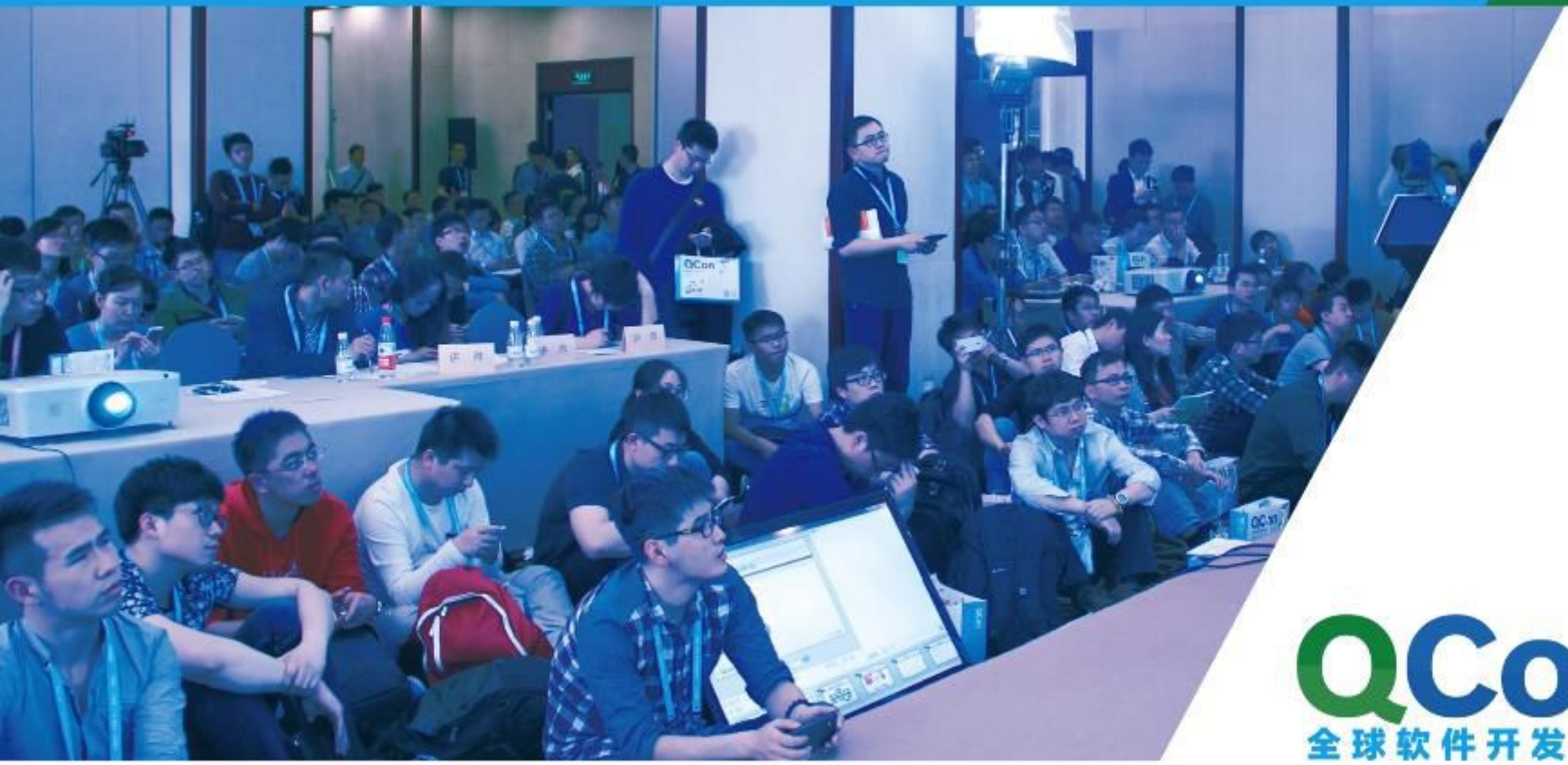
QCon全球软件开发大会

International Software Development Conference



QCon
全球软件开发大会

IM通讯云技术路线的选择

——移动时代的即时通讯系统架构的考虑要素

容联云通讯 许志强

移动IM的极致追求

快

登录快
发送消息快

1. 无DNS设计
2. 小包体协议
3. 后台轮询测速
4. 精简认证重连
5. 多媒体消息通道复用
6. 长短连接并用

省

省电
省流量

1. 低流量协议
2. 压缩机制
3. 高频词编码
4. 智能多包合并
5. 自适应、最小心跳包技术

稳

不丢消息
99.99%的稳定性

1. 多段ACK确认
2. 永久化存储
3. 排序队列控制
4. 故障自动迁移
5. 负载均衡、无单点故障

要解决的问题

消息的可靠（不丢、不重复）

消息的有序

多终端登录

云平台的大并发

多数据中心容灾

协议选择

传统的IM协议：

XMPP 、SIP SIMPLE 等

优点：

开源的框架很多(Openfire、Tigase, Ejabberd)

