 离屏渲染严重

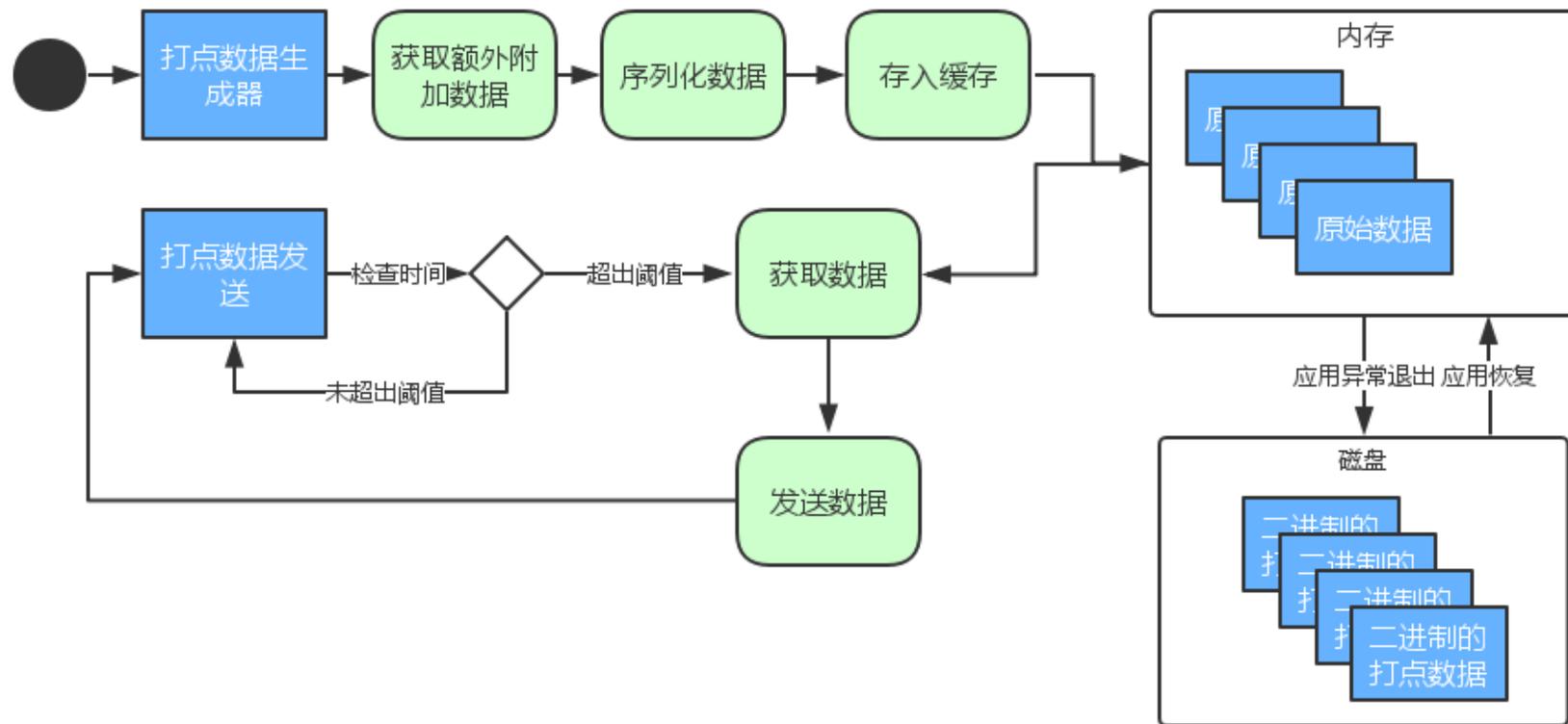
优化方案：使用instruments检测离屏渲染严重的视图，例如去除圆角，栅格化等损耗CPU性能的方式



