



# 大规模、高并发实时通信平台的 挑战思路



# 实时通信能干什么

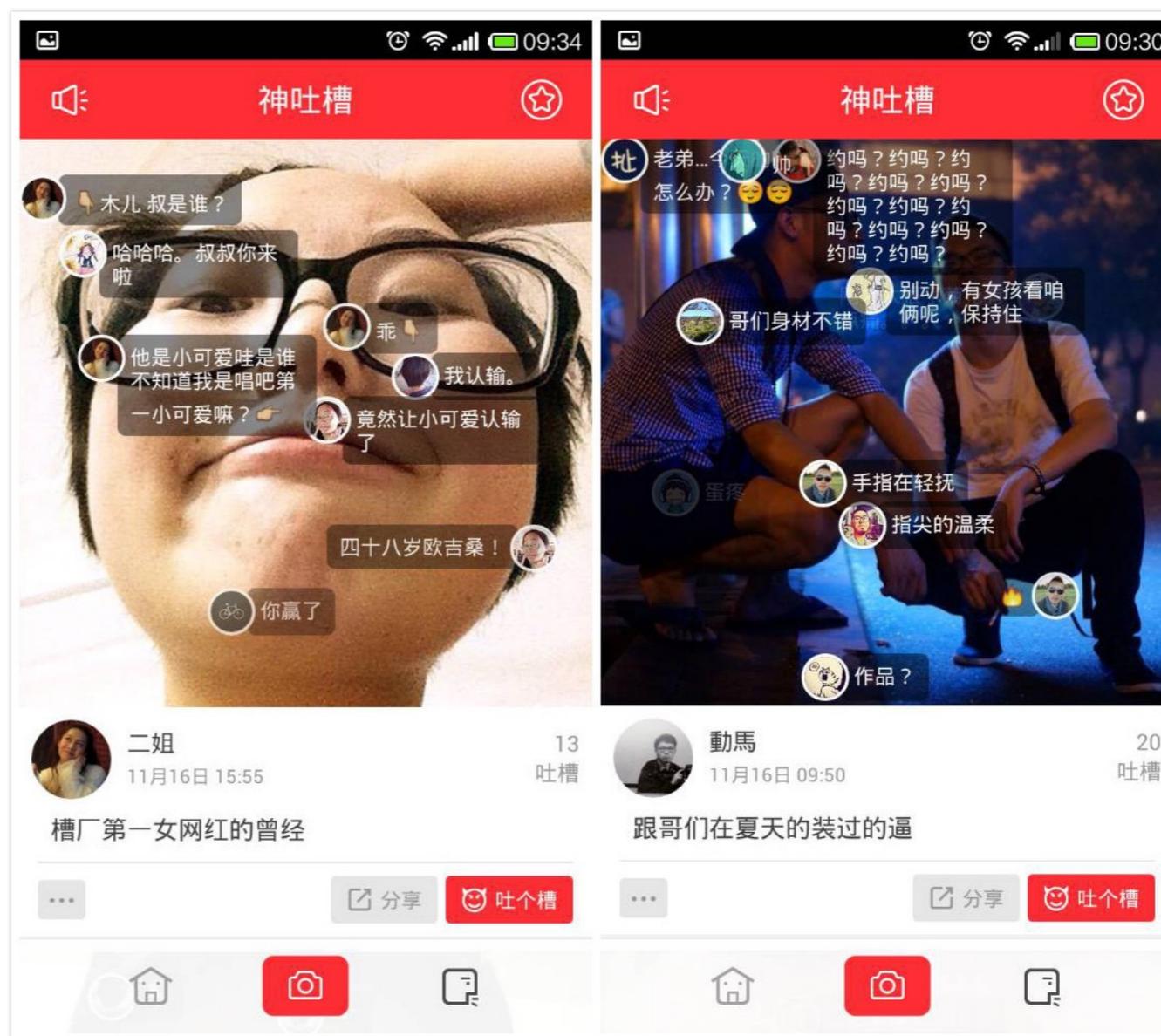
- 聊天、私信





# 实时通信能干什么

- 弹幕、抽奖





# 实时通信能干什么

- 互动游戏





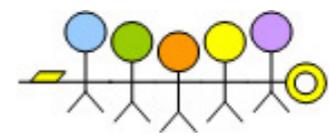
# 实时通信能干什么

- 协同编辑



# 一条「消息」的旅程

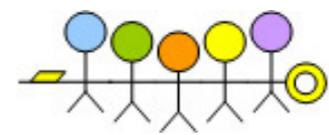




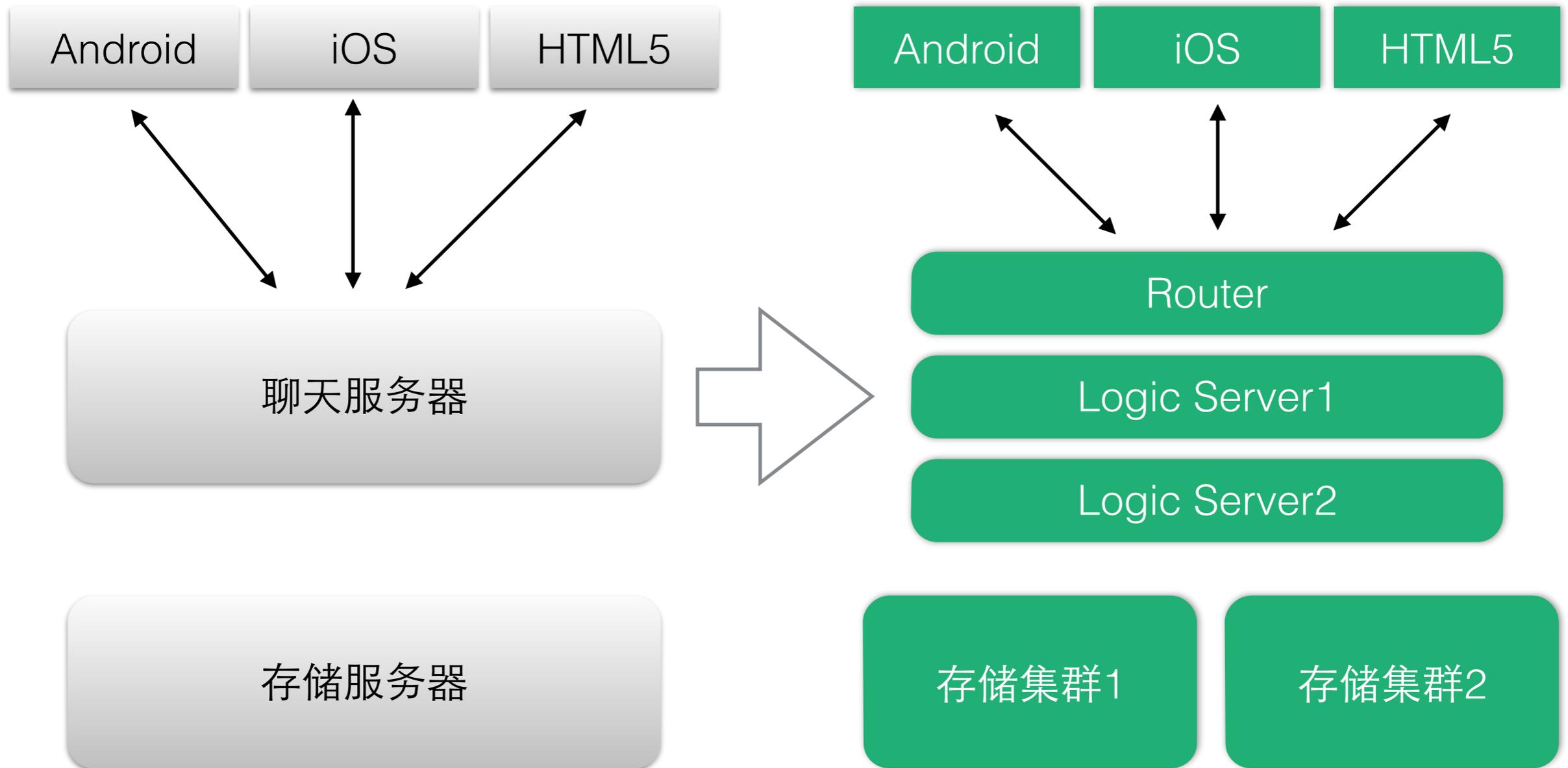


# 要解决的问题

- 高并发——同时支持上百万连接
- 实时 ——聊天还能磨磨唧唧吗？
- 扩展性——流量成倍增长的时候能简单加机器？
- 安全性——隔墙有耳、有人乱入，怎么办。。。。
- 手段多样性——对方离线了怎么办。。。。
- 使用便利性——我要定制，在各个环节定制。。。。



# 服务端架构变化





# LeanMessage 服务端架构

- 功能模块





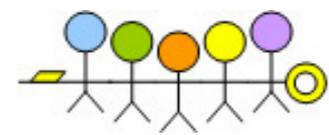
# LeanMessage 服务端架构

- 微服务 (Microservice)
  - 细粒度，针对功能和性质拆分
  - 传输：消息队列和 RPC
  - 元编程：实现本地和远程调用 API 一致
  - 服务作为一层抽象独立于进程



