Instagram Direct
高性能聊天产品

李晨 @ Instagram Direct

关于我

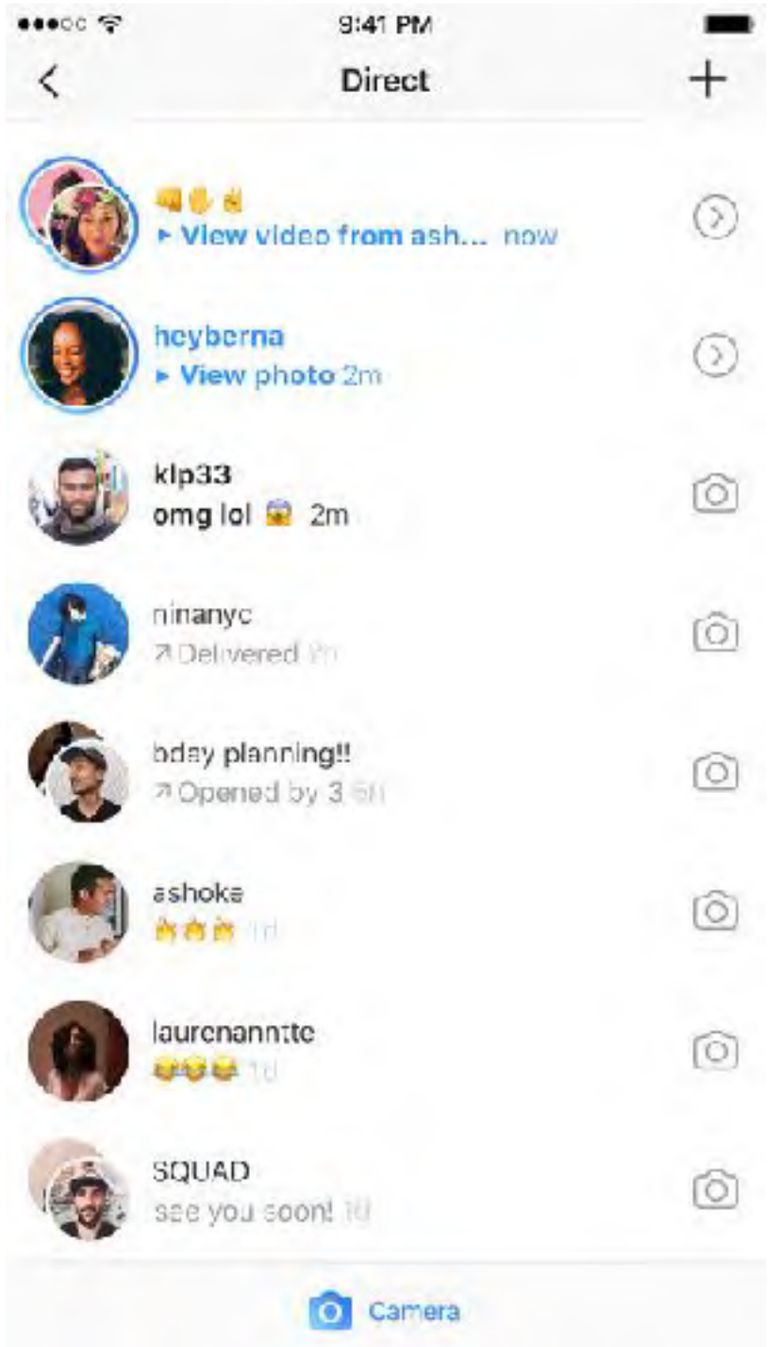
- 2009-2010：密歇根理工大学，iPhone开发俱乐部
- 2010-2015：Apple，苹果商店App
- 2015-现在：Instagram，Direct