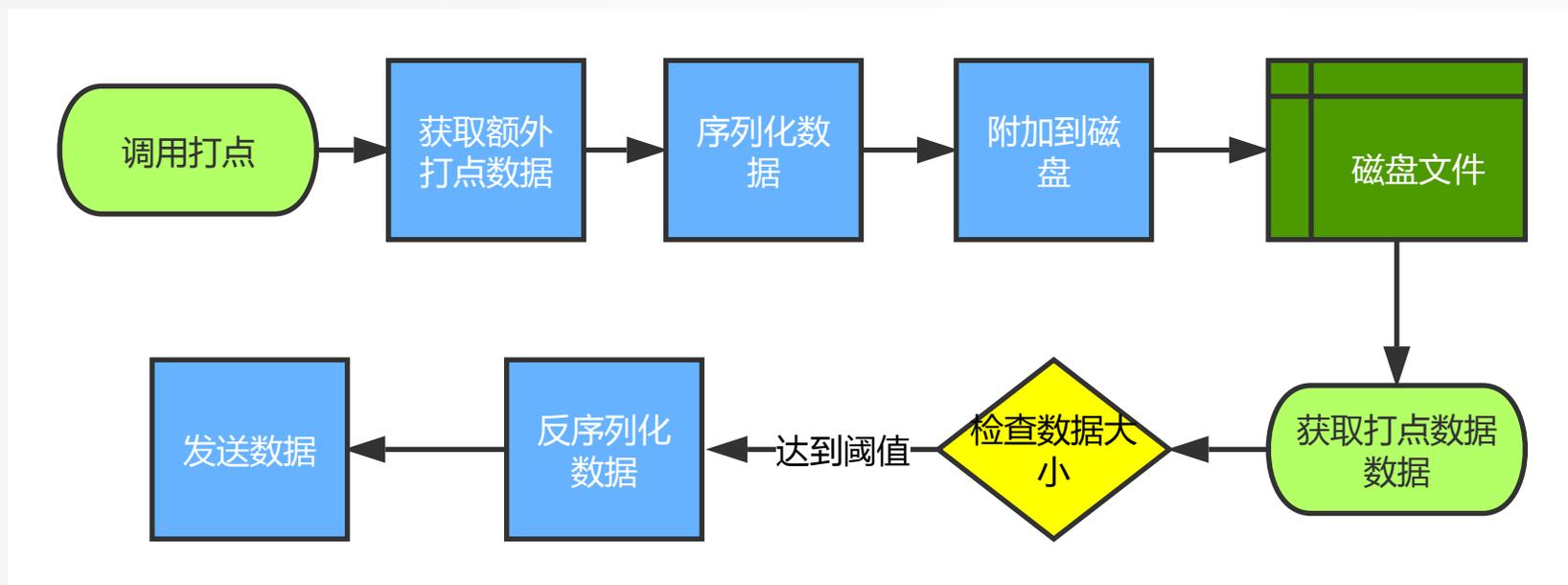
### 序列帧图片缓存

解决方案：使用YYImage三方库处理了直播内部的大量序列帧动画，及时释放动画的图片缓存

# 性能优化 – 打点



## 性能优化 – 打点



