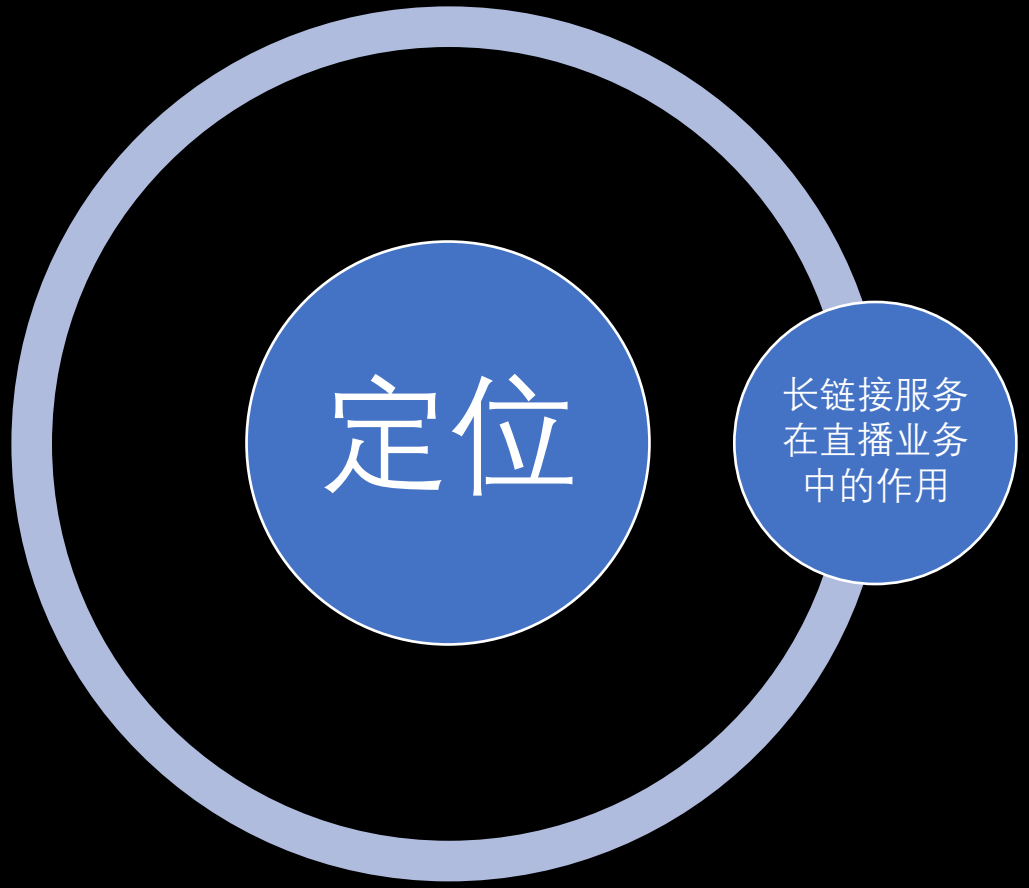
GOPHER CHINA 2020

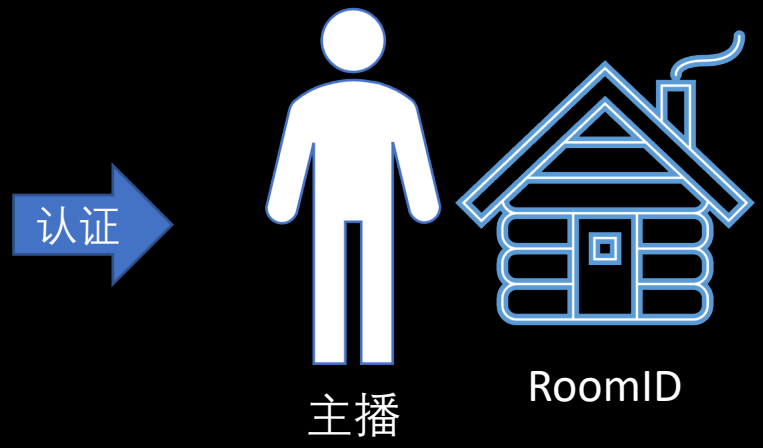
中国 上海 / 2020-11.21-22

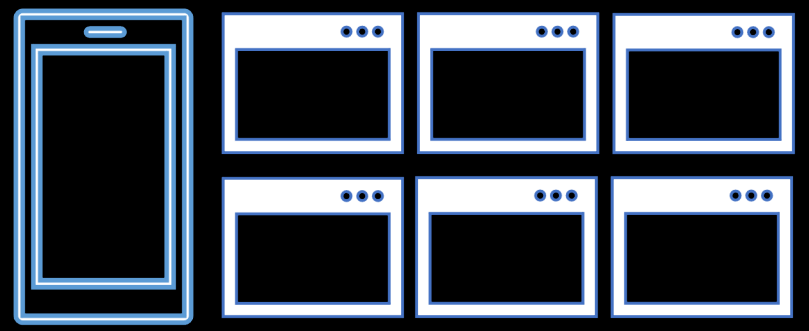
探探直播长链接架构的演进

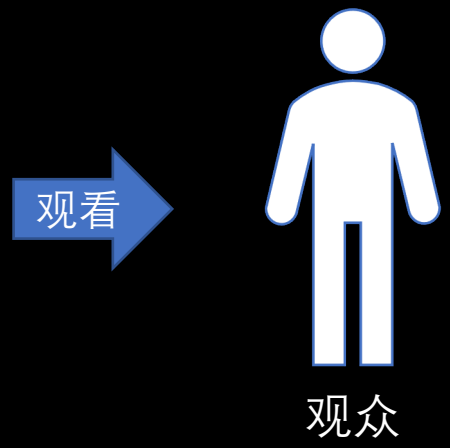
周长斌


定位
• 长链接服务在直播业务中的作用
架构
• 长链接服务的架构
演进
• 长链接服务的演进过程
总结
• 收获
• 规划


定位
长链接服务
在直播业务
中的作用



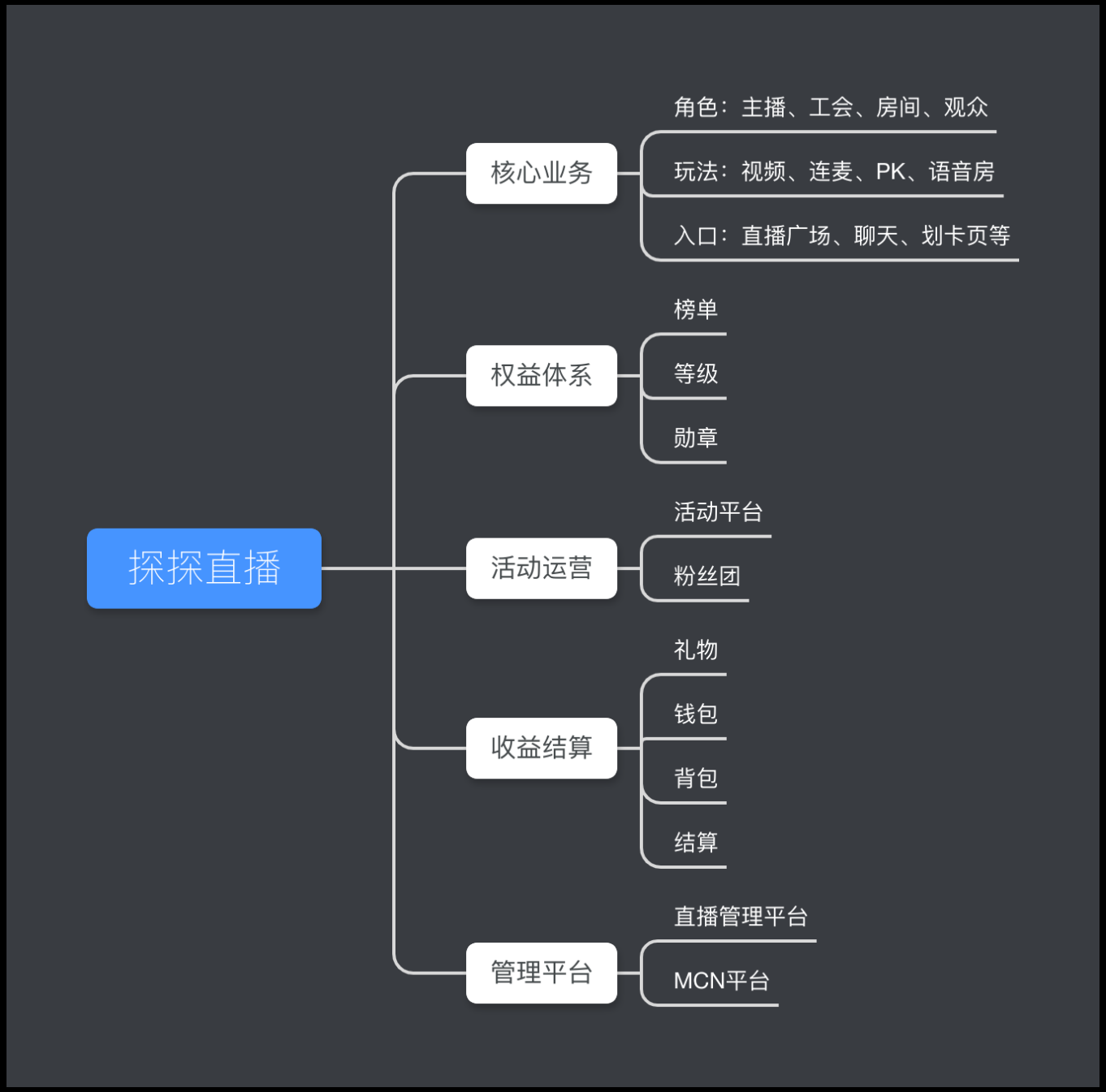


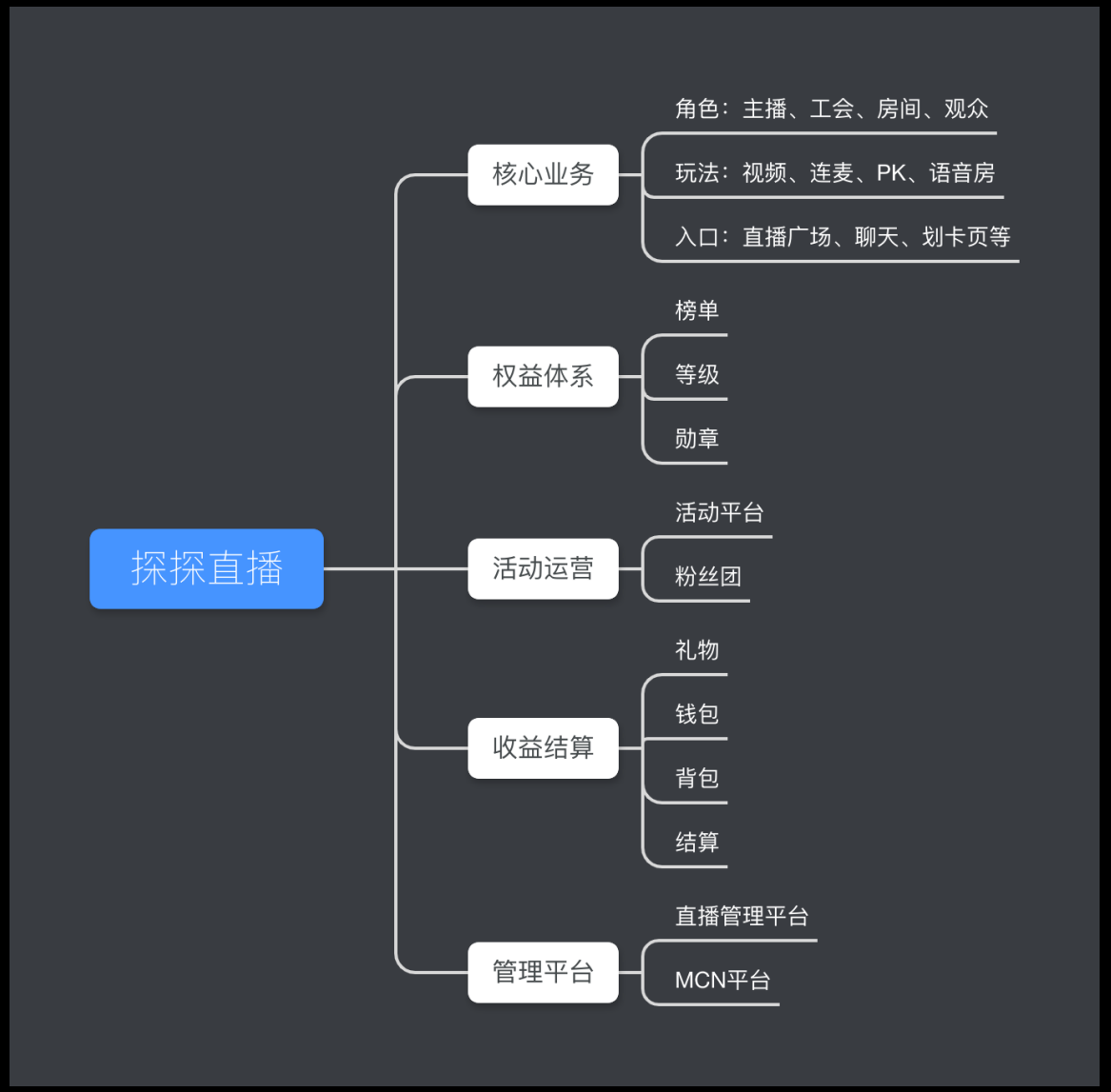


主播和观众都是普通用户，唯一标识：UserID

主播认证后绑定房间：RoomID

观众进入直播间观看直播，资源独立（重新连接长链接）

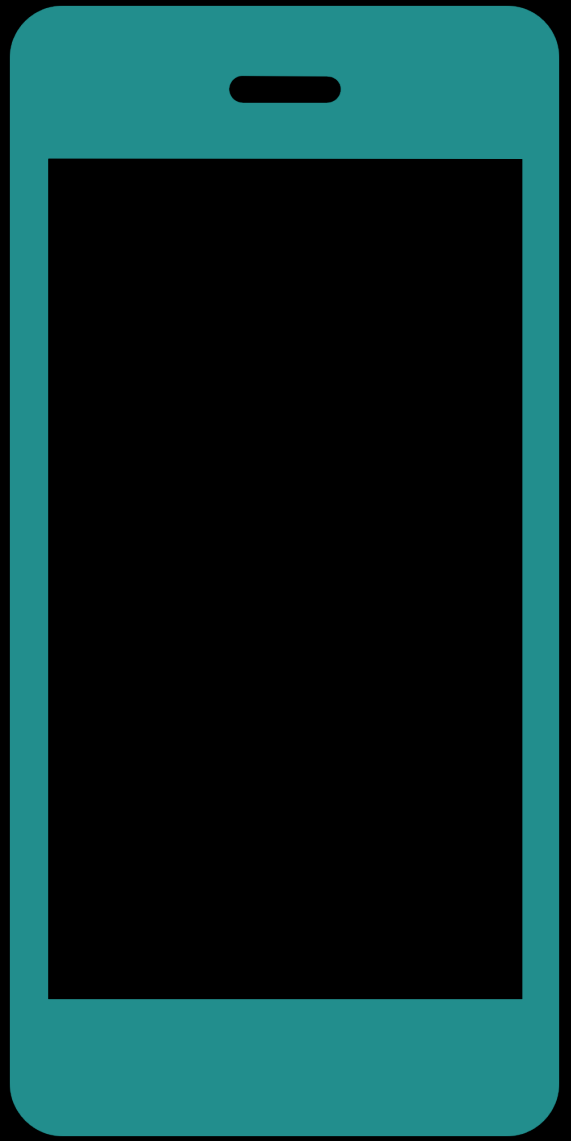

探探直播
核心业务
角色：主播、工会、房间、观众
玩法：视频、连麦、PK、语音房
入口：直播广场、聊天、划卡页等
权益体系
榜单
等级
勋章
活动运营
活动平台
粉丝团
收益结算
礼物
钱包
背包
结算
管理平台
直播管理平台
MCN平台