Instagram

- 图片视频社区；明星、网红
- 年轻人为主
- 7亿月活，80%在美国以外地区
- 信息流、直播、Story、聊天

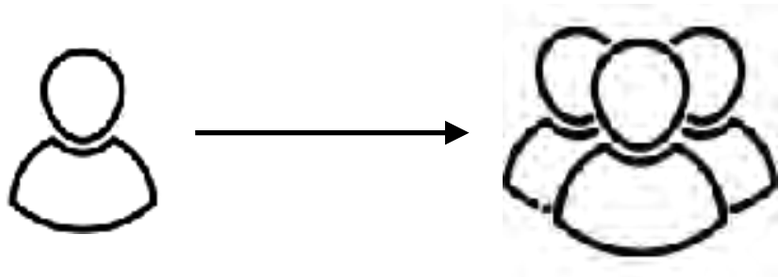
Direct

- Instagram旗下的聊天产品
- 密友、年轻人
- 3.75亿月活跃用户



挑战

- 用户增长：2年，+4亿
- 功能增加
- 团队扩大



- 负责众多功能的单平台团队 —> 专注单个功能的跨平台团队

第一部分：前端架构

- 响应速度：
  滑动时卡一点，日活降0.5% - 2%

- 代码可读性、可维护性：
  团队扩大，新人加入，怎么确保少bug

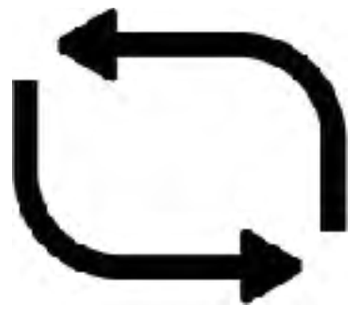
- 功能易扩展：
  - Instagram一周一发布
  - 应对产品经理加功能