- 每次打点都要读写磁盘，IO和CPU负荷重
- 打点数据超过1K就要发送，短时间频繁打点导致网络频繁发送请求

## 性能优化 – 打点

移除每次都要进行的磁盘读写操作

将打点数据放入内存中作为缓存

使用一个队列从缓存中读取打点数据发送，避免频繁请求网络的情况



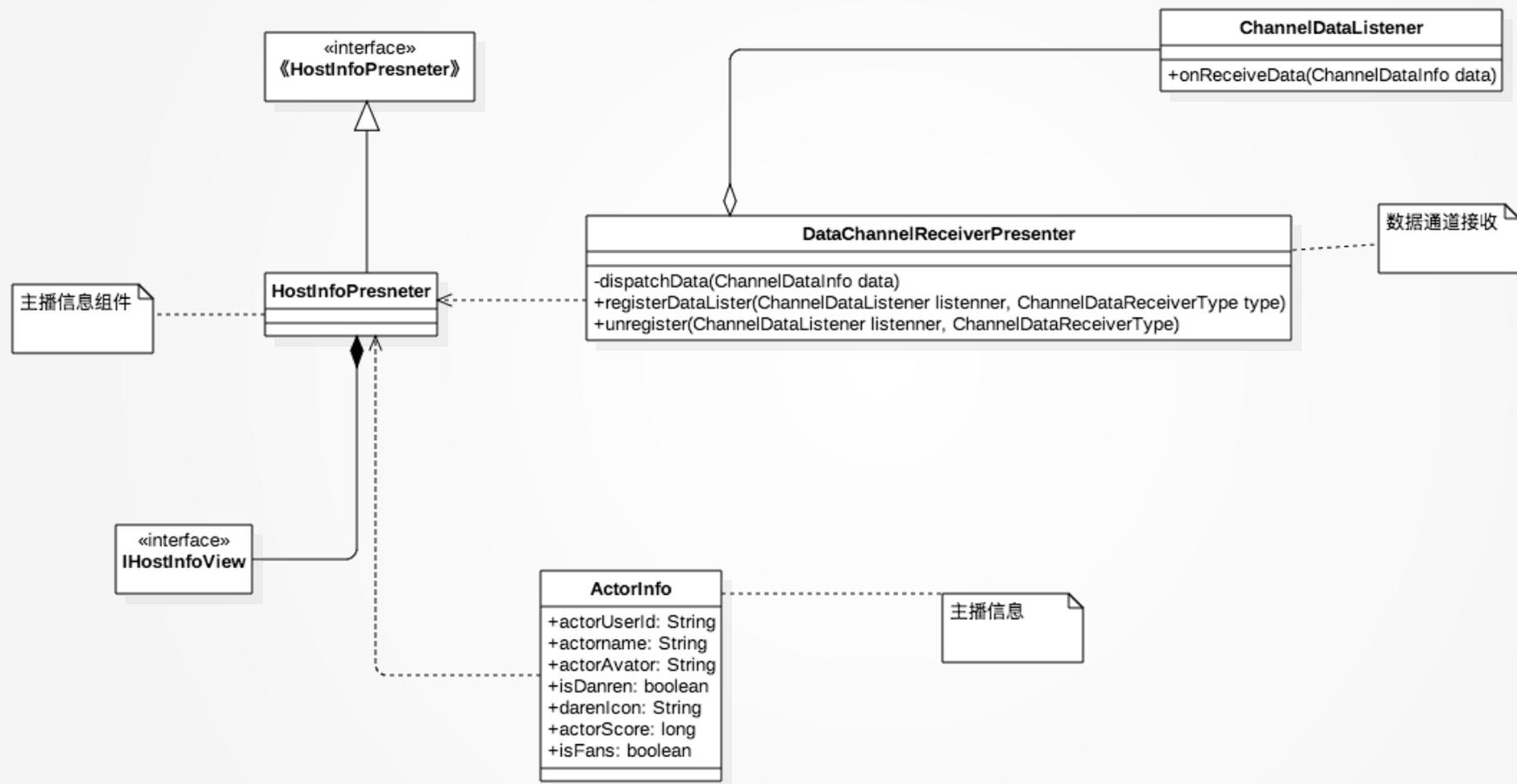


# 组件化

# 直播组件化



# 直播组件化 – 案例直播信息

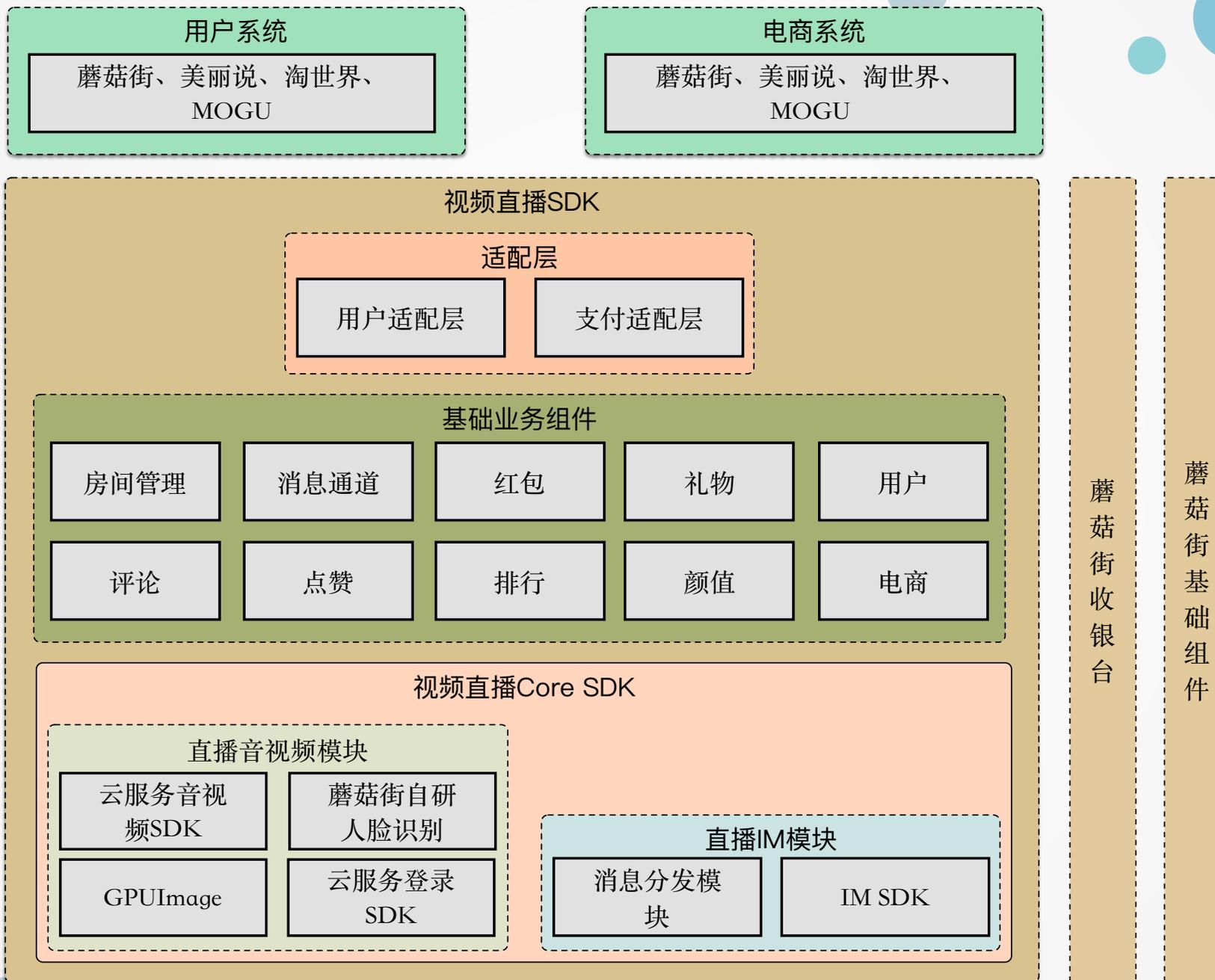