缺点：

交换复杂、流量大，不可靠

不适合移动互联网

演进路线：

在传统的协议上改进

另起炉灶、高效的编解码协议（protocol buffer等）加自定义上层协议

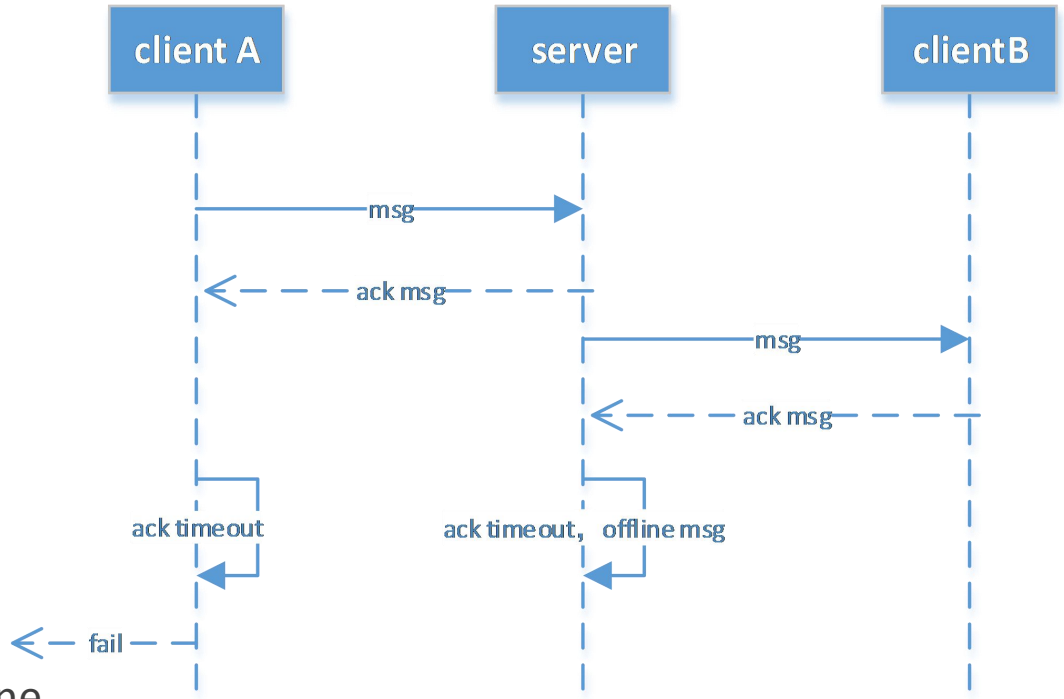
消息可靠

UDP 不可靠，TCP 可靠？

发送消息的选择
- 最多一次 At most one
- 最少一次 At least one
- 有且仅有一次 Exact one

不丢消息，不重复不可兼得？
- SMC 定理

通常的解决方案
- 增加ACK确认、at least one
- 增加msgid、客户端剔重



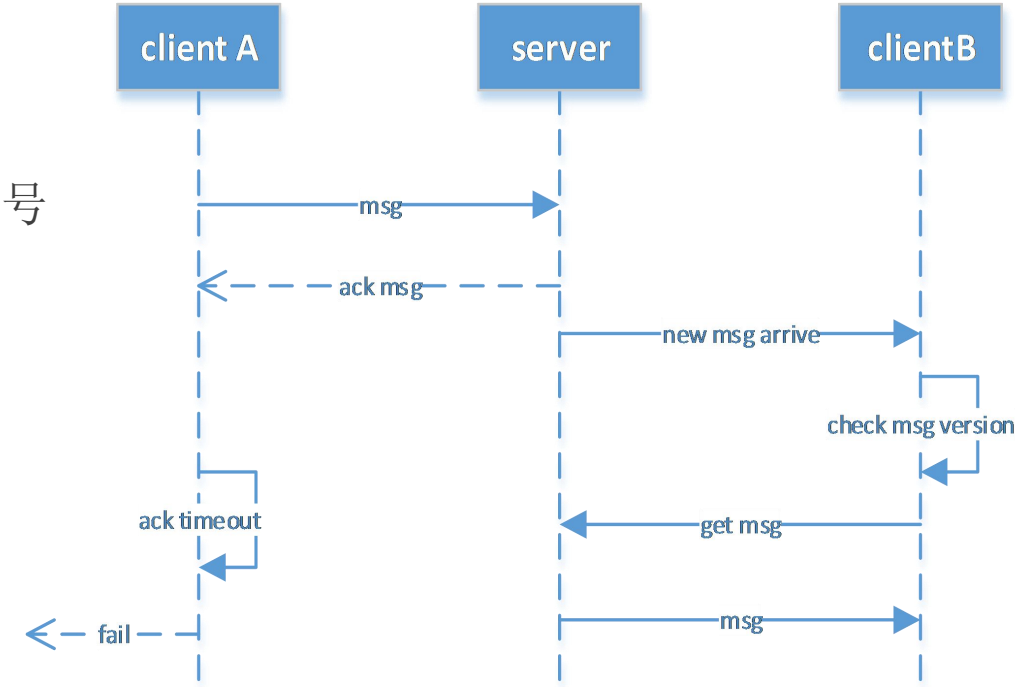