前端架构：
性能、模块化、信息流

- 性能：
  纯Native开发

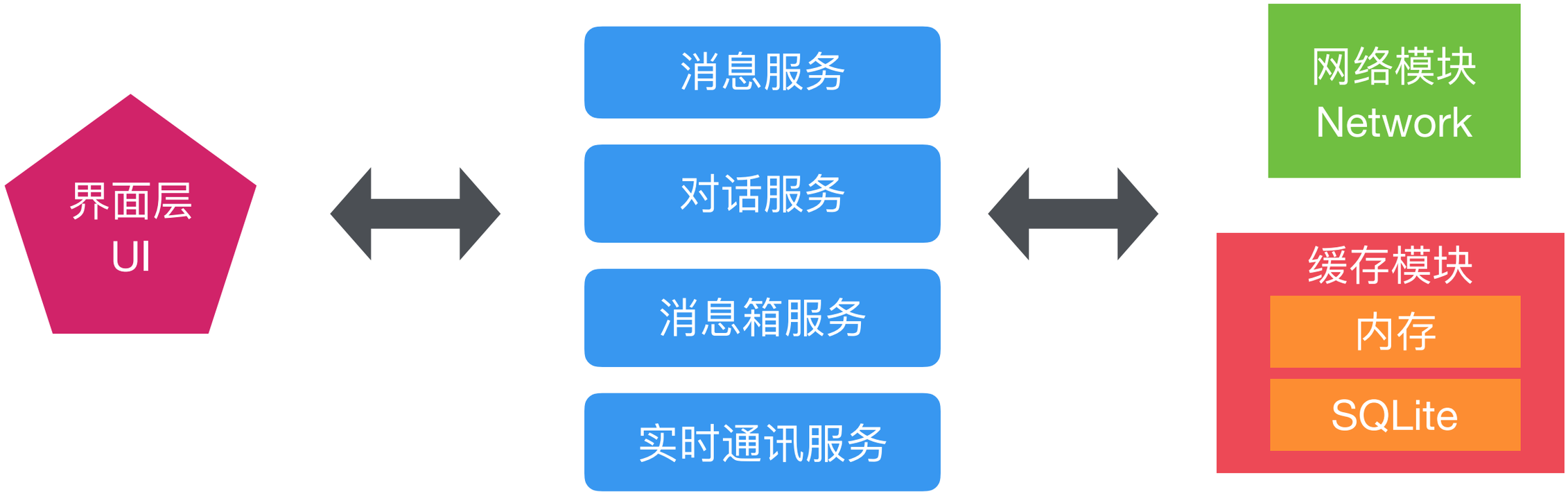
- 模块化：
  - 单个class不能太大
  - 代码清晰易读

- 信息流：
  - AB测试
  - 便于调试
  - 易于增加功能



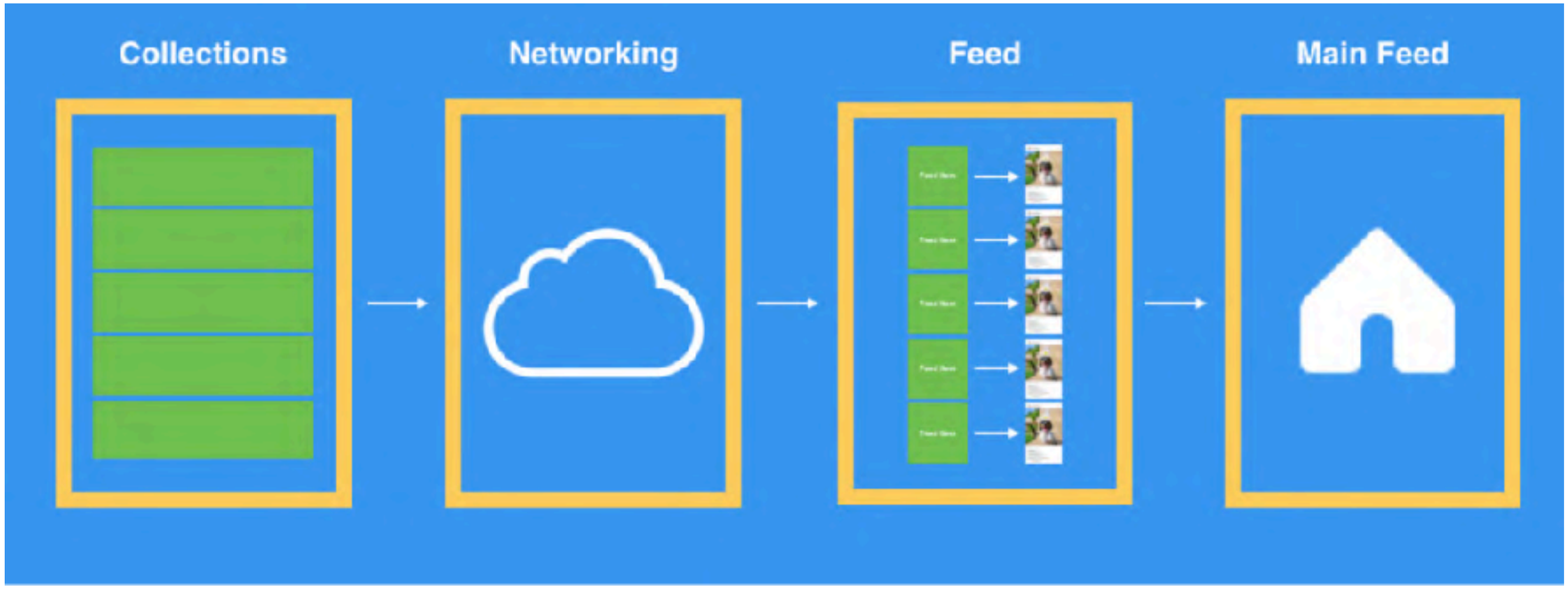
前端架构：信息流

前端架构：信息流



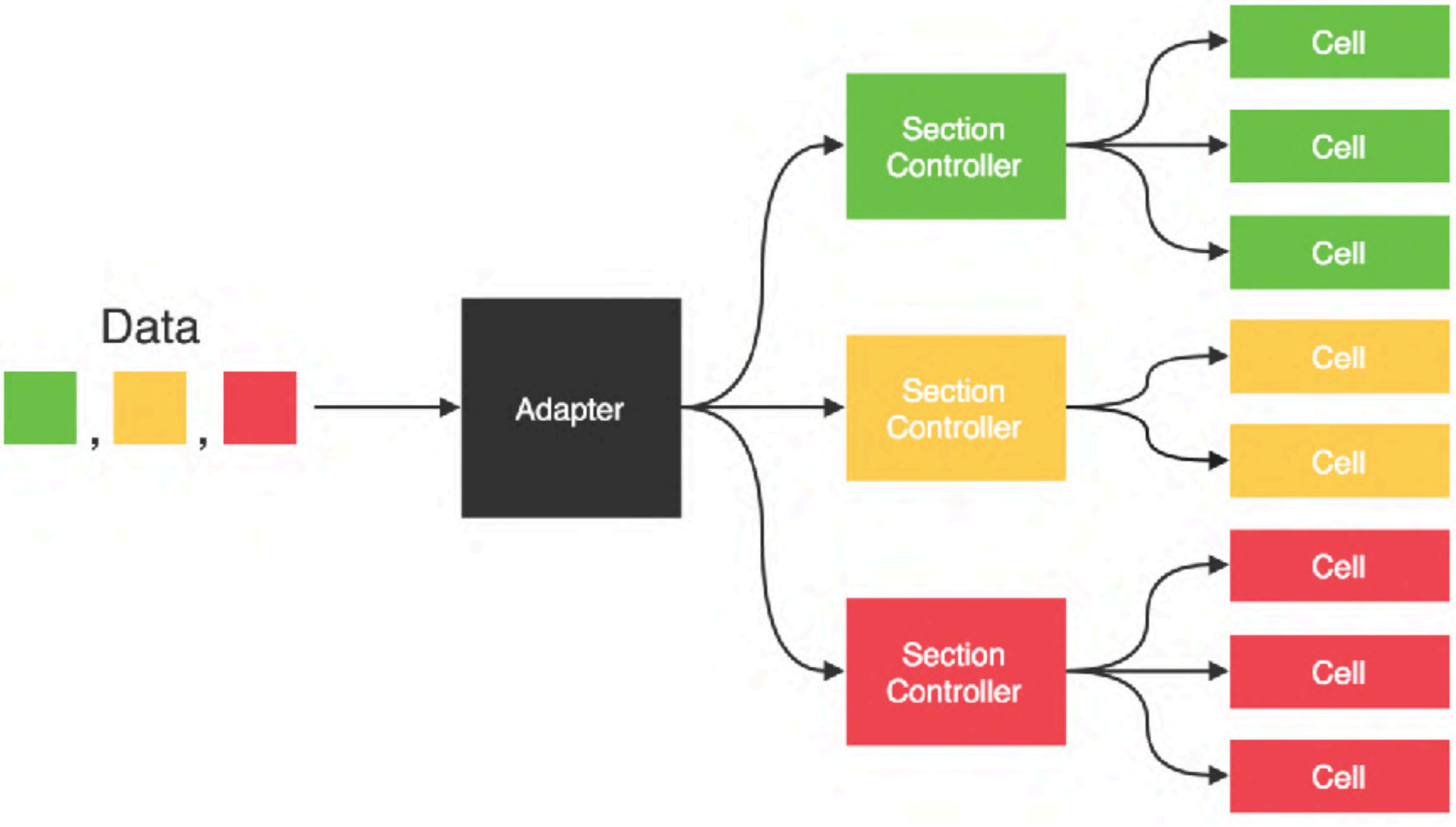
前端架构：长列表优化

- 常见问题：长列表代码过于臃肿



前端架构： IGListKit

- 防止单个Class过于臃肿



前端架构：MVVS

- Model：消息模型，包括文字、图片、视频、信息分享等
- View Model：每一个消息的尺寸、颜色、显示方式等
- View：具体显示消息内容，主要是cells
- Section Controller：负责联络一个消息类型相应的MVV

前端架构：示例

Model

好友：“周末去GMTC大会，约么？”

我：“约”

好友：❤️

View Model

“Today at 4:55 AM”,
中间对齐，{120, 40}

好友，“周末去GMTC大会，约么？”,
左边对齐，{240, 50}

我，“约”,
右边对齐，{50, 50}

好友，❤️,
左边对齐，{60, 50}

前端架构：示例

View Model

“Today at 4:55 AM”,
中间对齐，{120, 40}

好友，“周末去GMTC大会，约么？”,
左边对齐，{240, 50}

我，“约”,
右边对齐，{50, 50}

好友，❤️,
左边对齐，{60, 50}

Section Controller

DateSectionController

TextSectionController

TextSectionController

LikeSectionController

前端架构： 示例

Section Controller

DateSectionController

TextSectionController

TextSectionController

LikeSectionController

View / Cell

前端架构：优势

- 每个模块的职责明确，不会过于臃肿
- 每一条新消息进入，只需要增加一个Section Controller，更新一个Cell，刷新效率和响应速度高
- 添加一个新消息类型，新增一套MVVS，不用改动其它代码
- 团队暴兵，功能迭代