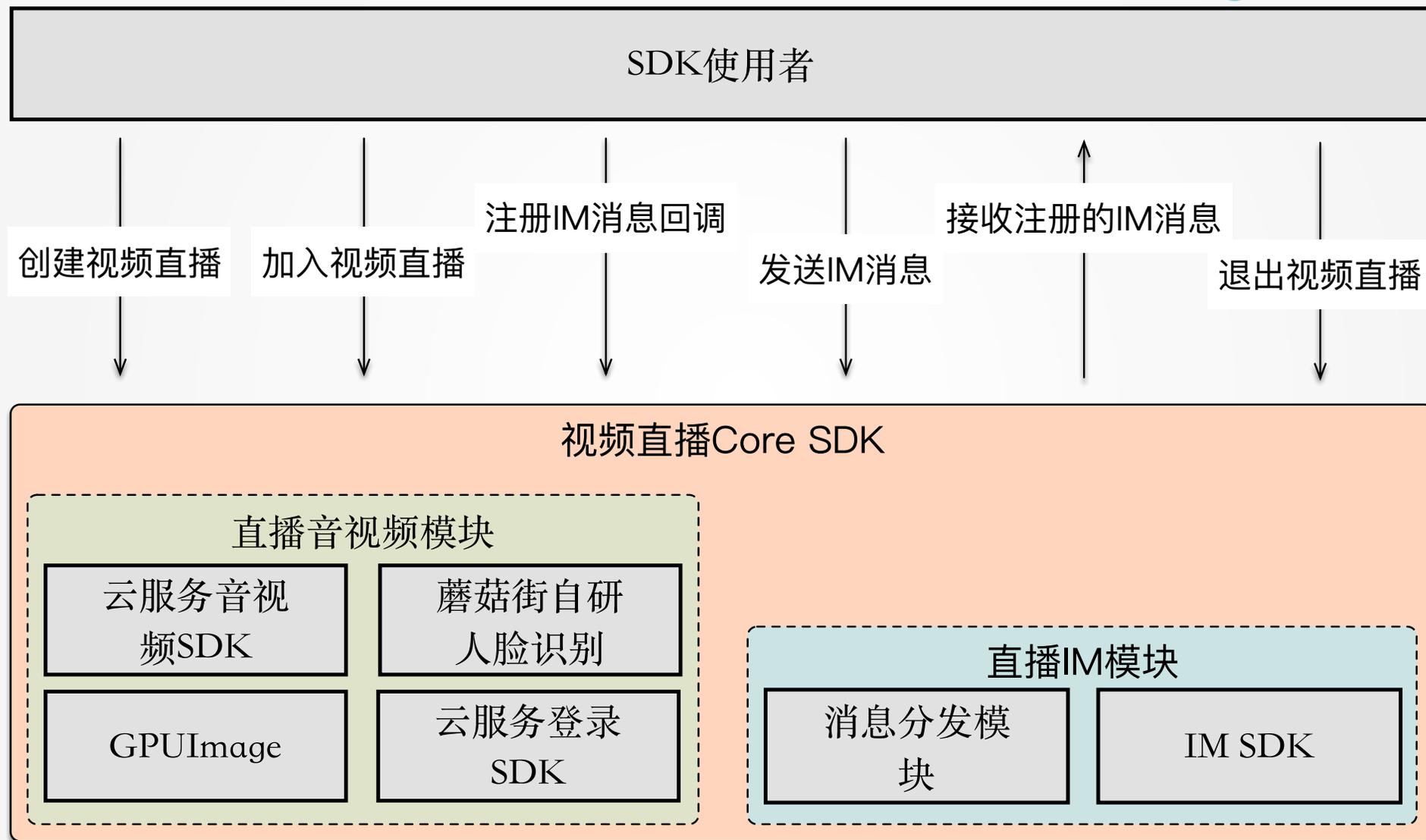
# SDK化

- 降低集团内其他App接入成本
- 统一的接口和文档
- 良好的框架设计和扩展性
- 业务功能可配置化和UI定制化