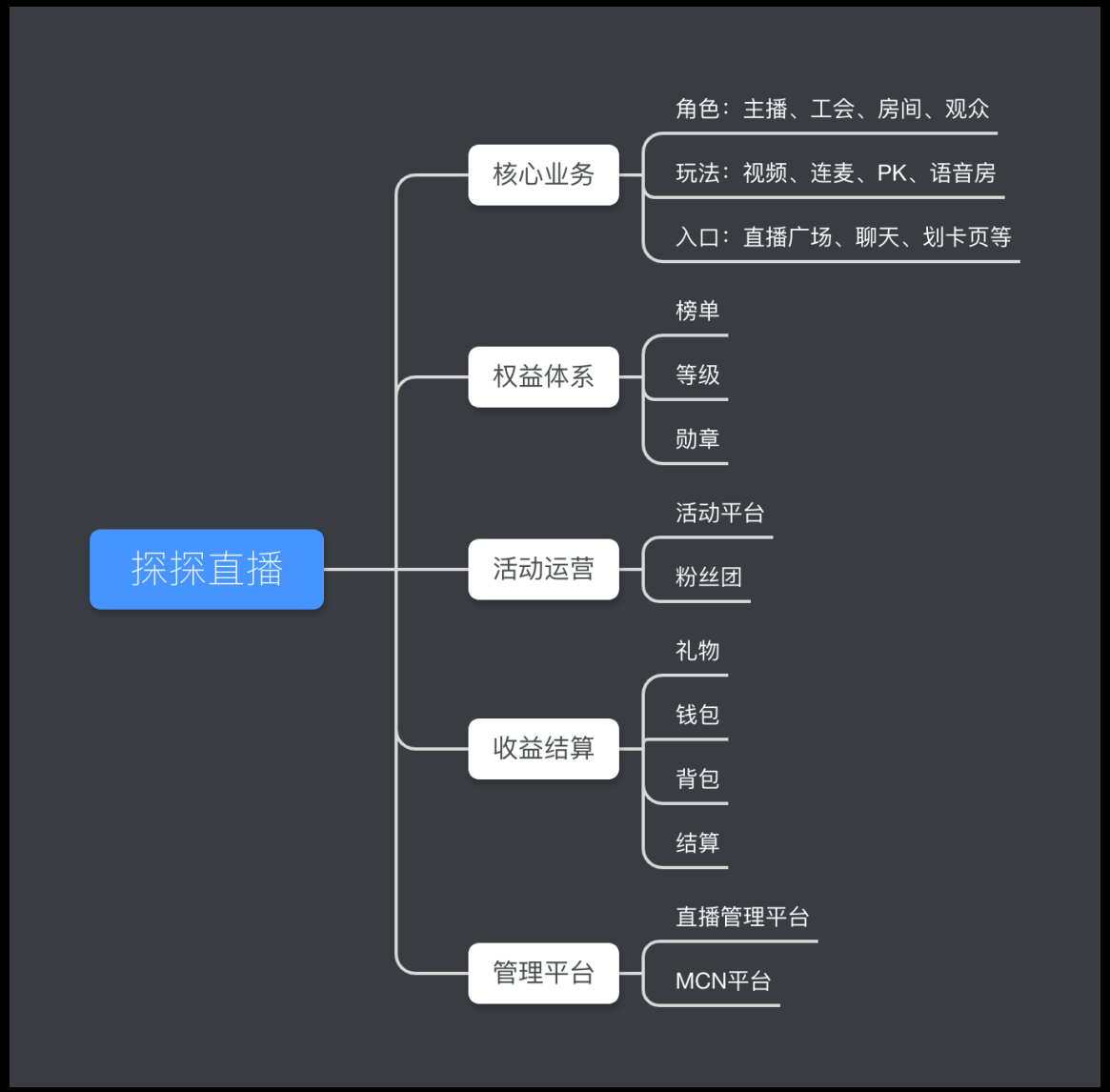


系统消息

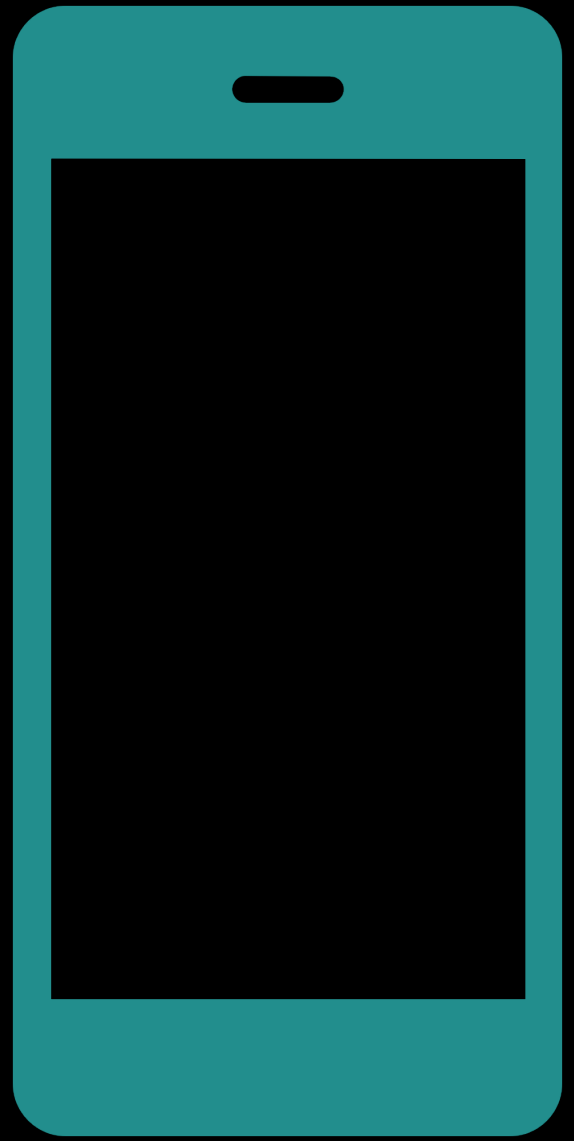
礼物消息

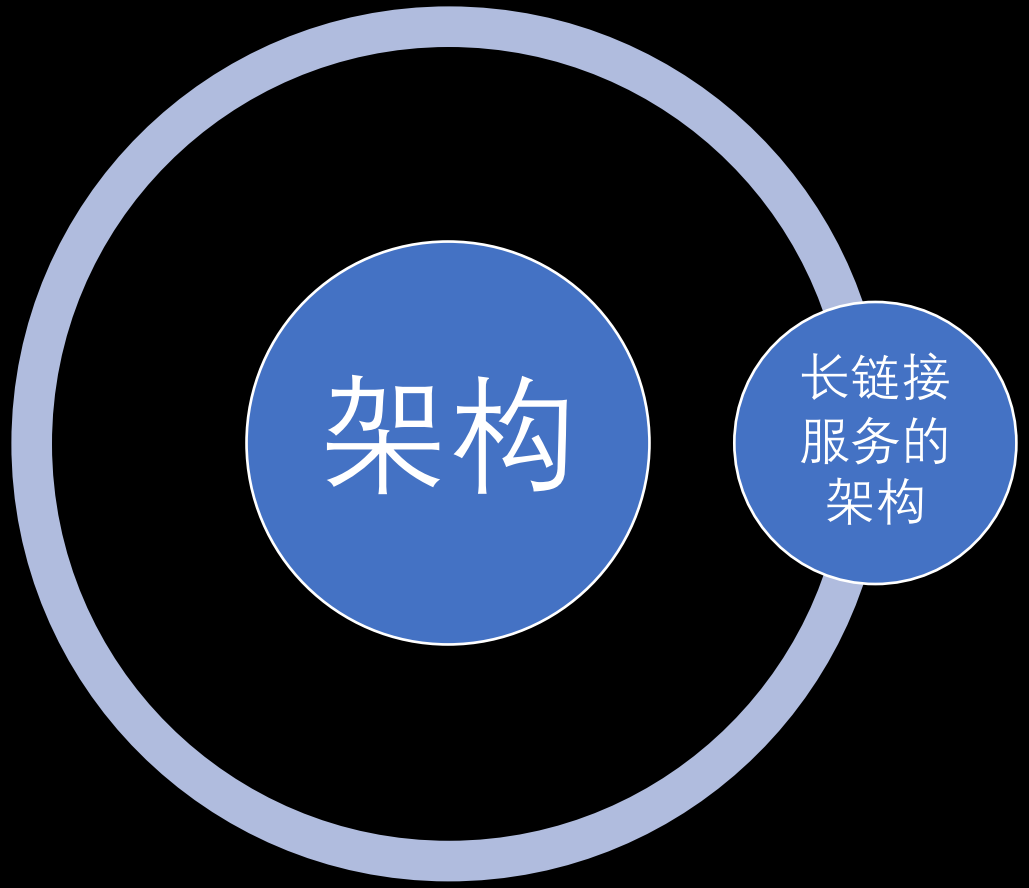
聊天消息

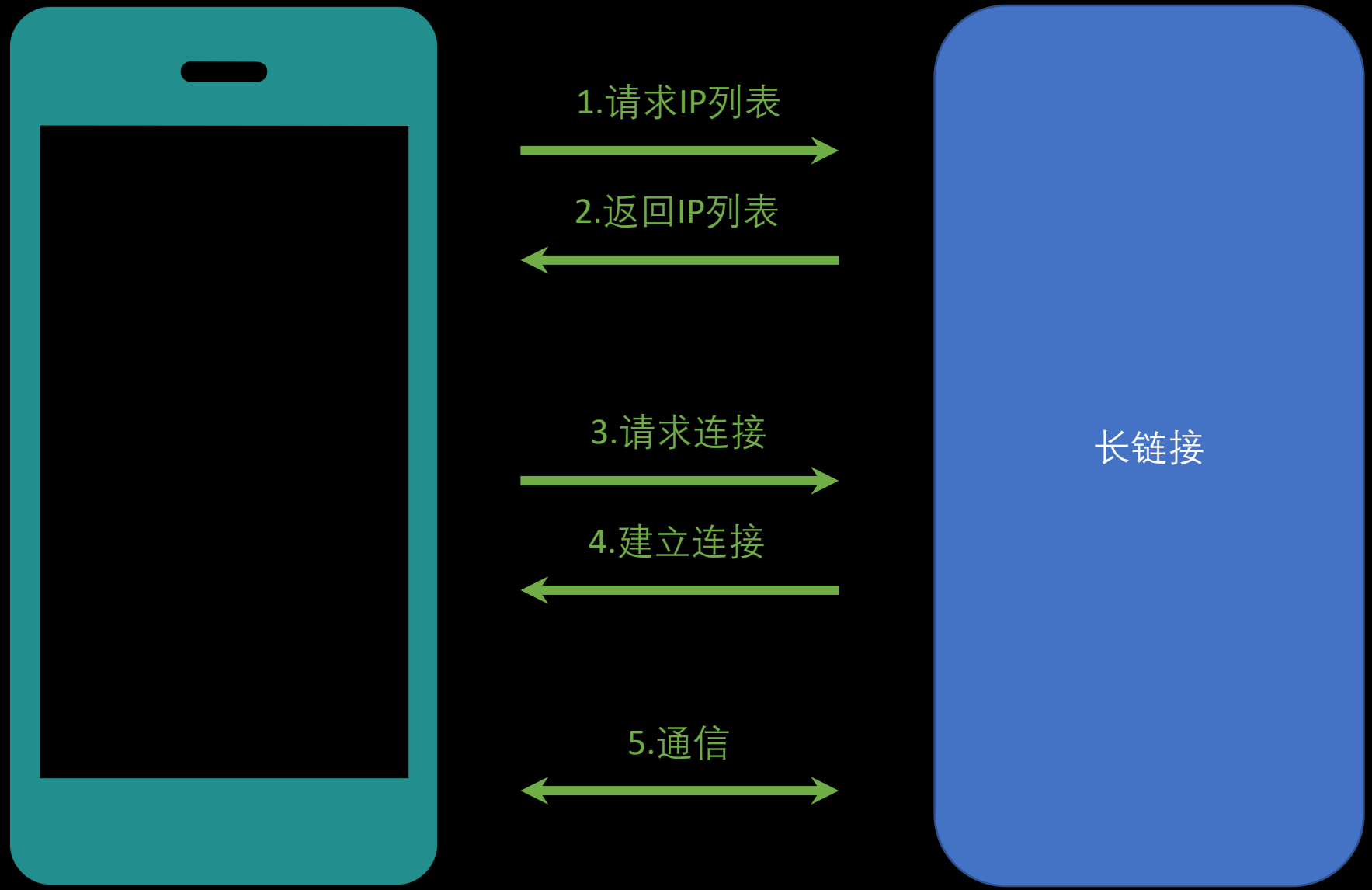


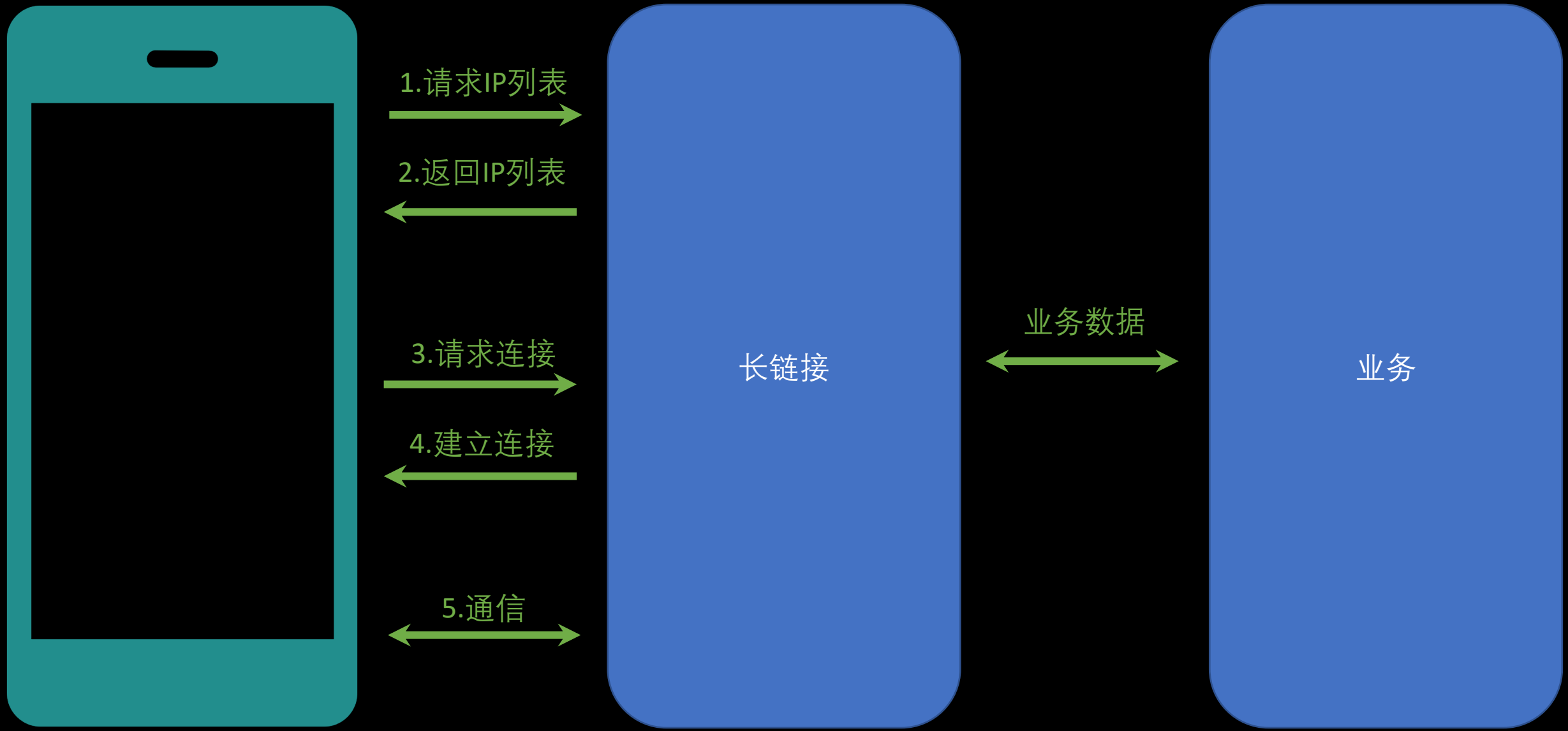

探探直播
核心业务
角色：主播、工会、房间、观众
玩法：视频、连麦、PK、语音房
入口：直播广场、聊天、划卡页等
权益体系
榜单
等级
勋章
活动运营
活动平台
粉丝团
收益结算
礼物
钱包
背包
结算
管理平台
直播管理平台
MCN平台