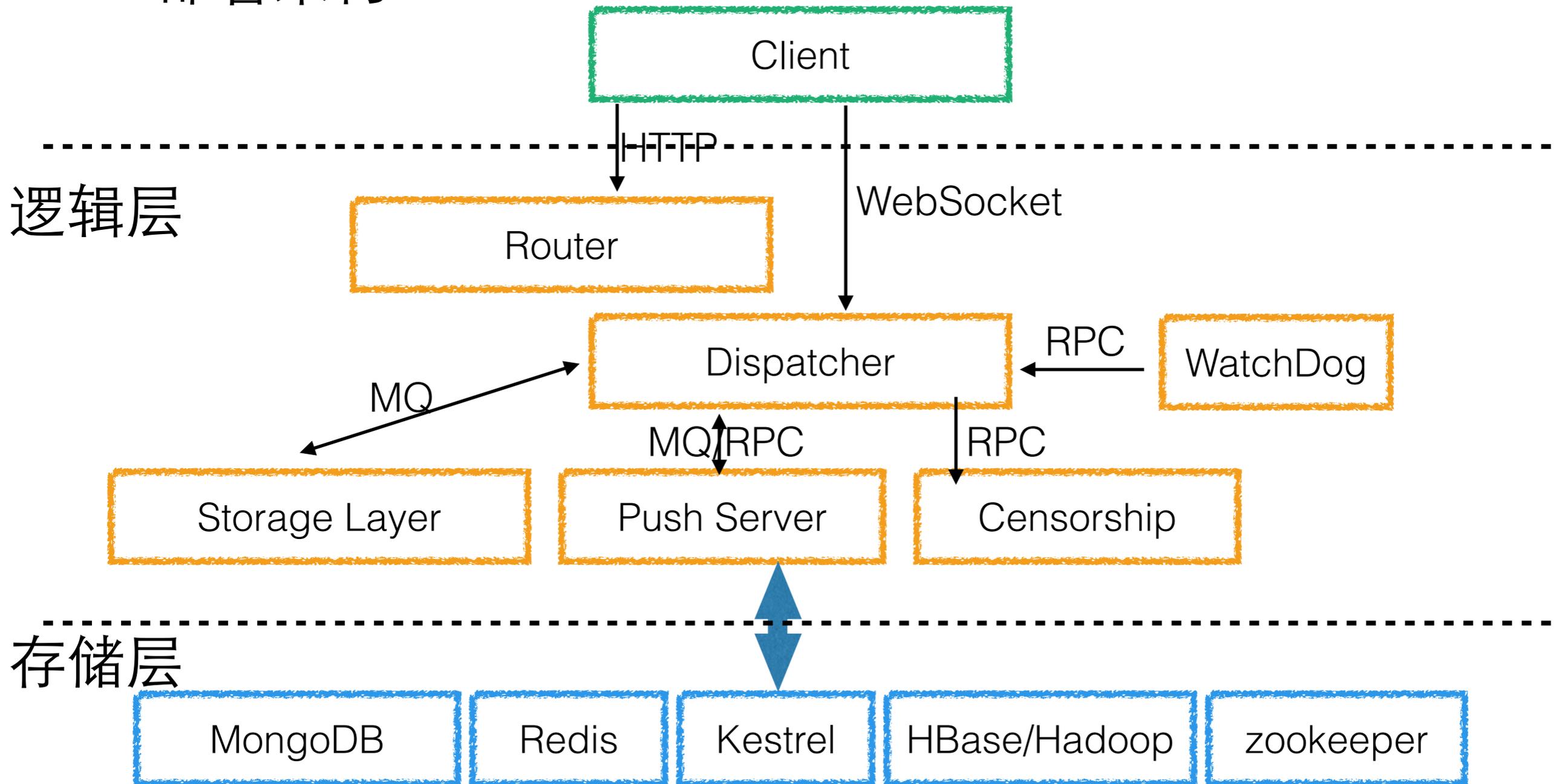
# LeanMessage 服务端架构

- 技术栈
  - Clojure
  - Netty NIO
  - core.async, SEDA pipeline
  - Slacker cluster RPC
  - zookeeper
  - Kestrel
  - MongoDB
  - Redis
  - HBase



# LeanMessage 服务端架构

- 部署架构





# LeanMessage 服务端架构

- 服务发现
  - 利用 zookeeper 注册服务
  - ephemeral 节点
  - watch
    - 自动 failover
    - 自动引导流量到新的部署
- 动态配置管理



# LeanMessage 服务端架构

- 横向扩展和负载均衡
  - 相应服务层面增加节点即可扩容
  - 应用层的负载均衡
    - DNS 负载均衡不够灵活
    - REST 无状态服务
  - 根据长连接服务器的负载重新分配新连接
    - 支持其他业务规则：灰度发布
- 自动更新可用列表