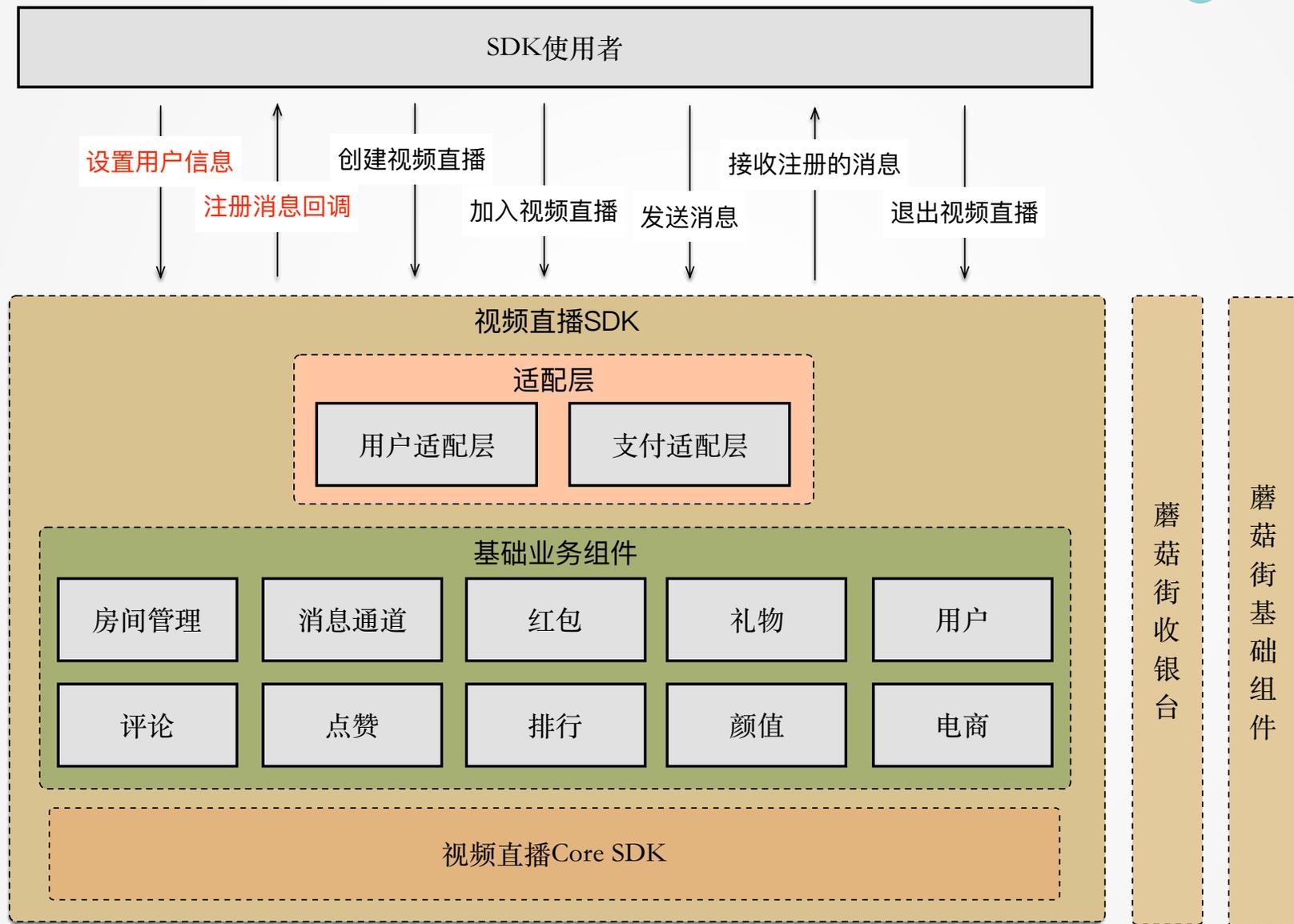
# 直播SDK化



# 直播SDK化 - Core



# 直播SDK化 - 业务

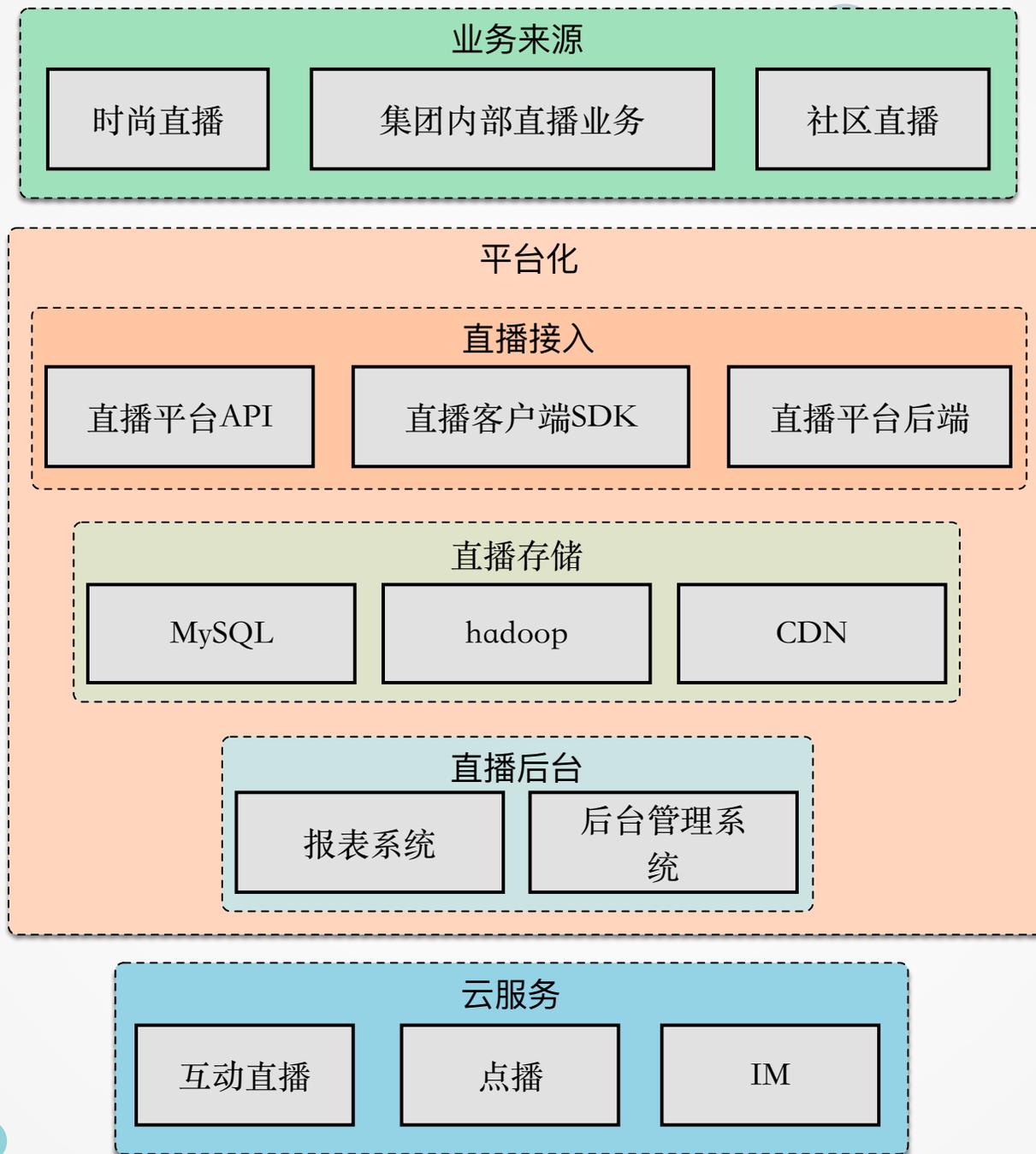




# 平台化

- 更好的业务方接入方式