消息可靠

优化的解决方案：

- 发送端通过msgid剔重
- 每个用户消息设置自增版本号
- push pull 相结合
- 投递不等ack，无需retry

极端情况不丢消息：

- ACK 之前永久化存储

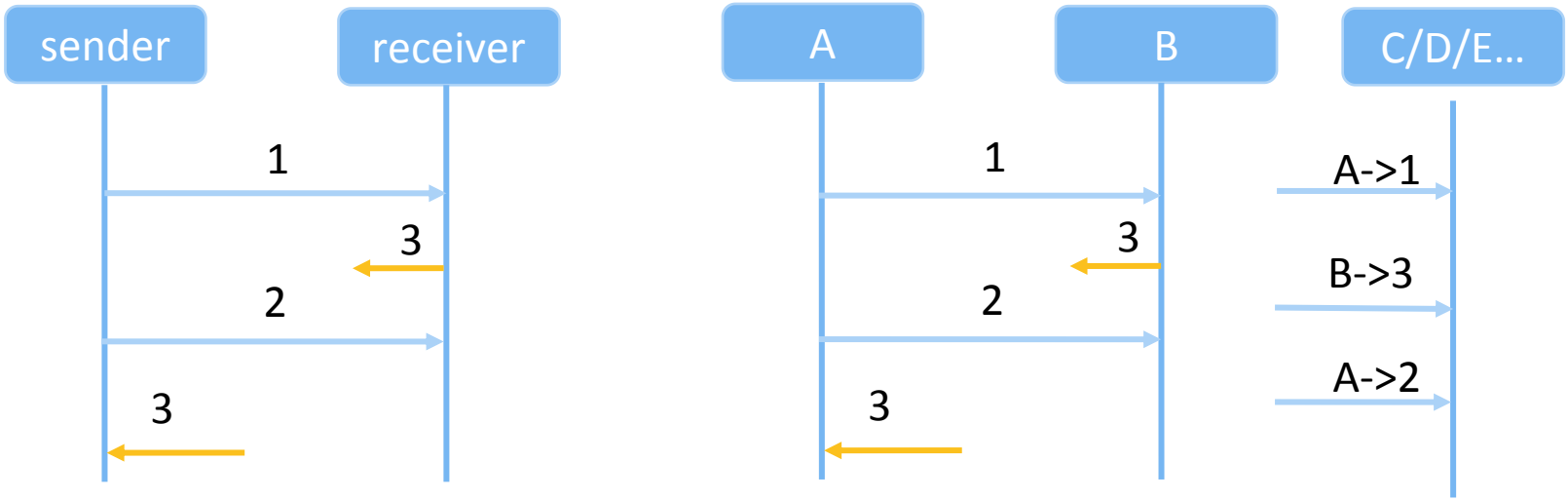


消息有序

消息的有序：

点对点聊天---- 基于发送者的序列有序

群组聊天---- 对话的因果有序



消息有序

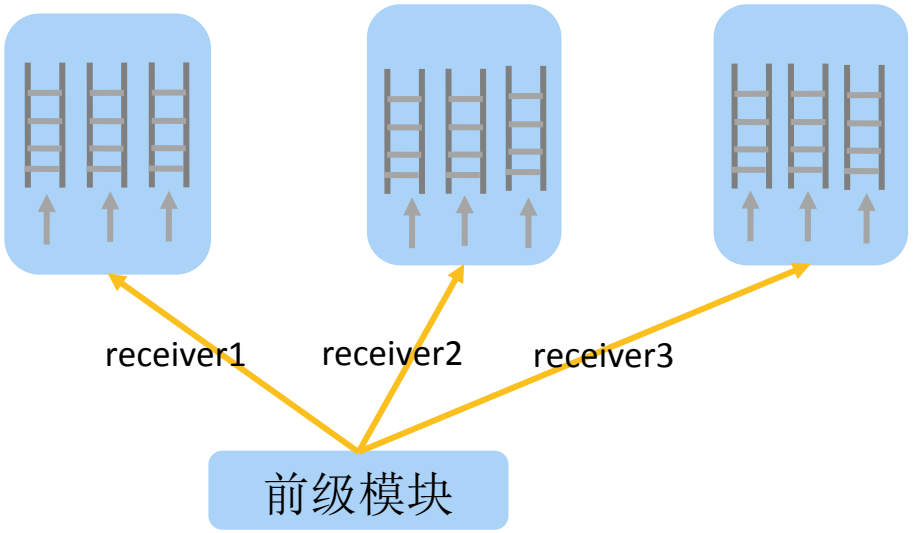
解决方案：

通过单个tcp的长连接保证客户端发送有序

对于语音、图片等异步发送的消息采用“预占坑”