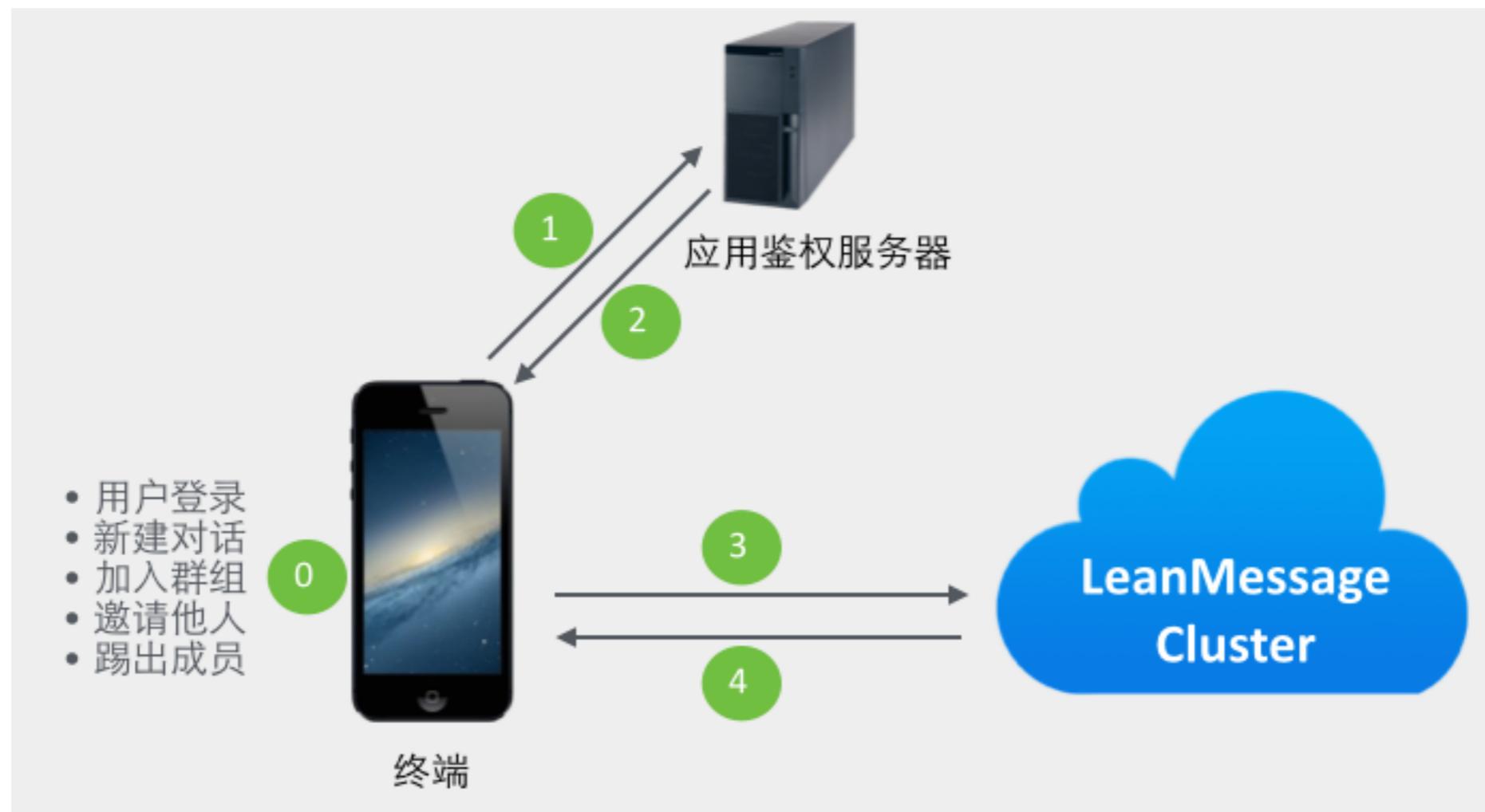


# LeanMessage 服务端架构

- 报表、监控
  - 统一的系统级（网络、硬件、OS）监控
  - 服务内置监控模块，实时汇报业务数据
  - 流量异常报警

# 通道安全

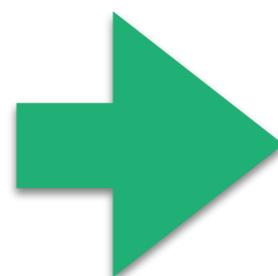
- HTTPS
- 第三方鉴权





# 通用接口满足多种需求

- 单聊
- 群聊
- 开放聊天室



- 对话
  - 普通对话
  - 暂态对话

- 消息
  - 普通消息
  - 暂态消息



# 定制化开发 – API hook

- messageReceived
- receiversOffline



# 传输协议

- XMPP
- 自己造轮子



# 持久连接的客户端

- Android
  - 后台运行 service
- iOS / Windows Phone
  - 应用在前台时建立连接
  - 被切换到后台时主动断开连接，改为推送通知
- HTML5
  - websocket 或者降级方案



# 下一步

- 把连接管理和消息路由分离
- 二进制协议
- SDK 中增加 ui 库