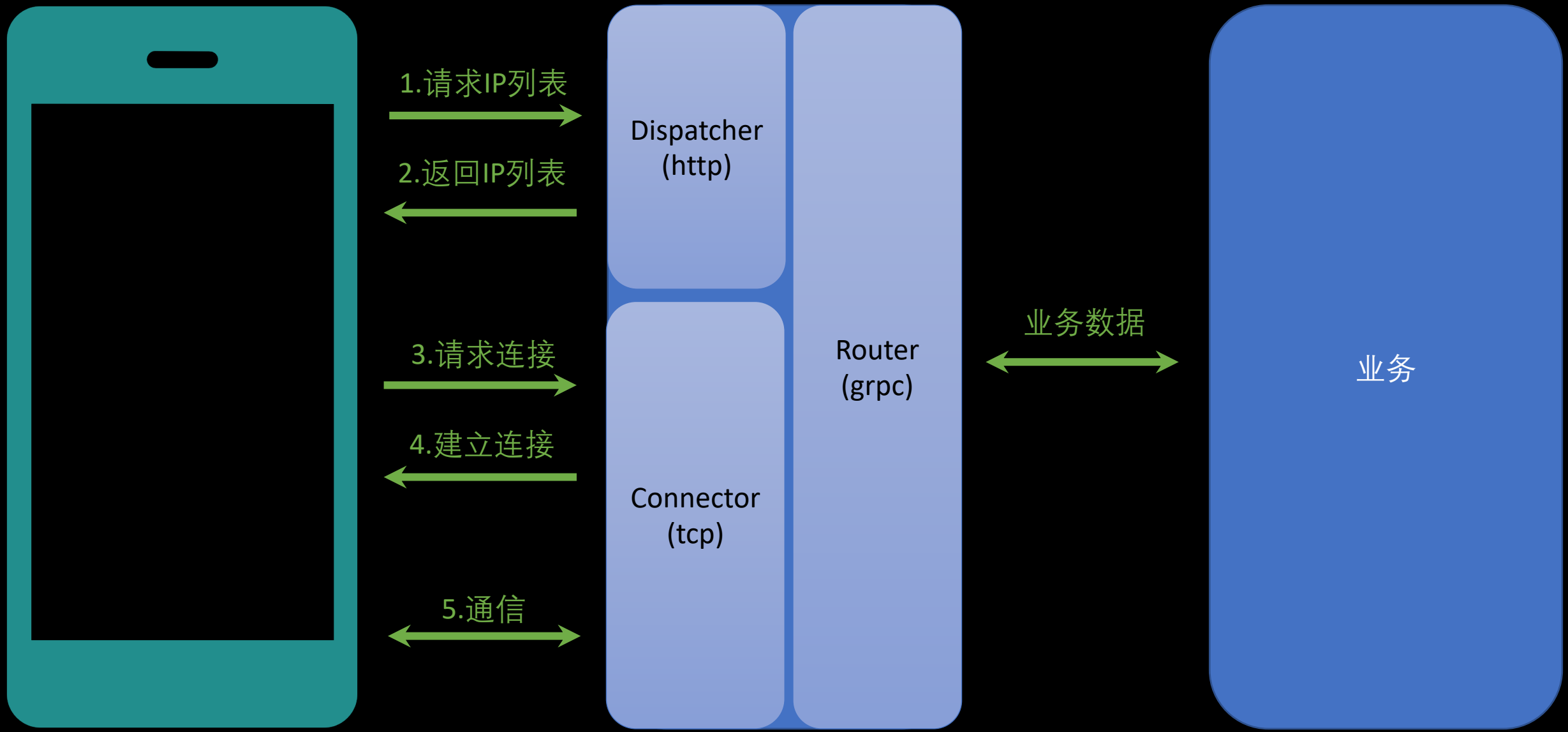

长链接

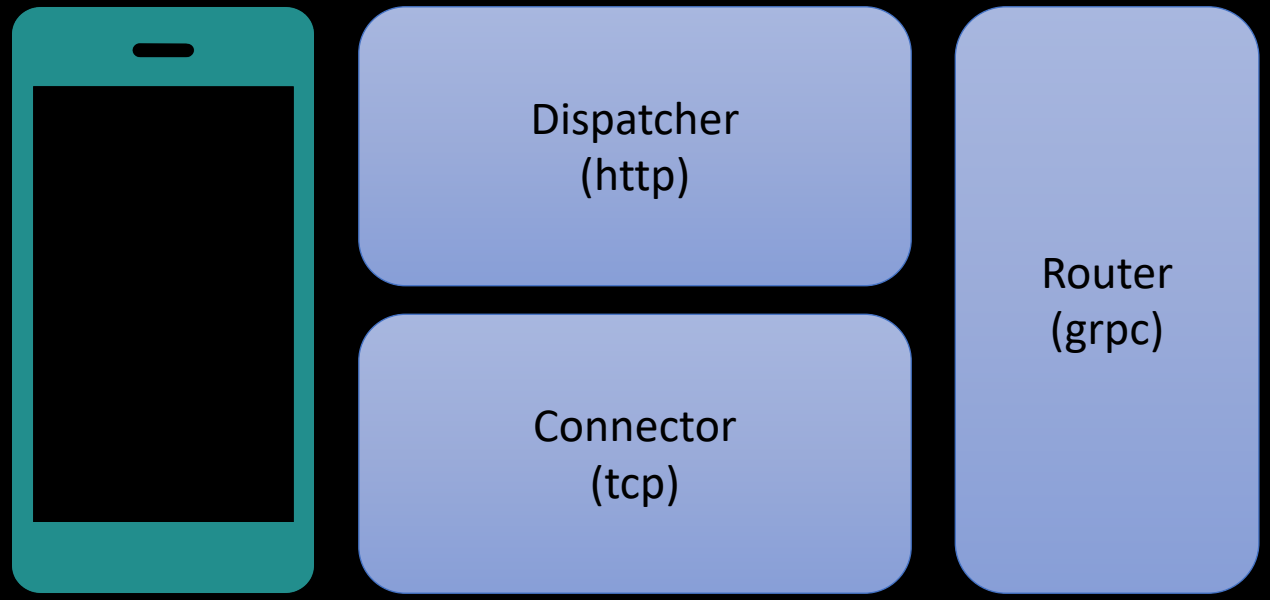


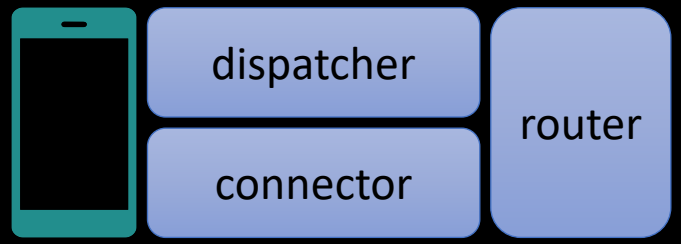

架构
长链接服务的架构


1.请求IP列表
2.返回IP列表
3.请求连接
4.建立连接
5.通信
长链接


1.请求IP列表
2.返回IP列表
3.请求连接
4.建立连接
5.通信
长链接
业务数据
业务


1.请求IP列表
2.返回IP列表
3.请求连接
4.建立连接
5.通信
Dispatcher
(http)
Connector
(tcp)
Router
(grpc)
业务数据
业务

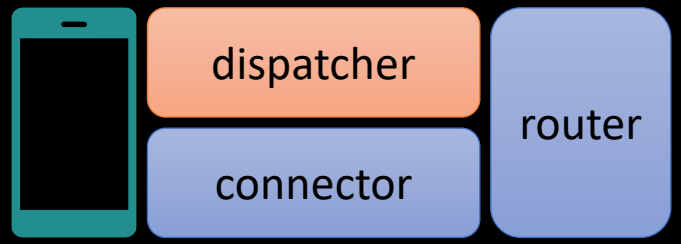

Dispatcher
(http)
Connector
(tcp)
Router
(grpc)



架构

HTTP
TCP
GRPC
接入层
长链接
业务

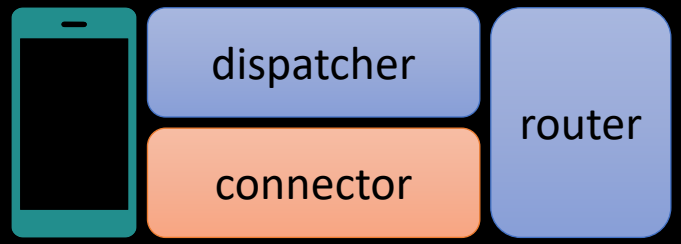
1. HTTP
Access Point
2. HTTP
3.TCP
Connector IP1
4. GRPC
Connector IP2
router
Connector IP3
5. GRPC
dispatcher
Live
Chat
Gift
Admin



Dispatcher

```
// 提供Http接口，返回长链接connector节点的IP列表
func (s DispatcherServer) Route(g *gin.Engine) {
    g.GET("/v2/live-metadata", s.liveMetadata)
}
```

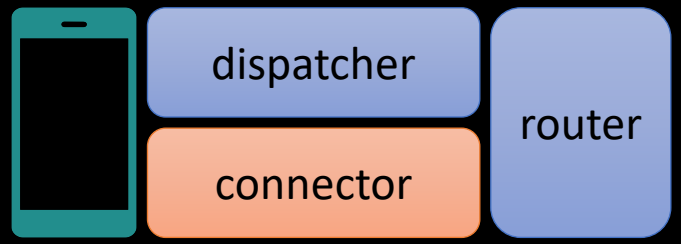
```
/*
 * 1. 随机返回4个IP地址
 * 2. 根据客户端信息返回对应的IP协议类型
 */
realTcpAddrs = allTcpAddrs
if len(allTcpAddrs) > 4 {
    realTcpAddrs, err = util.Random(allTcpAddrs, 4)
    if err != nil {
            realTcpAddrs =
[]string{allTcpAddrs[rand.Intn(len(allTcpAddrs))]}
    }
}
```



Connector

长链接数据结构

消息上行下行

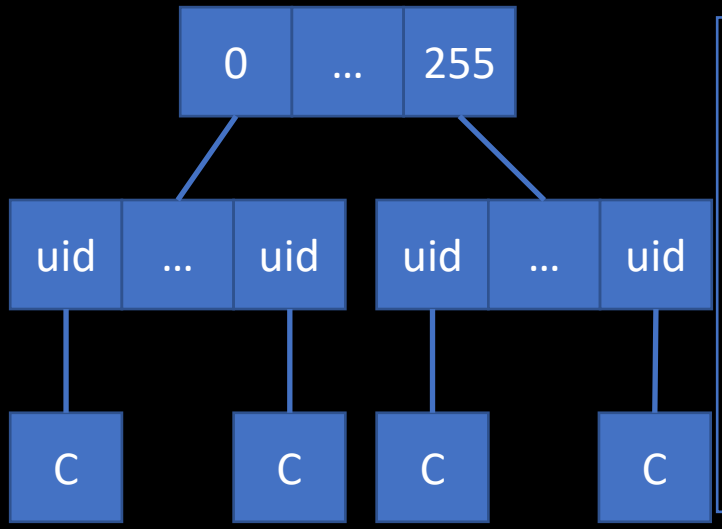


Connector

长链接数据结构

client
tcpConn UserID RoomID

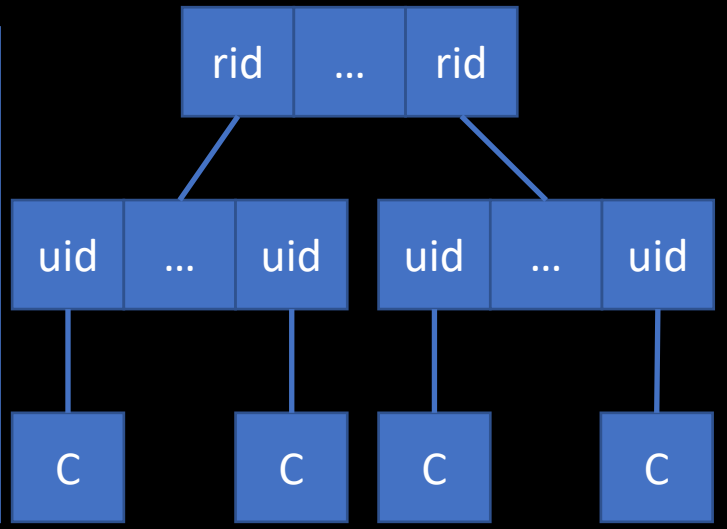
```
type Client struct {
    mutex sync.RWMutex
    // The tcp connection.
    tcpConn  *net.TCPConn
    // The user info.
    UserId int
    RoomId string
    roomType string
}
```



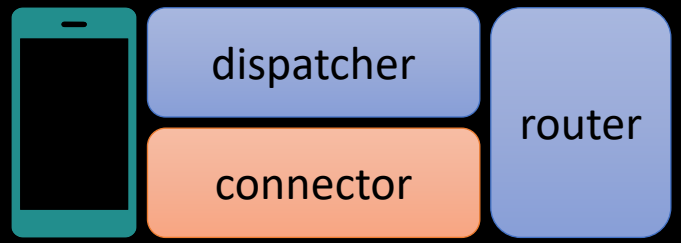

```
// Store client by userId.
var hub *Hub

type Hub struct {
// router is a array of 256 subRouter item
router []*subRouter
mutex  *sync.RWMutex
}

type subRouter struct {
//key is userId, value is Client point
r      map[int]*Client
mutex *sync.RWMutex
}
```