- 更加便捷精准的数据平台

# 直播平台化 - 总体结构



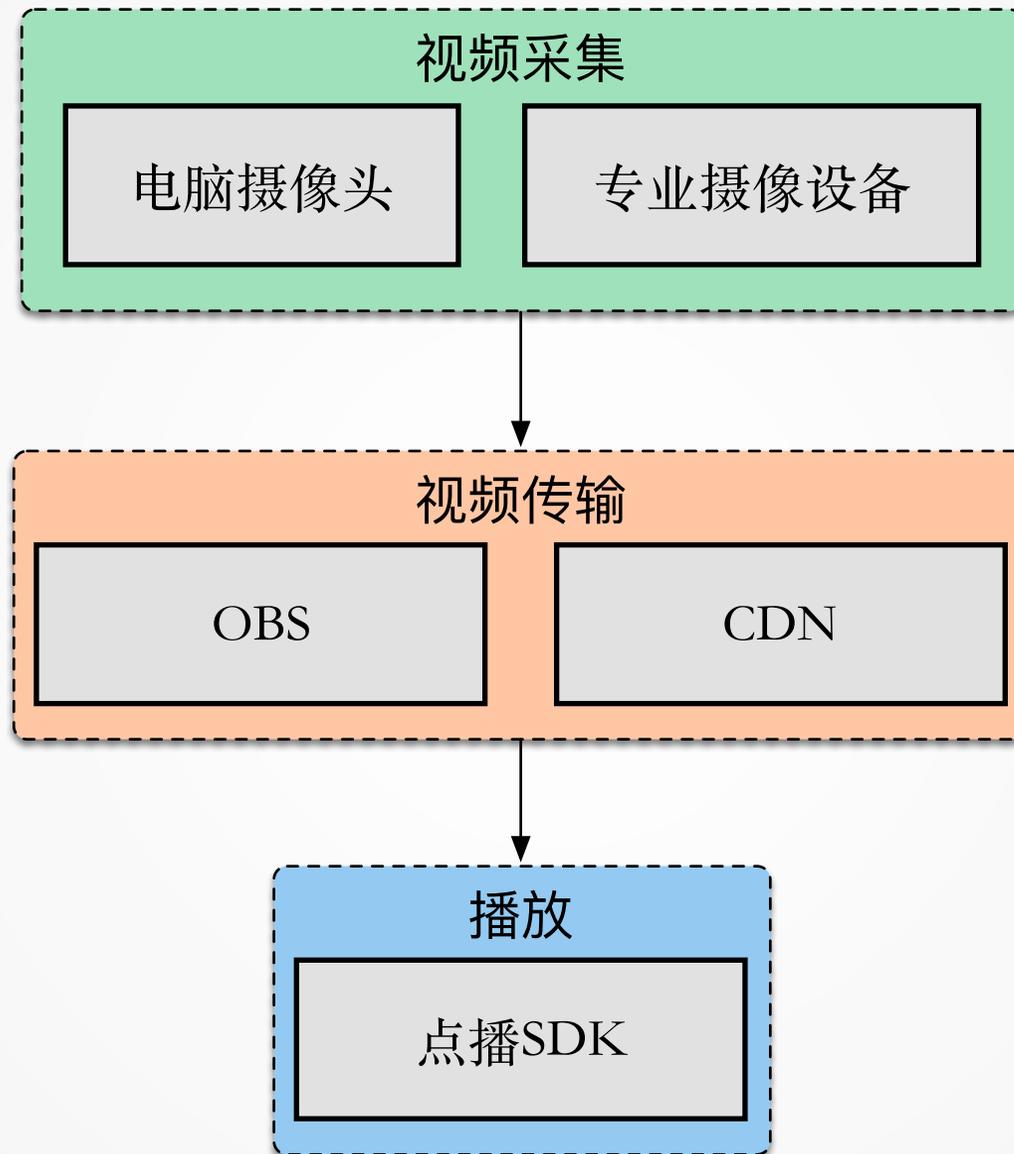


# 专业设备推流播放

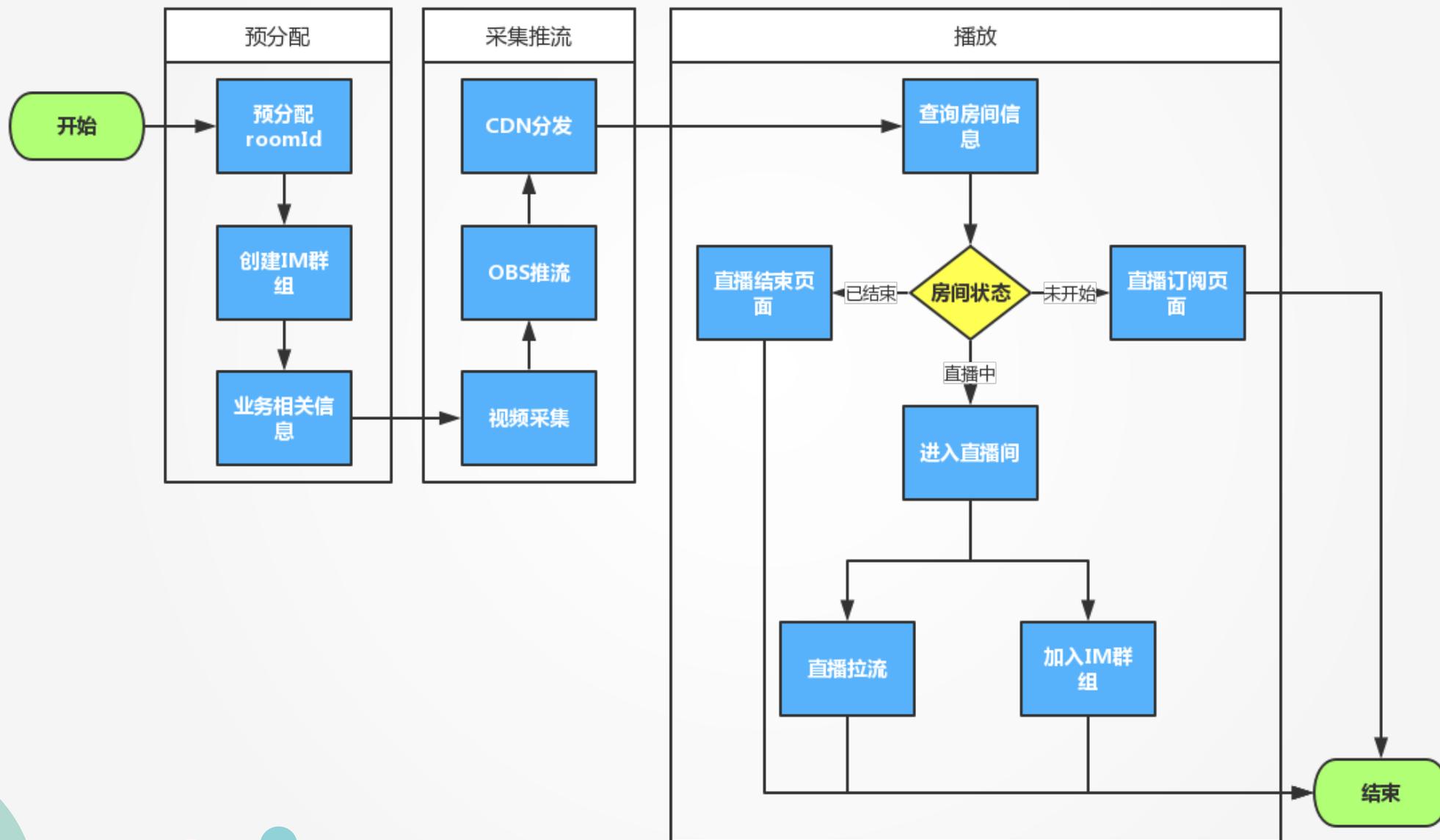