模块内部采用接收者hash特定线程处理

模块之间采用接收者hash到特定的模块



后级模块集群

多终端登录

终端类型

手机、PC、pad

多终端消息同步 （三个版本号信息）

用户消息最新版本号

用户已读最后版本号

终端本地最后读取版本号信息

同类型的终端登录，采用强制下线方式。

强制下线必须以消息方式通知、消息丢失怎么办？

高并发

高并发

IM 登录用户 ----- 用户永远在线、千万级用户在线

消息发送的高并发（特别是大群组情况）

解决方案：

设计上保证模块无状态、可水平扩展。 GLBS 、LVS ， 控制负载分配

尽可能减少交互

异步I/O ---- Netty ， Mina， libev、libevent，seastar

缓存 ---- 模块一级缓存、内存数据库二级缓存

数据库 ----- NOSQL数据库 、SQL数据库

NOSQL 数据库的选择

NOSQL 数据库种类繁多：

MongoDB、Cassandra、Redis、Hbase等

选择业务场景合适的

什么样的数据适合放NoSQL？

逻辑简单，数据量大

Redis + Cassandra

Redis 解决版本号的线性自增、数据二级缓存

Cassandra 解决 IM消息存储的线性扩展

多数据中心

多数据中心的难点

数据同步

CAP原理限制

IM 类应用更多的是可用性，非一致性。

三个或以上的数据中心多活才经济

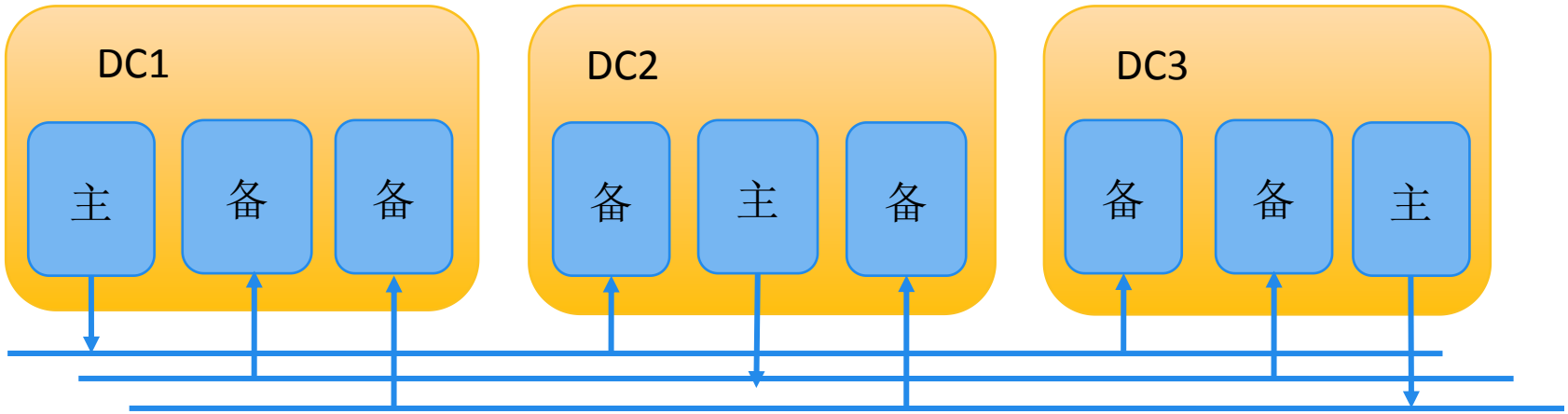
将用户接入不同的数据中心、区域自治

多数据中心

方案：

全局关系型数据采用数据分区，任何一个数据只在一个中心更新。

redis 必要数据采用异步双写。



维稳措施

灰度发布

服务监控

限流、服务降级

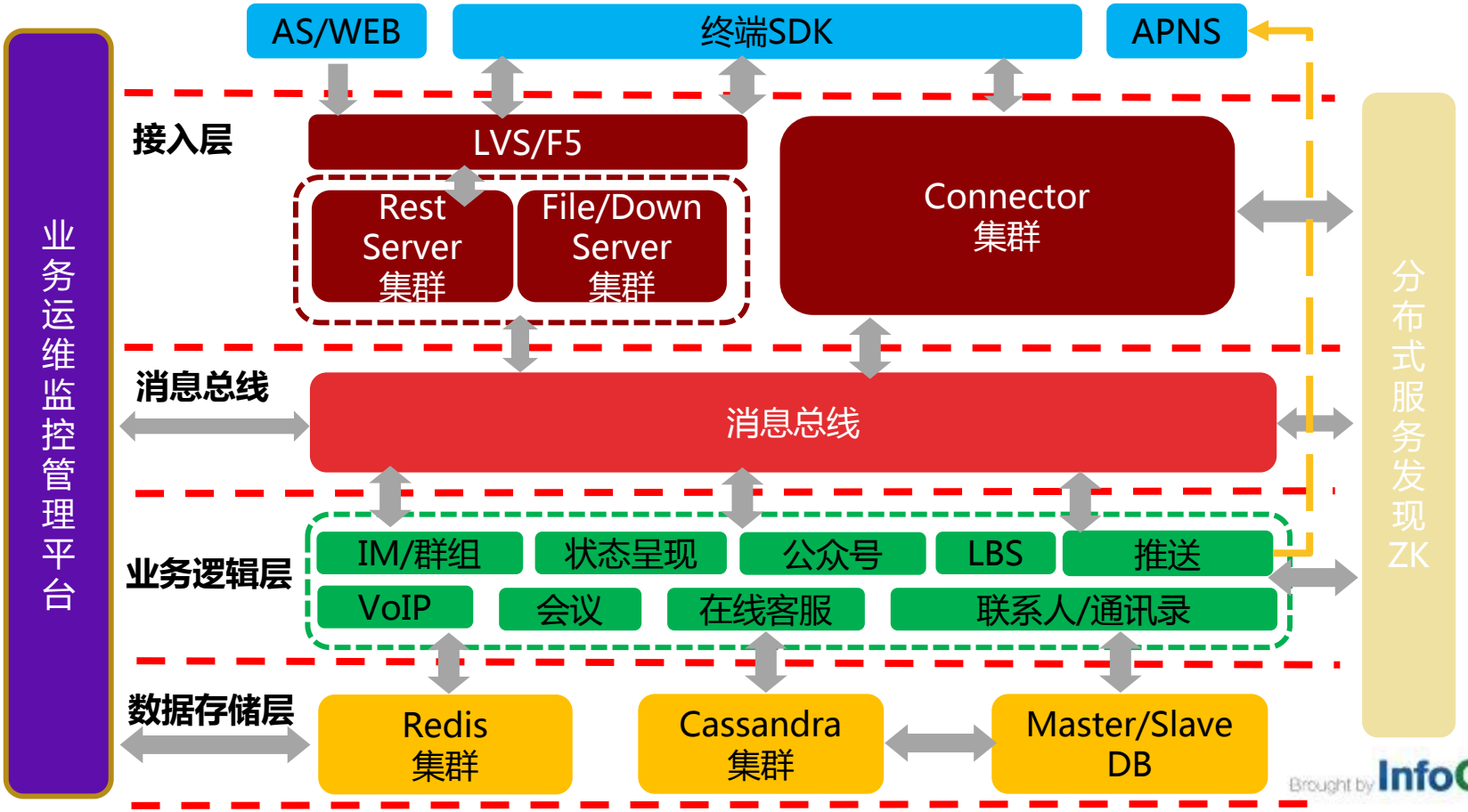
防止雪崩

故障、超负荷时，最前端丢弃流量，防止造成雪崩

通知SDK 多久时间不要连接服务器

服务降级的触发是以后端的门限指标为依据。

架构选择



IM通讯云

Q&A

THANKS