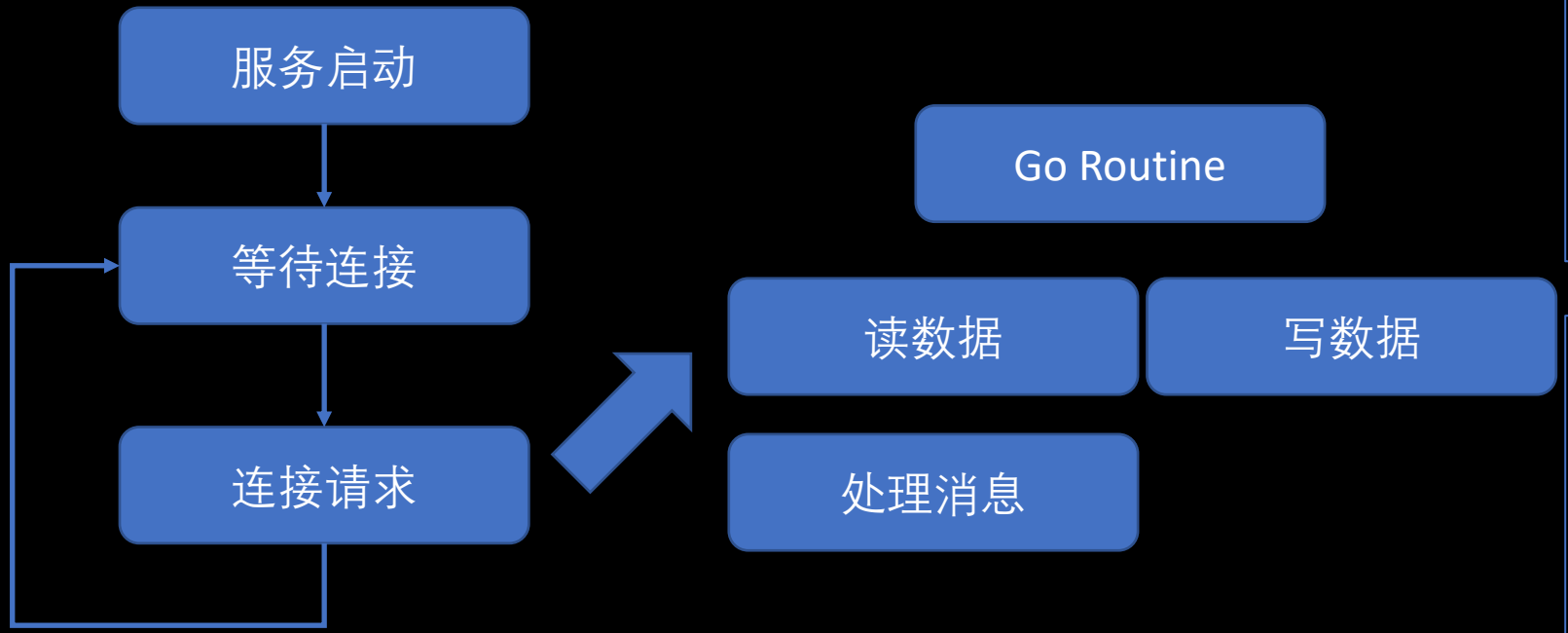



```
// Store client by roomId.
var tagMapList []TagMap
type Tag struct {
users    map[int]*Client
mutex    *sync.RWMutex
roomKey string
}
type TagMap struct {
tags  map[string]*Tag
mutex *sync.RWMutex
}
```



Connector

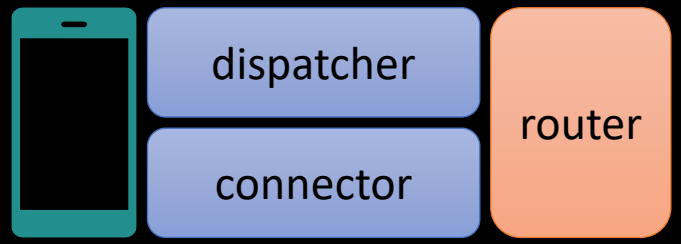
消息上行下行




```go
conn, err := listener.AcceptTCP()
if err != nil {
    slf.Errorw("AcceptTCP", slf.Error(err))
    continue
}
client := NewClient(conn)
go client.readPump()
```

```go
func (c *Client) readPump() {
    for {
        // Read.
        tcpData, err := c.dataPack.Unpack(c.tcpConn)
        // Handle message.
    }
}
```

```go
func (c *Client) write(data []byte) error {
    c.mutex.Lock()
    defer c.mutex.Unlock()
    if _, err := c.tcpConn.Write(data); err != nil {
        return err
    }
    return nil
}
```



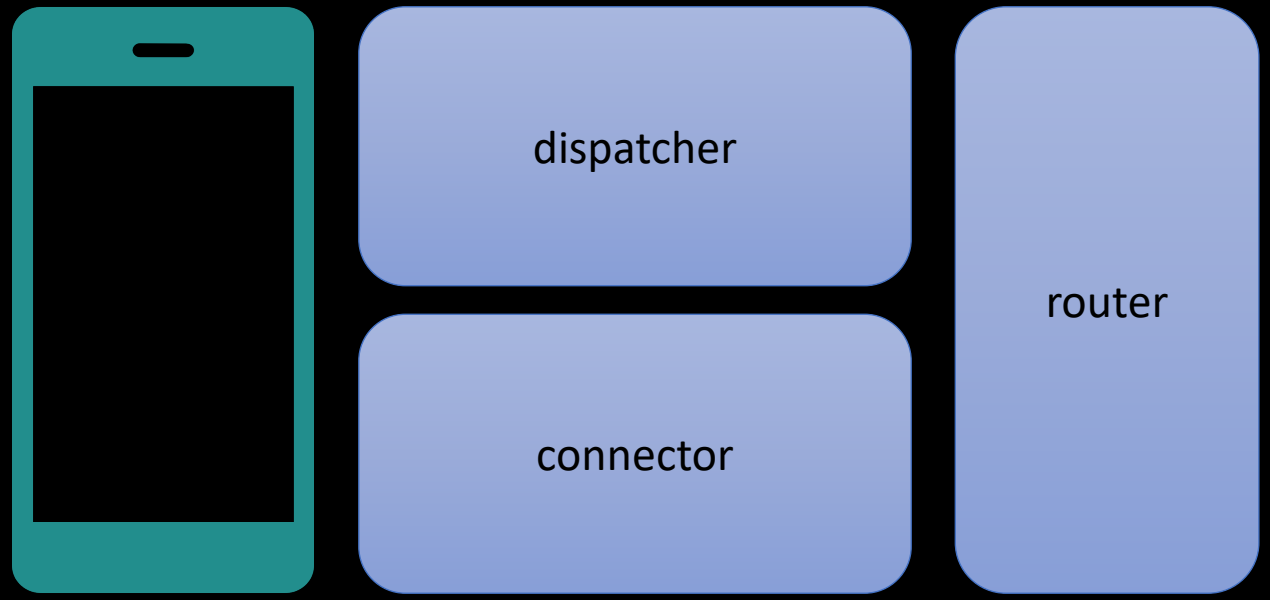
Router

```
service LiveRouterService {
    rpc RegisterRouter(RegisterRouterRequest) returns (RouterReply) {}  // connector 注册
    rpc UnregisterRouter(UnregisterRouterRequest) returns (RouterReply) {} //删除注册信息

    rpc GetConnInfo(GetConnInfoRequest) returns (GetConnInfoReply) {}   // 获取连接信息

    rpc TransferUnicastMsg(TransferUnicastMsgRequest) returns(RouterReply) {} // 单播
    rpc TransferMulticastMsg(TransferMulticastMsgRequest) returns(RouterReply) {} // 组播
    rpc TransferBroadcastMsg(TransferBroadcastMsgRequest) returns(RouterReply) {} // 广播
}
```

```
for _, liveConnectorClient := range liveConnectors {
    newCtx := tracing.PropagateContextWithServiceContext(ctx)
    go func(ctx context.Context, connectorCli *rpcclient.LiveConnectorClient) {
        defer commonutil.Recovery()
        connectorCli.TransferMessage(ctx, convertMsg)
    }(newCtx, liveConnectorClient)
}
```


