计算机系统基础

Computer System Fundamentals

想听什么？你来定义！

shidinglin@gmail.com
@林仕鼎

欢迎入群！

计算机系统课程

该微信群二维码将在2015年7月12日失效

Class 6

IM设计分享

韩兴凯 云校

讨论

- 1：IM是什么？
- 2：IM需要注意些什么？

目录

- 1：传输协议选取
- 2：传输格式选取
- 3：协议设计
- 4：架构设计

传输协议选取

- http
- Udp
- Tcp
- websocket

传输协议选取

- 和用户直连的接口，采用tcp长连接
- 内部系统采用http短连接

目录

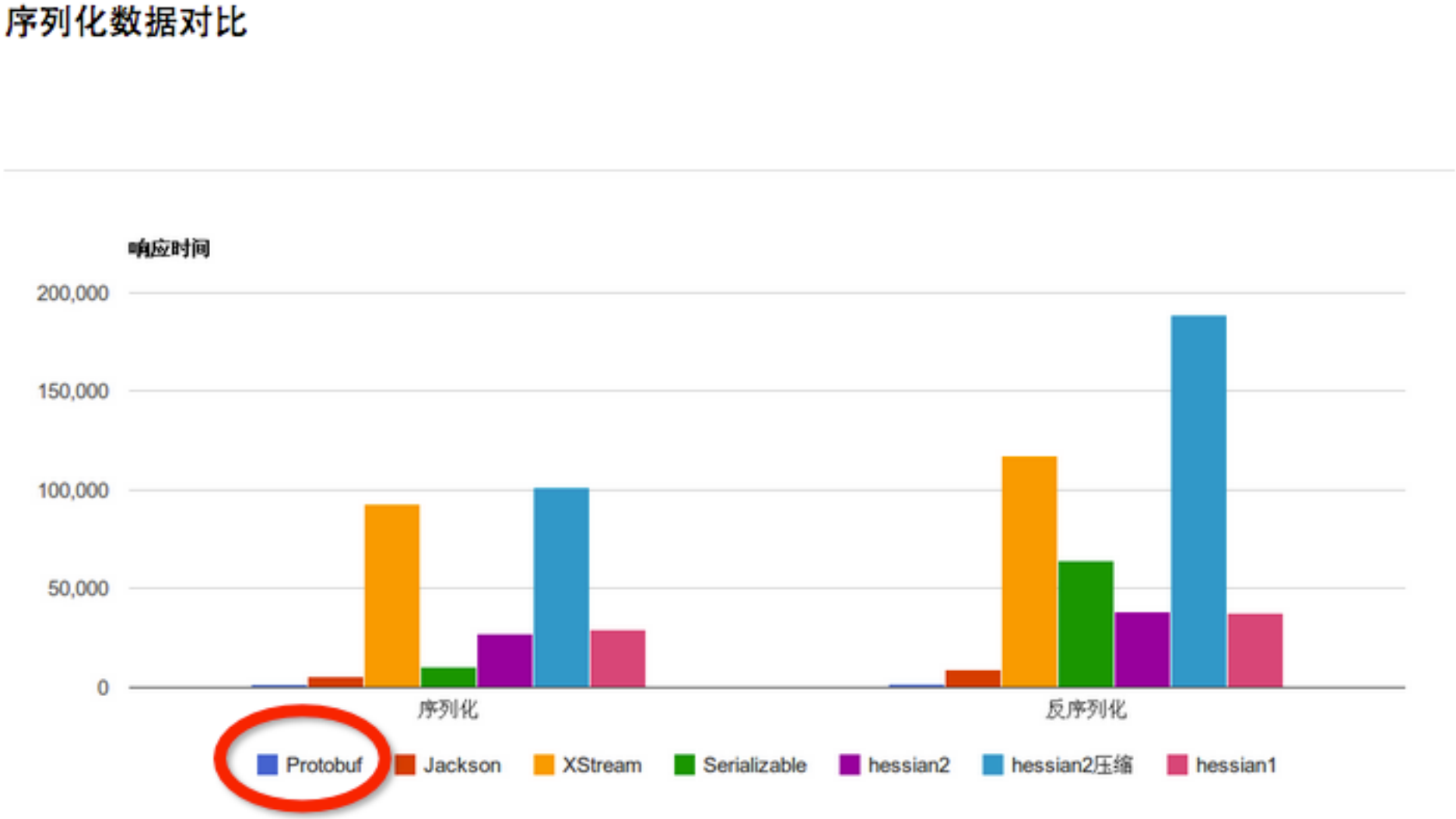
- 1：传输协议选取