第二部分：消息重试机制

- 聊天产品的设计需求
  - 消息传送成功率
  - 消息发送速度
  - 用户感知到的消息发送速度
  - 确保消息顺序




消息机制：发送失败的原因

- 随机性的失败
  - 服务器短暂宕机
  - 信号不稳定
  - 无网络信号

- 解决方案：自动重试

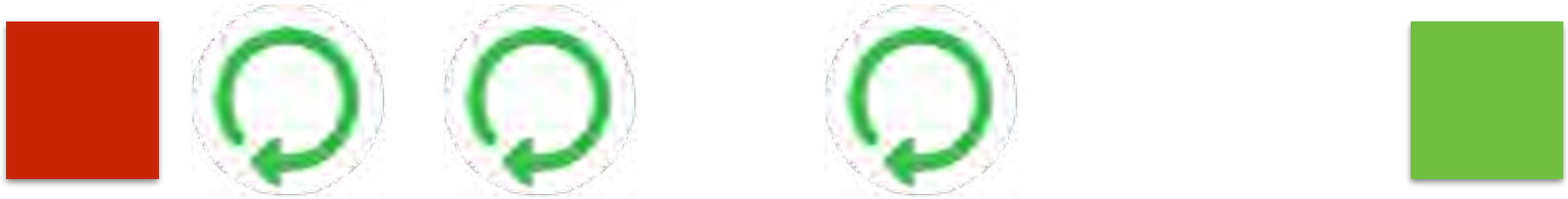
消息机制：自动重试

- 针对随机性消息传输失败
- 用户不会感知到自动重试的存在
- 每一次重试都能增加消息发送成功率

  (1-90%) * (1-90%) = (1-99%)

消息机制：单个消息重试

- Exponential Backoff:
  利用时间多样性，最多重试5次



- 发送过期机制（Timeout for HTTP / Websocket）
  图片和视频：60秒；其它消息：10秒

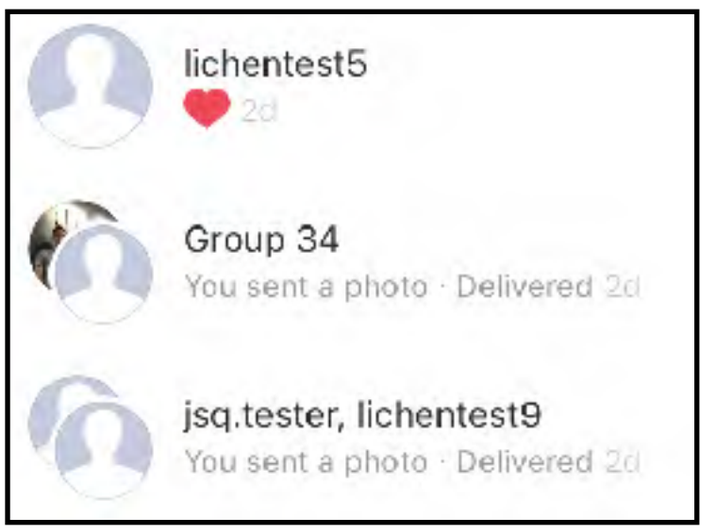
消息机制：队列

- 自动重试有可能导致消息顺序的混乱
  1,2,3,4 -> 1,2,4,3

- 保存顺序：将所有消息放入队列当中；只有当早先的消息发送最终完成后，才开始发送稍后的消息

消息机制：多队列

- 消息序列只对于单个对话有意义，因此不要将所有消息放入一个序列中
- 每个对话有一个消息队列
- 图片和视频的发送速度比顺序重要，将它们统一放入一个多媒体队列当中