dispatcher
connector
router

代码

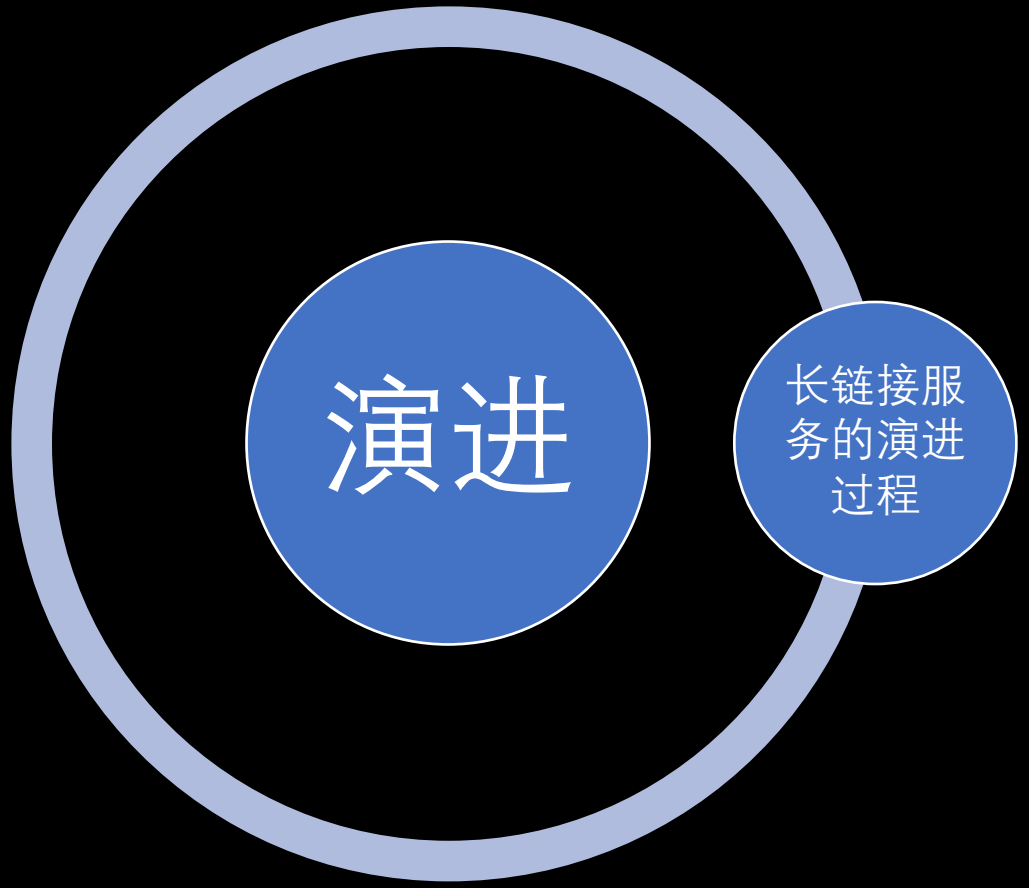
- 100% GO
- 精简

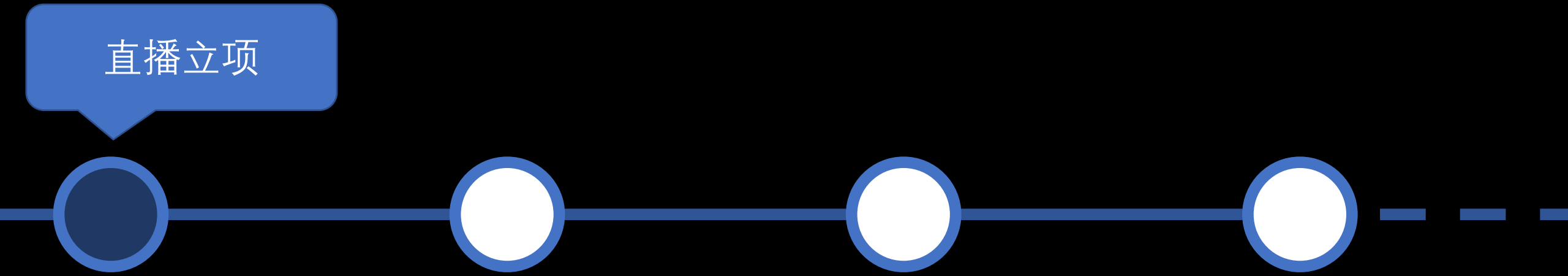
效率

- V1.0 三周 1人
- 4个大版本
- 发现和解决问题的速度快

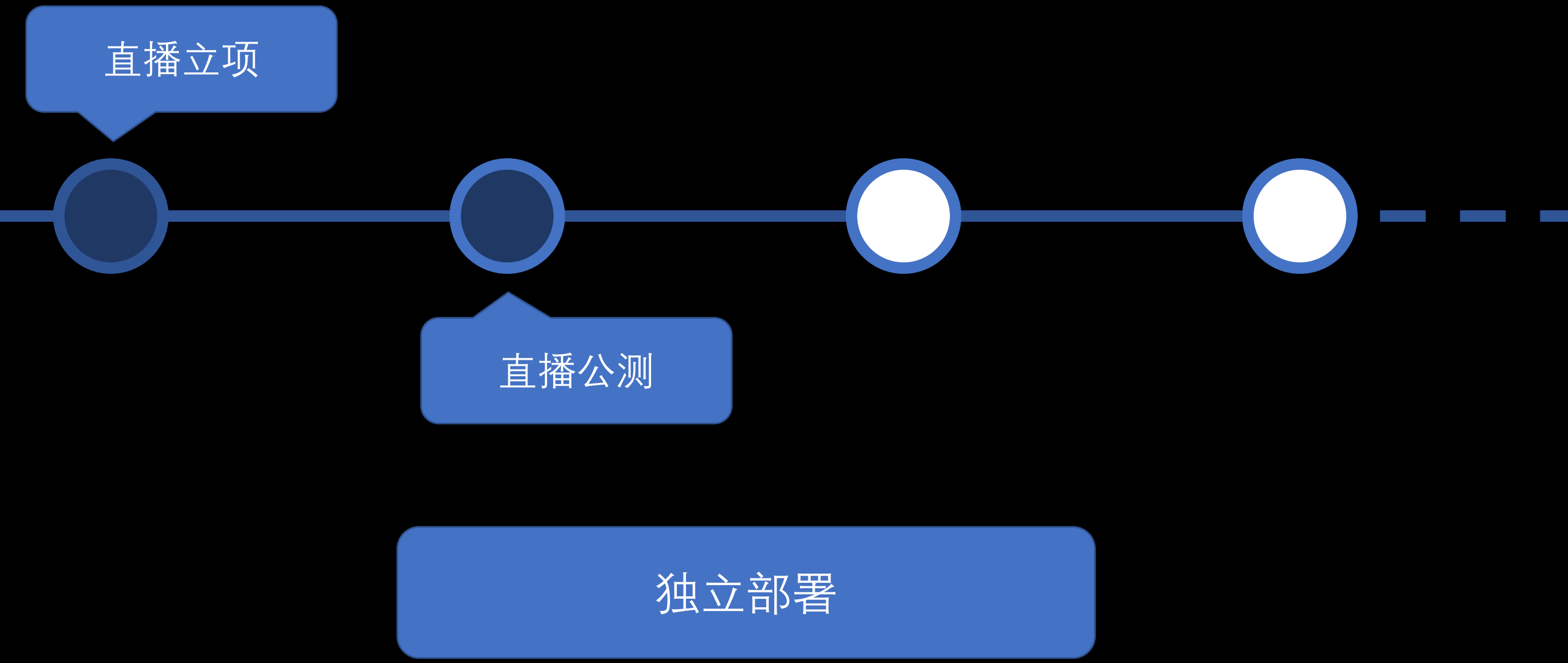
容量

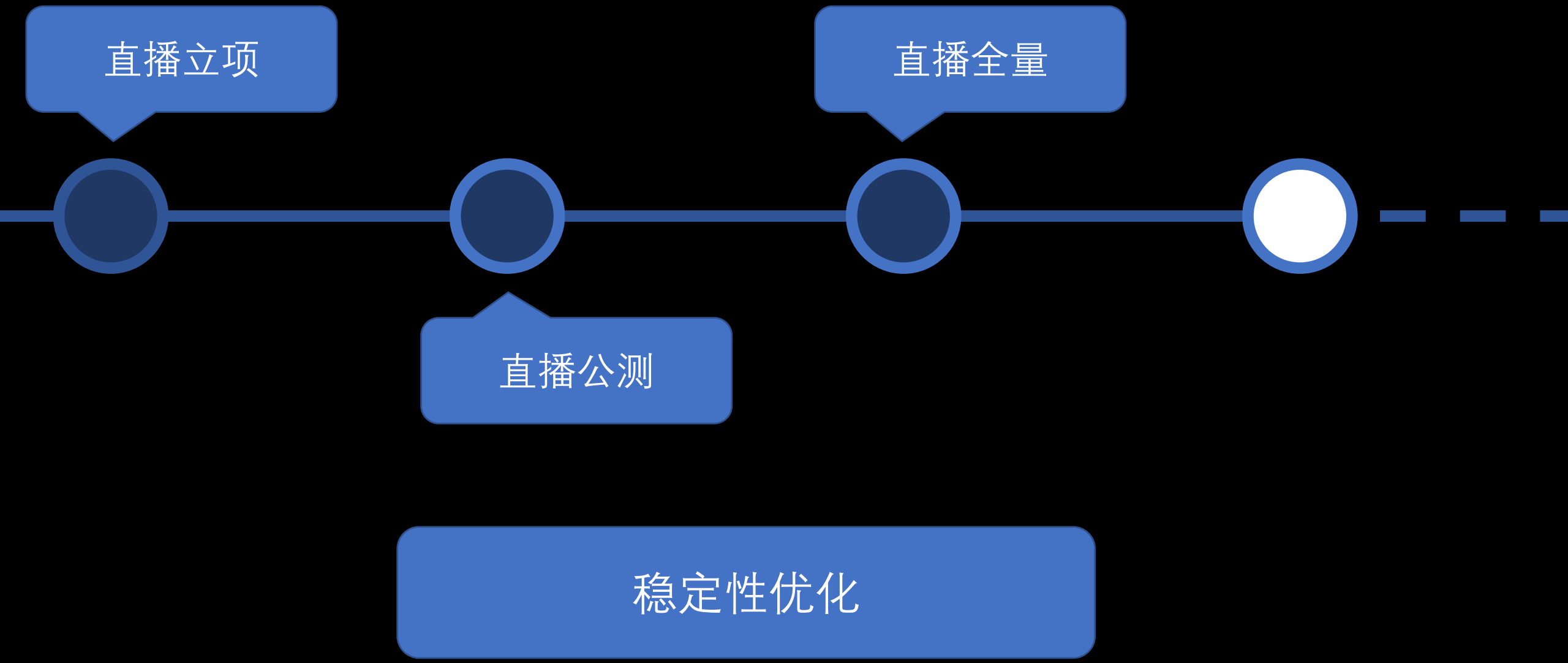
- W级链接数
- 带宽 600M/6G

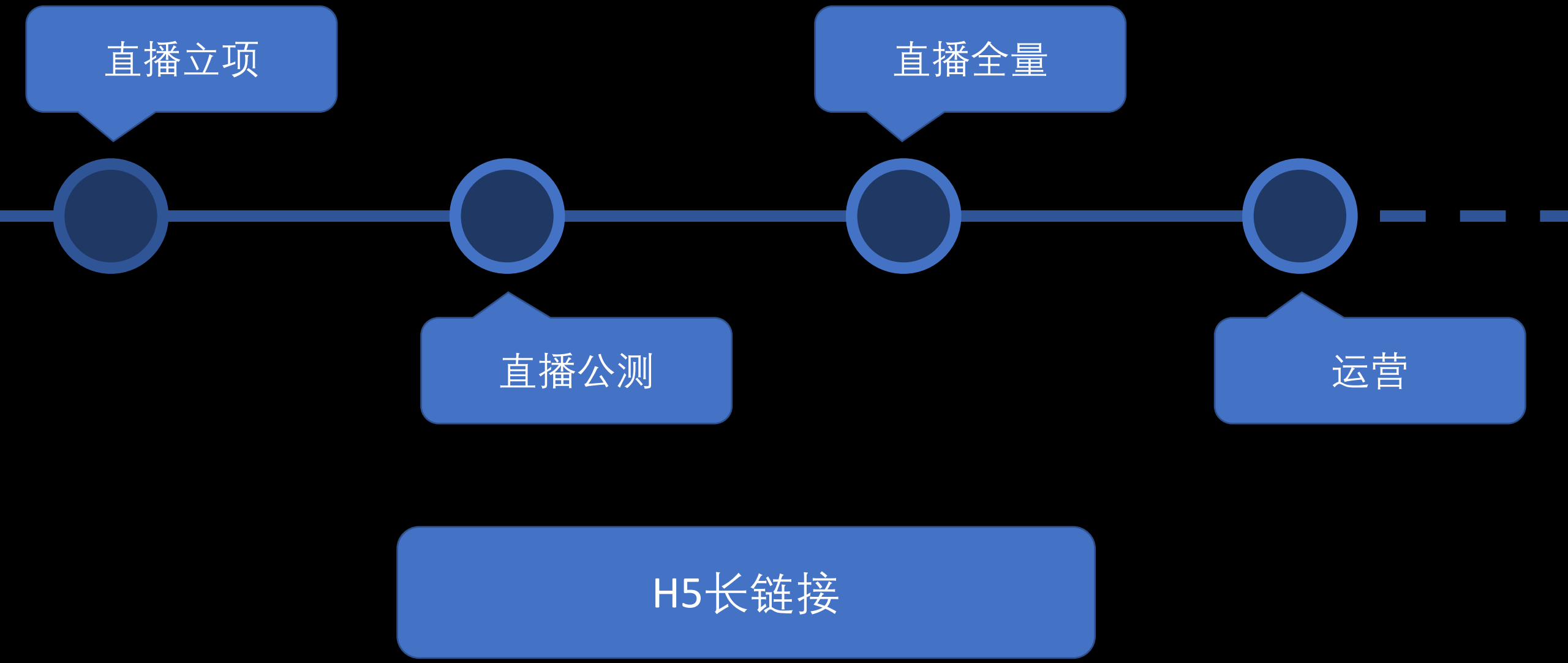

演进
长链接服务的演进过程



快速实现


直播立项
直播公测
独立部署


直播立项
直播公测
直播全量
稳定性优化


直播立项
直播公测
直播全量
运营
H5长链接

直播立项:快速实现

探探长链接 V1.0	直播长链接 V1.1
代码	分支：feature/live
协议	广播/组播 RoomID
部署	以AB实验的方式部署

直播立项:快速实现

```
for _, client := range roomClients {
        go client.sendMessage(msg)
}
```

问题：房间人数多 消息量大
2000人 QPS 400 ==> 80W go routine
Go routine 泄漏

```
msgEventChan := make(chan MessageEvent, ChanSize)
```

```
func (h *Hub) runSendMessage(poolSize int) {
    for i := 0; i < poolSize; i++ {
        go func() {
            for event := range h.msgEventChan {
                msg := event.message
                event.client.sendMessage(msg, event.msgInfo)
            }
        }()
    }
}
```

```
for _, c := range clients {
    msgEventChan <- MessageEvent{client : c , message: msg }
}
```