- 2：传输格式选取
- 3：协议设计
- 4：架构设计

传输格式选取

传输格式选取

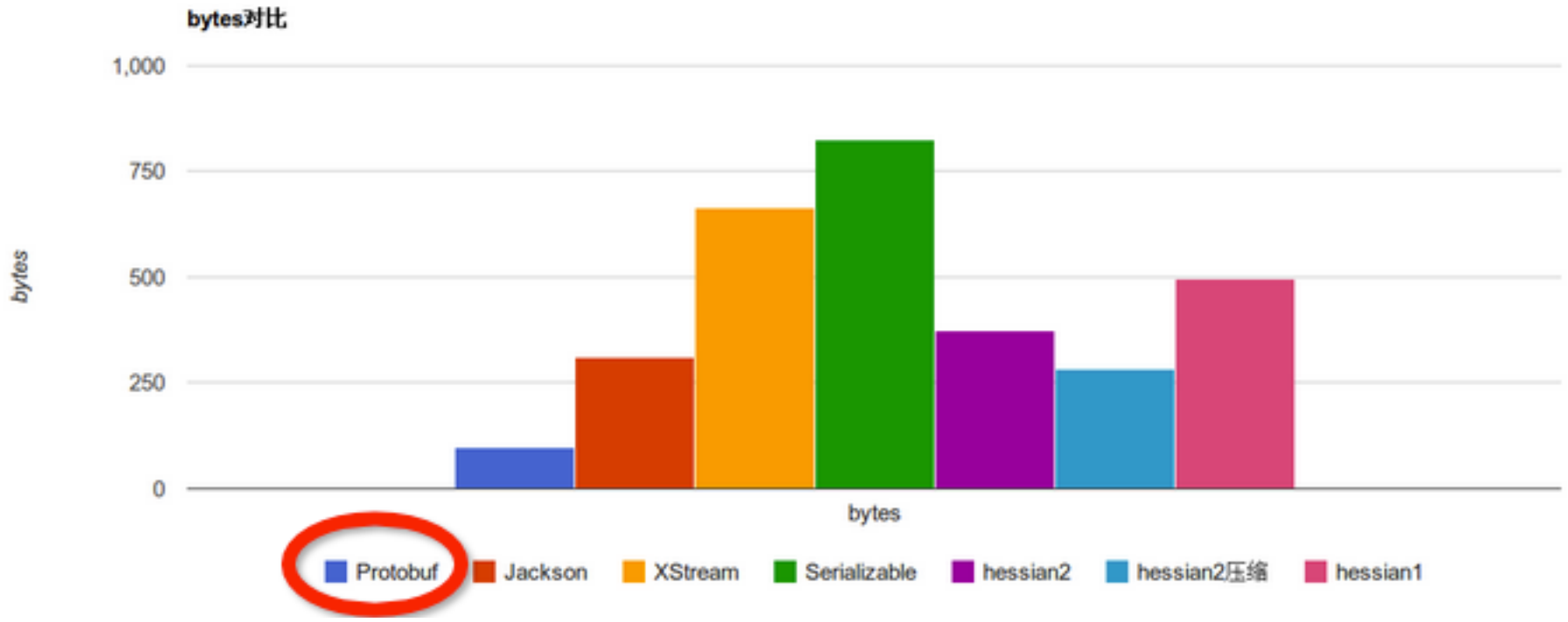
- Xml
- Json
- 对象的二进制化
- Protobuf

序列化　反序列化



Byte字节数

bytes字节数对比



protobuf

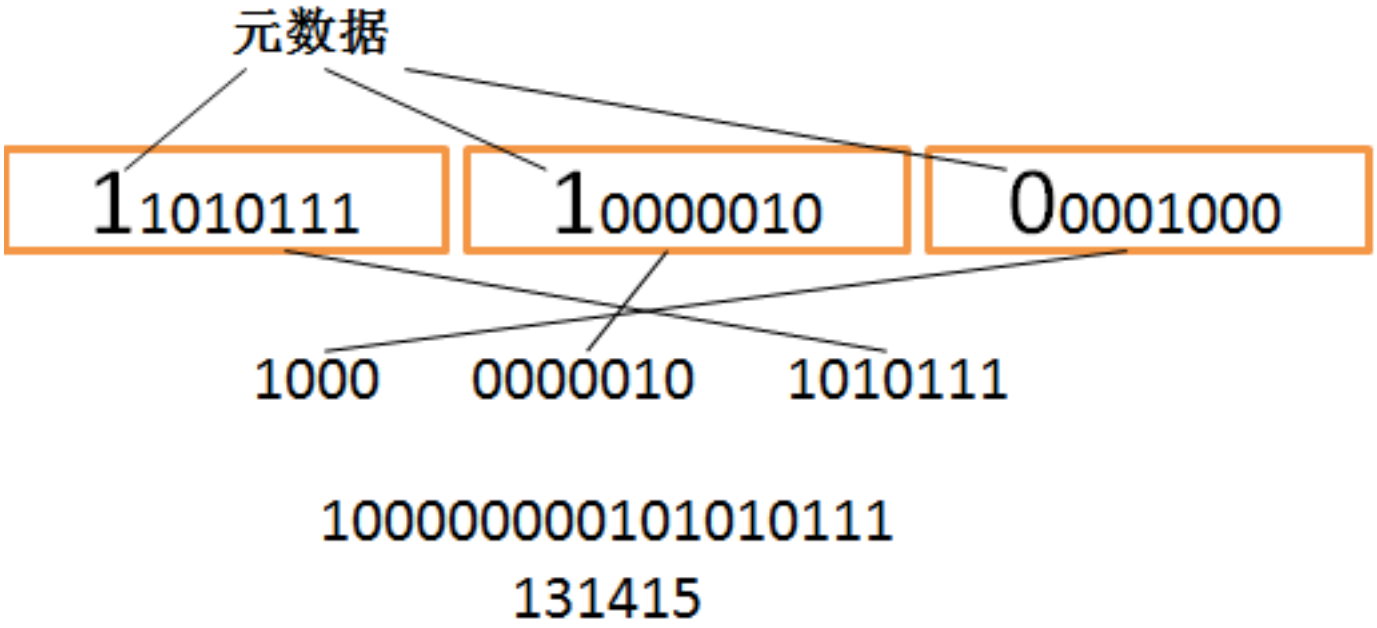
- 序列化性能高:
  - json的1/10
  - Xml的1/20
  - 对象序列化的1/10
- 字节数少，压缩效果好
- 可读性差

Protobuf编码

- Variant
  - 针对整形（int32,uint32,int64,uint64）
  - 字节最高位为1表示下一字节还有有效数据