# 专业设备推流播放

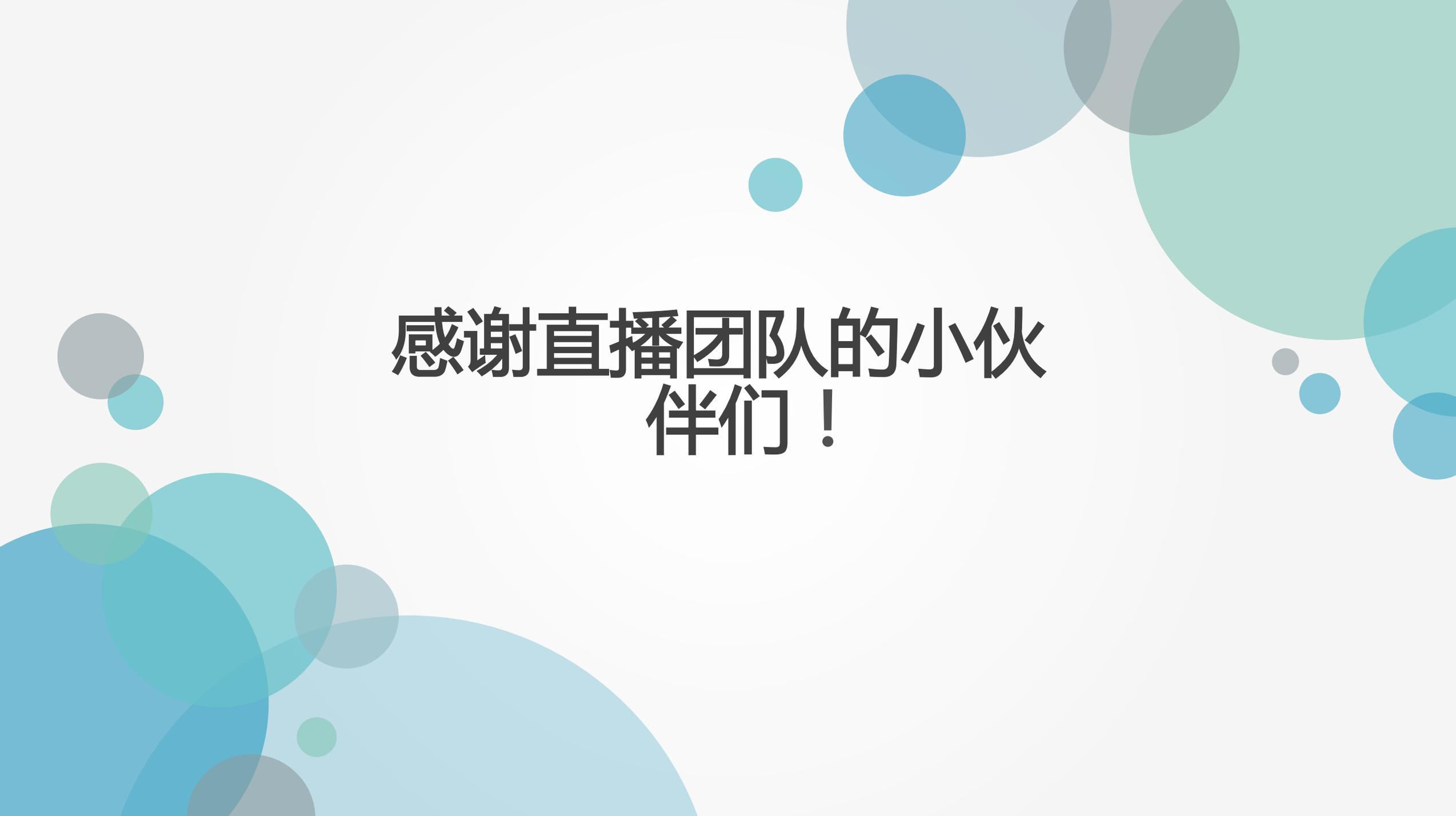


# 专业设备推流播放



# 流程图





感谢直播团队的小伙  
伴们！



谢谢大家！

蘑菇街-花荣



# Q & A