直播公测：面临的问题

业务快速发展

代码耦合

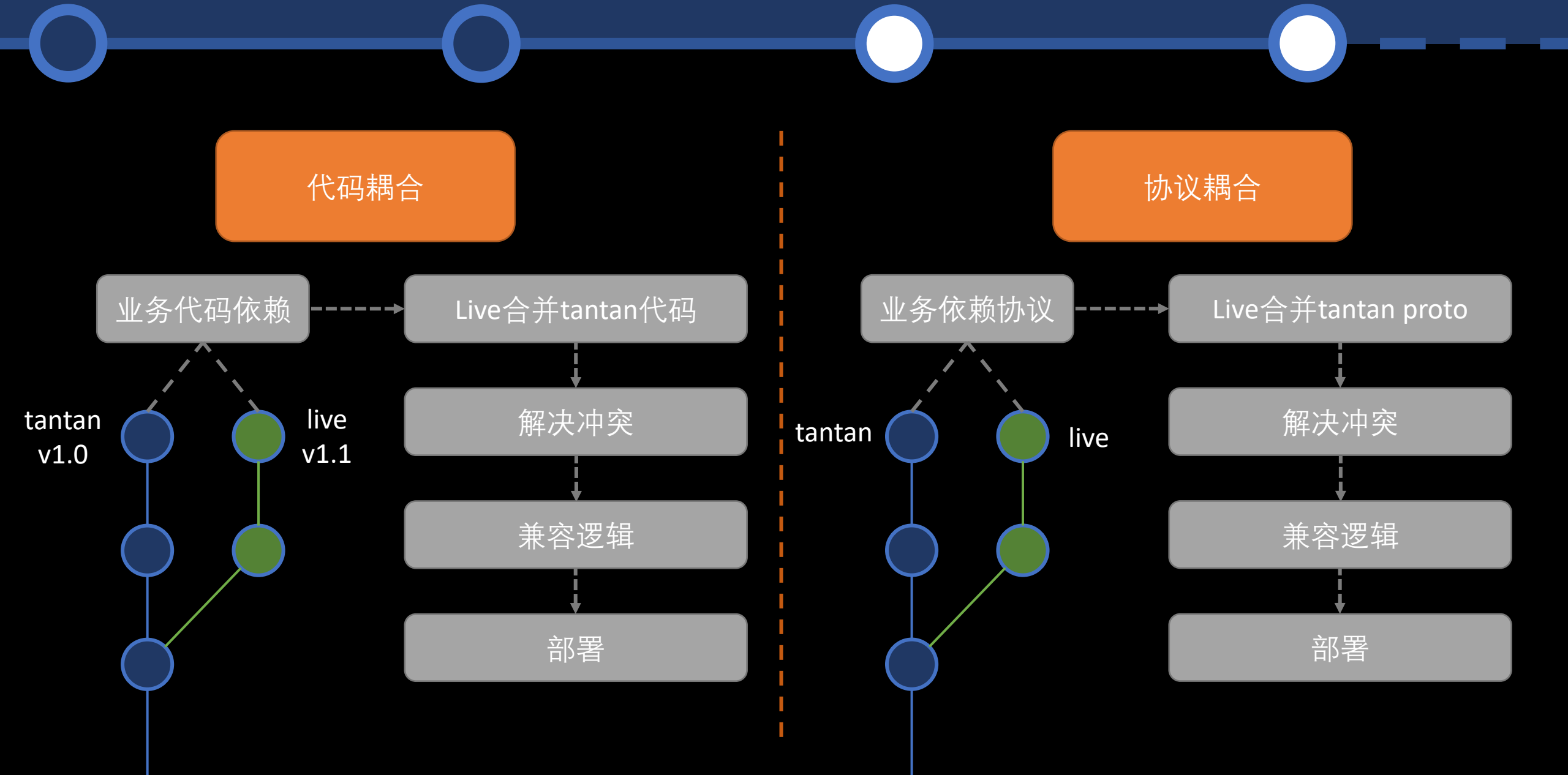
业务同时依赖v1.0 和 v1.1 需要兼容

协议耦合

改动频繁，需要兼容

业务改动需要长链接支持的时候，效率较低

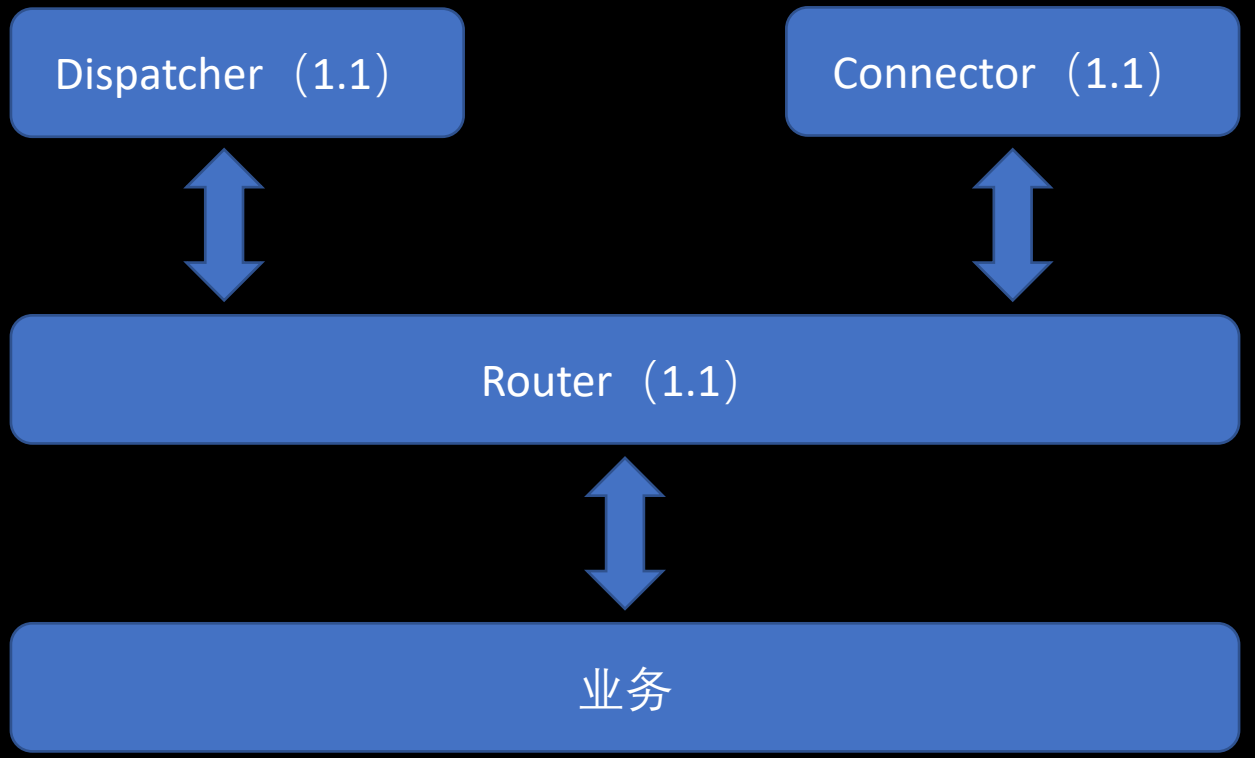
直播公测：面临的问题


代码耦合
业务代码依赖
Live合并tantan代码
tantan
v1.0
live
v1.1
解决冲突
兼容逻辑
部署
协议耦合
业务依赖协议
Live合并tantan proto
tantan
live
解决冲突
兼容逻辑
部署