  - 字节最高位为0表示该字节为最后一个有效数据
  - 最大需要5个字节

Protobuf编码

- variant



Protobuf编码

- Zigzag
  - 负数的存储依然会占很大空间
  - -1的存储如下：
    - 11111111 11111111 11111111 11111111
    - 如果用variant会用很大空间

Protobuf编码

- Zigzag

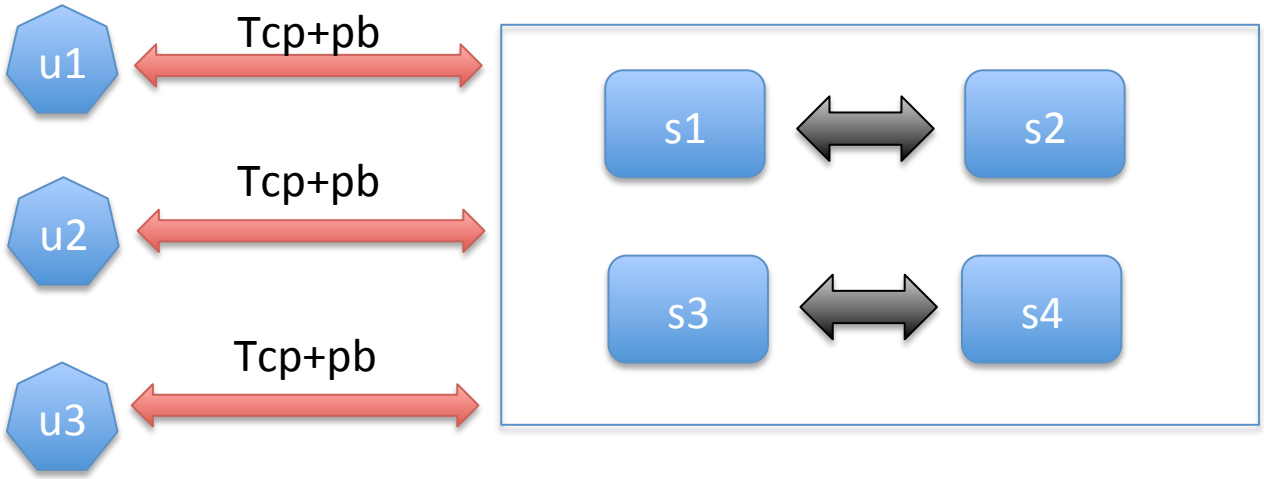
int32 ->	uint32
0 ->	0
-1 ->	1
1 ->	2
-2 ->	3
... ->	...
2147483647 ->	4294967294
-2147483648 ->	4294967295