队列一

队列二

队列三

多媒体队列

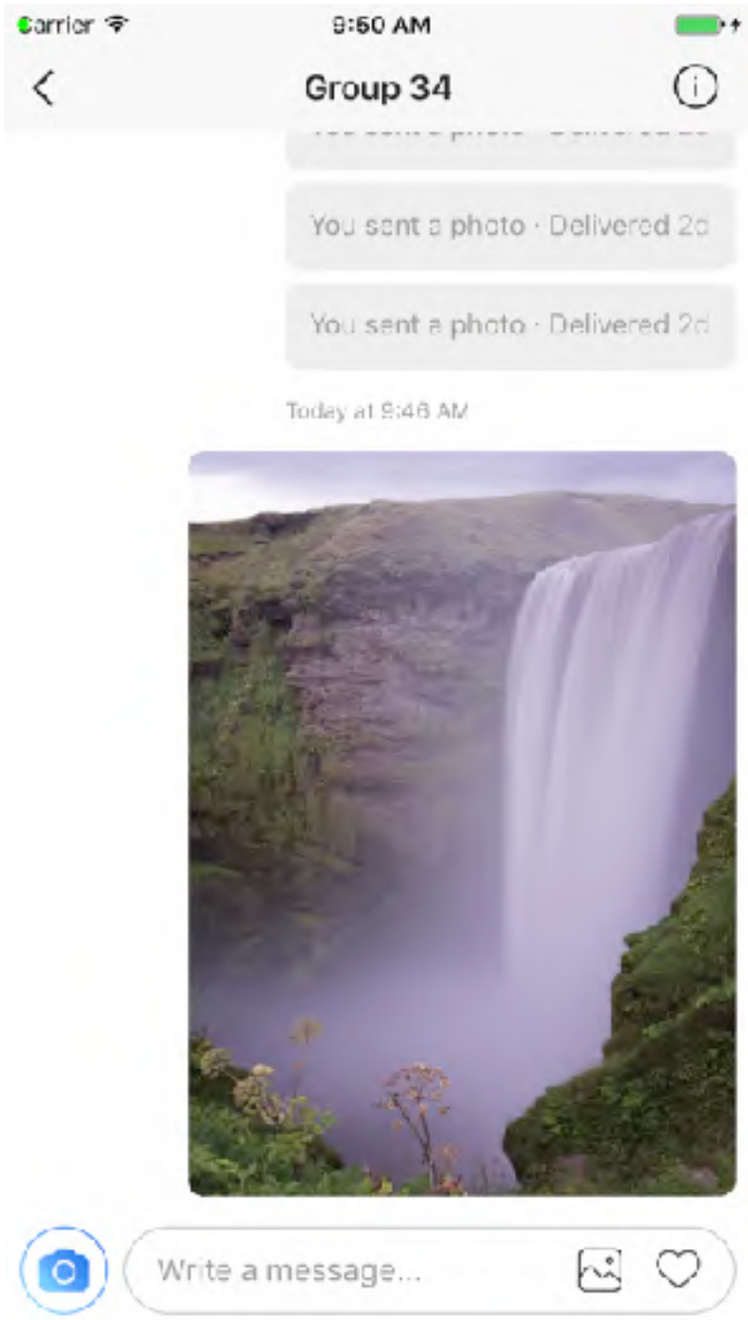
消息机制：自动重试时机

- 好的时机
  - 网络信号好
  - 服务器正常
- 糟糕的时机
  - App刚刚冷启动，CPU资源占用率高
  - 飞行模式

- 不要为确定性的失败而重试
  - 视频编码失败（客户端bug）
  - 授权错误

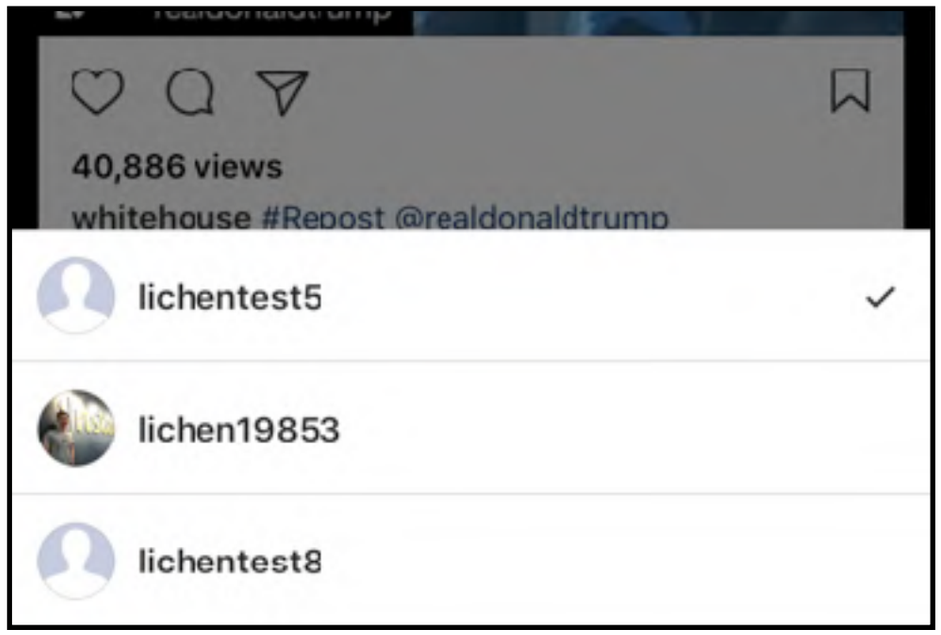
消息机制：图片和视频

- 发送图片和视频
  第一步：压缩编码
  第二步：发送至服务器
- 第一步无需重试，只重试第二步
- 最多同时传送3个多媒体消息，防止占用过多系统资源



消息机制：多账号系统

- 以下情况暂停消息发送：
  - 切换账号
  - 用户登出
- 当App重新启动或登录时，恢复消息发送



Q&A

- 联系方式：lichen@fb.com

Thanks!