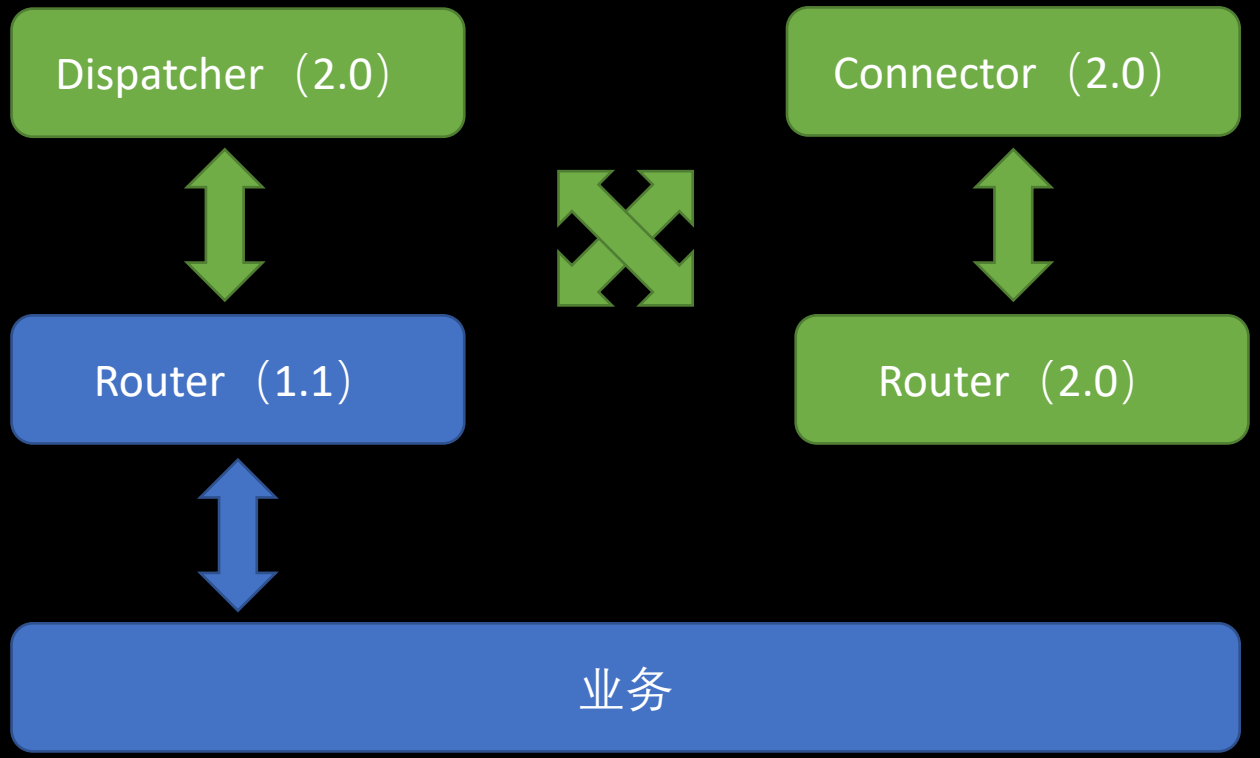
直播公测:独立部署

直播长链接 V1.1	直播长链接 V2.0
代码	独立代码仓库
协议	独立proto协议/消息改为string
部署	申请服务，独立部署

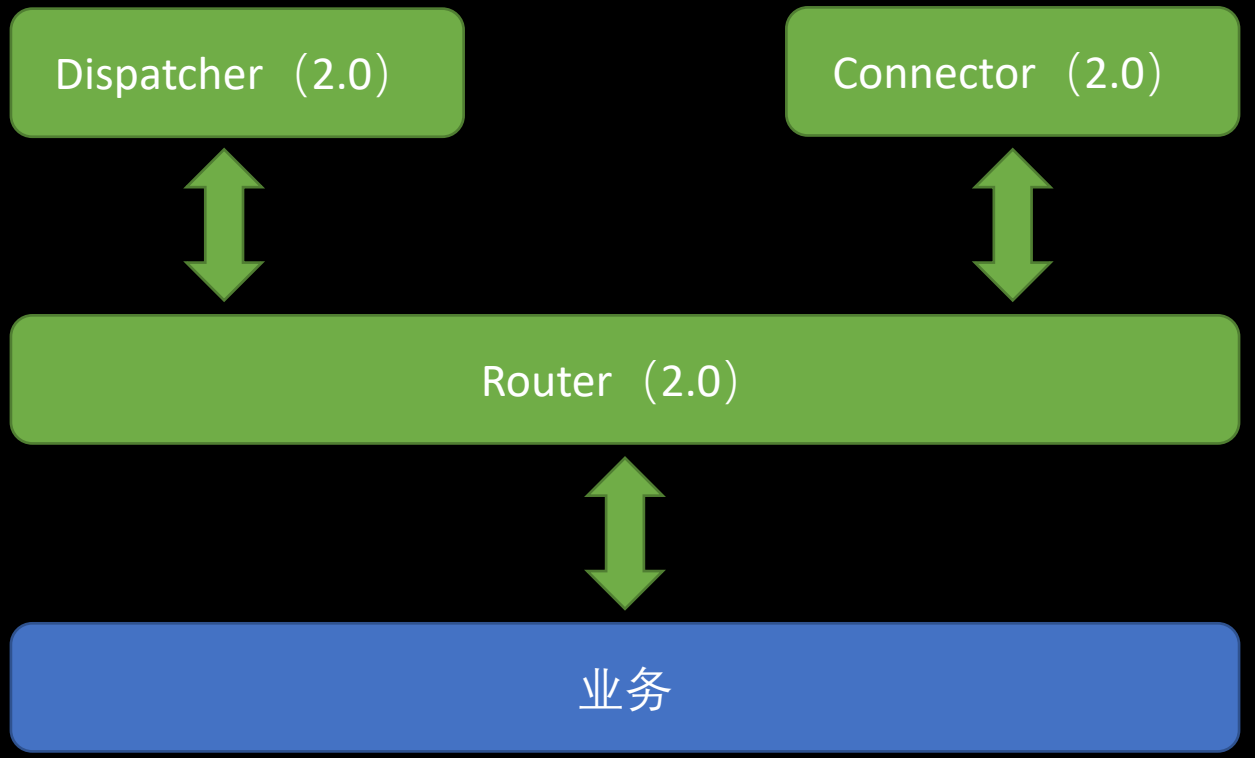
直播公测:独立部署



直播公测:独立部署



直播公测:独立部署



直播全量:稳定性优化

稳定性

压测

链接数量（内存）：单机20W * 20

消息量（带宽/网卡）：单机30W PPS → 2000 * 150 QPS

生产环境

礼物连击

活动消息

状态更新

突发流量

关键：控制消息量

直播全量:稳定性优化

关键：控制消息量

礼物连击

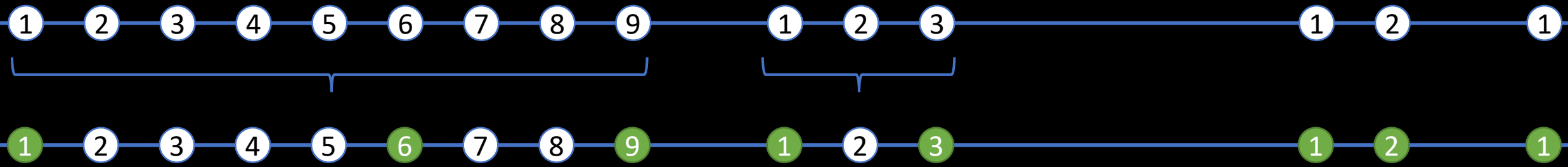
活动消息

状态更新

突发流量

直播全量:稳定性优化

礼物连击



礼物连击：连续两次送同样礼物时间小于2秒

极限可以每秒可以点击5 ~ 7次
连击一般是小礼物

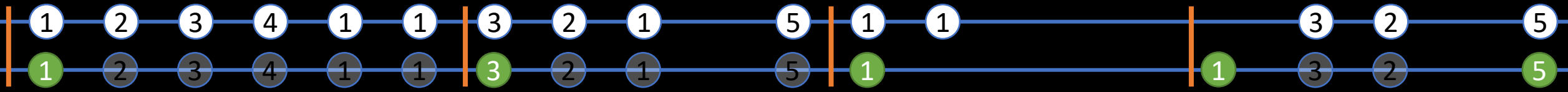
1. 聚合连击消息，每秒1条
2. 连击结束后，补发一条最终结果

==========

极限情况可以5~7条消息可以只发1条

直播全量:稳定性优化

状态更新



PK进度条

==========

两个主播可以进行PK
通过收礼物的数量判断胜负
过程中的进度通过进度条展示
两个直播间都需要进行进度条更新

PK场景下：所有送礼都会触发两个房间的状态变化

人数多/PK激烈 → 消息量暴涨

1. 限频，每2秒更新一次状态
2. 结束后，补发一条最终结果

==========

可以屏蔽掉绝大多数消息，且用户体验不会降低

直播全量:稳定性优化

消息降级

带宽
服务器带宽是直播长链接的瓶颈

客户端
客户端消息接收量也有处理上限

消息分级
==========
P0: 系统消息
P1: 大礼物消息
P2: 小礼物消息
P3:聊天或活动状态

房间大小的限频策略
==========
Level1: [0, a0]
Level2:[a0, a1]
Level3:[a1,a2]
Level4:[a2, ~]
==========
退避策略防止频繁更新频率

直播全量:稳定性优化

关键：控制消息量

礼物连击

活动消息

状态更新

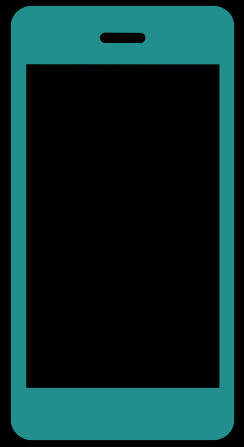
突发流量

直播运营:H5长链接

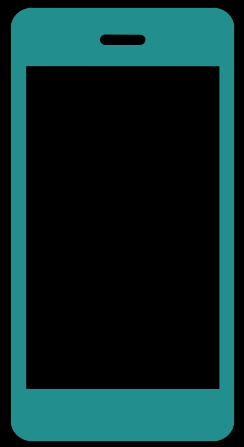
H5活动页面客户端转发

活动

长链接

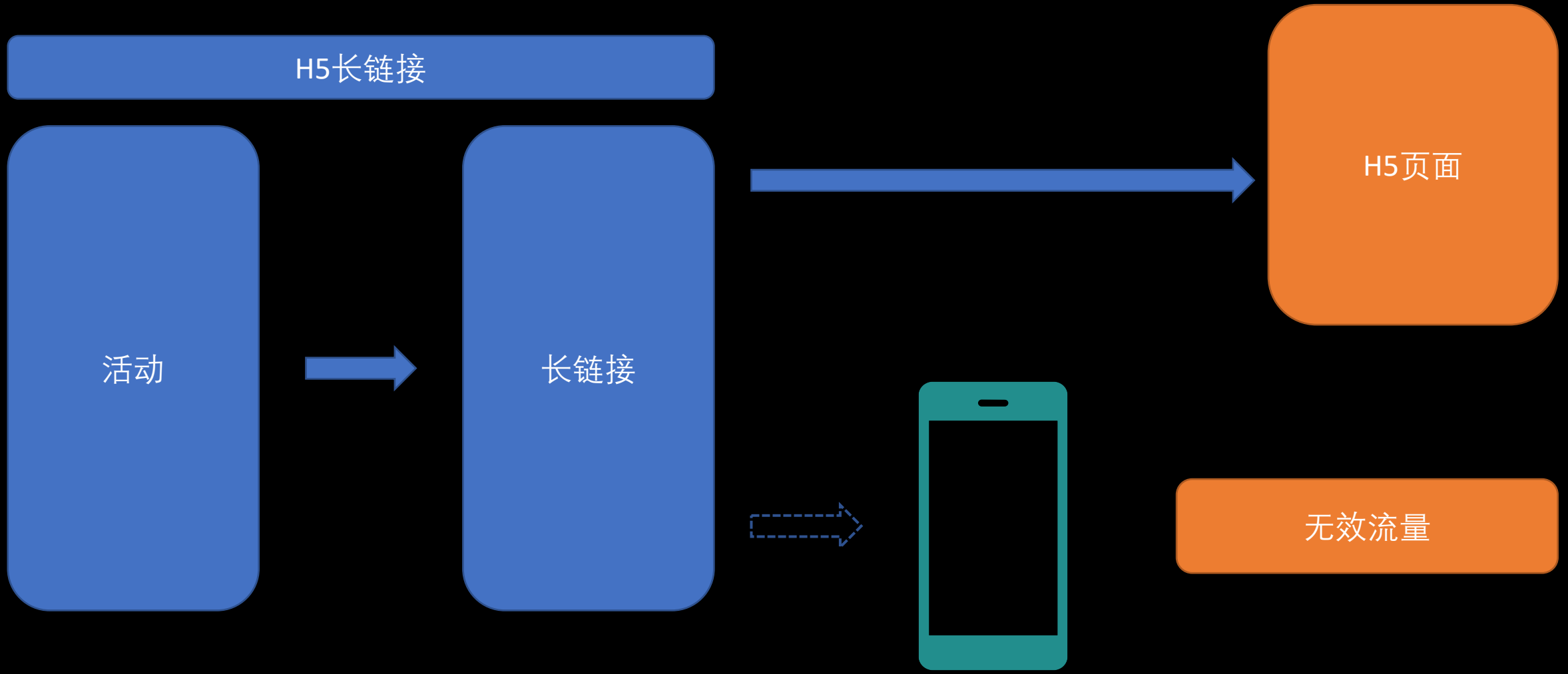


H5页面



无效流量

直播运营:H5长链接



直播运营:H5长链接

无效流量

独立部署ConnectorH5

Router添加H5相关RPC调用

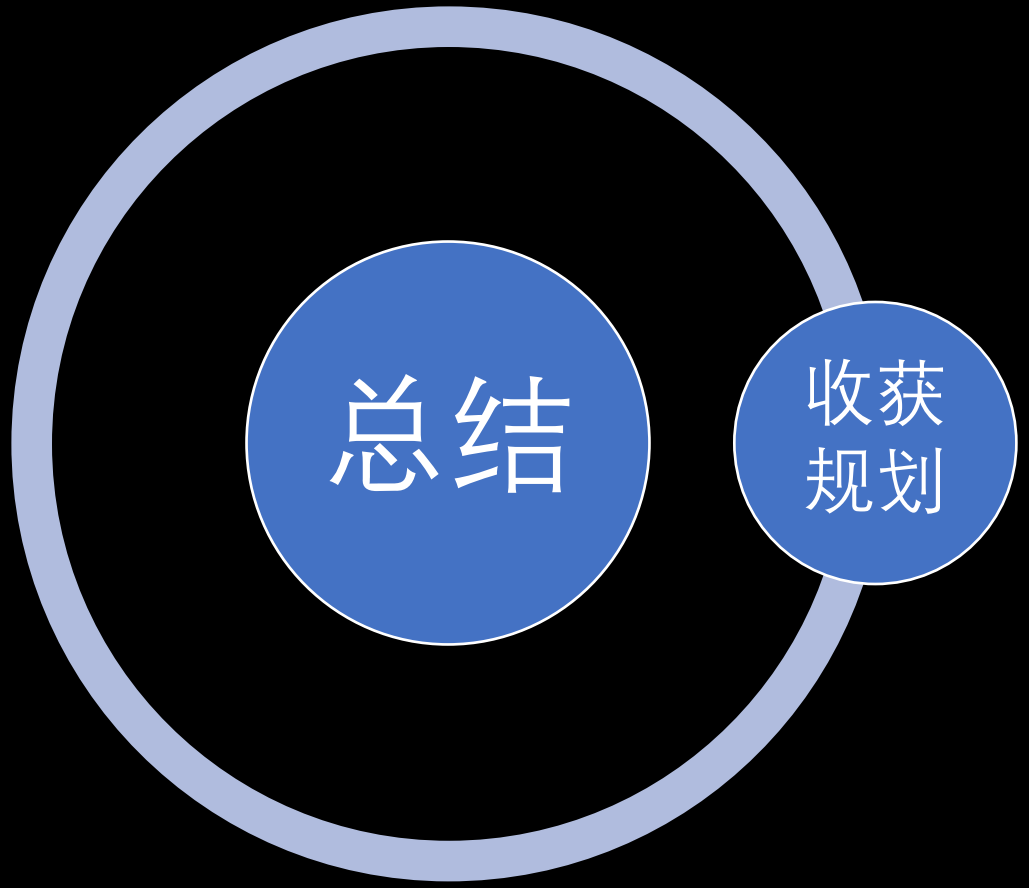
Client支持一个用户多个链接

客户端主业务消息量减少

链接解耦，服务更加稳定性

演进过程总结

直播立项	实现速度	复用已有代码，快速实现
直播公测	代码协议耦合	解耦：独立代码/协议/部署
直播全量	稳定性	控制消息量：礼物连击/状态更新/消息限频
直播运营	稳定性	链接解耦：独立H5长连接资源


总结
收获
规划

收获

设计	符合当前需求，从演进的角度去设计，做好扩展
稳定性	基础服务的核心：稳定性 抓住问题重点，提前发现系统薄弱环节
容量	服务容量要有把控：连接数/带宽/水位/...

规划

平台化	消息管理与资源管理 支持更多的业务，开放长链接能力
工具链	路由/抓包/回放/...
测试	压测常规化 业务测试自动化

GOPHER CHINA 2020

中国 上海 / 2020-11.21-22

Thanks