Json格式

```
{
    "name":"小明",
    "age":"20",
    "major":"CS",
    "like":["Dota","Swimming", "Basketball"],
    "phone":[
        {
            "mobile":"18500137269",
            "brand":"iPhone6 Plus"
        },
        {
            "mobile":"13662301205",
            "brand":"Samung Note3"
        }
    ],
    "email":"xiaoming@gmail.com"
}
```

传输格式选取

- 和用户直连的接口，采用tcp+protobuf
- 内部系统采用http+json的Restful接口



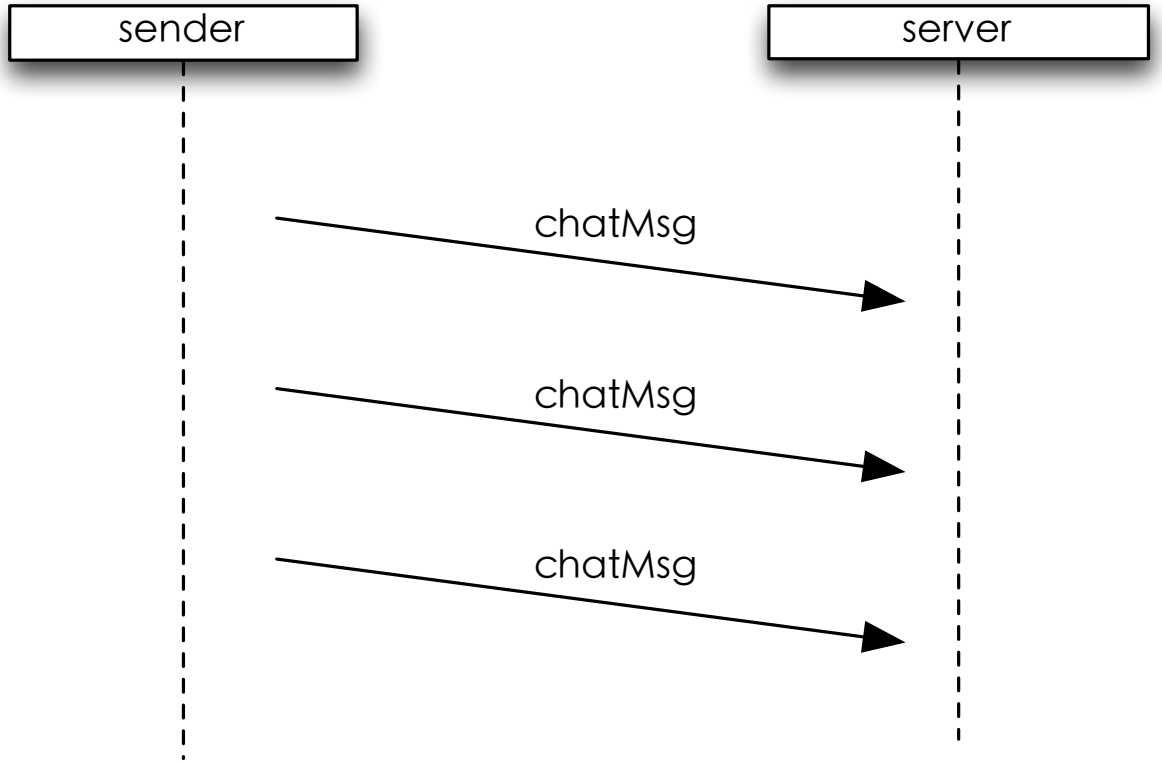
users

IM-servers

目录

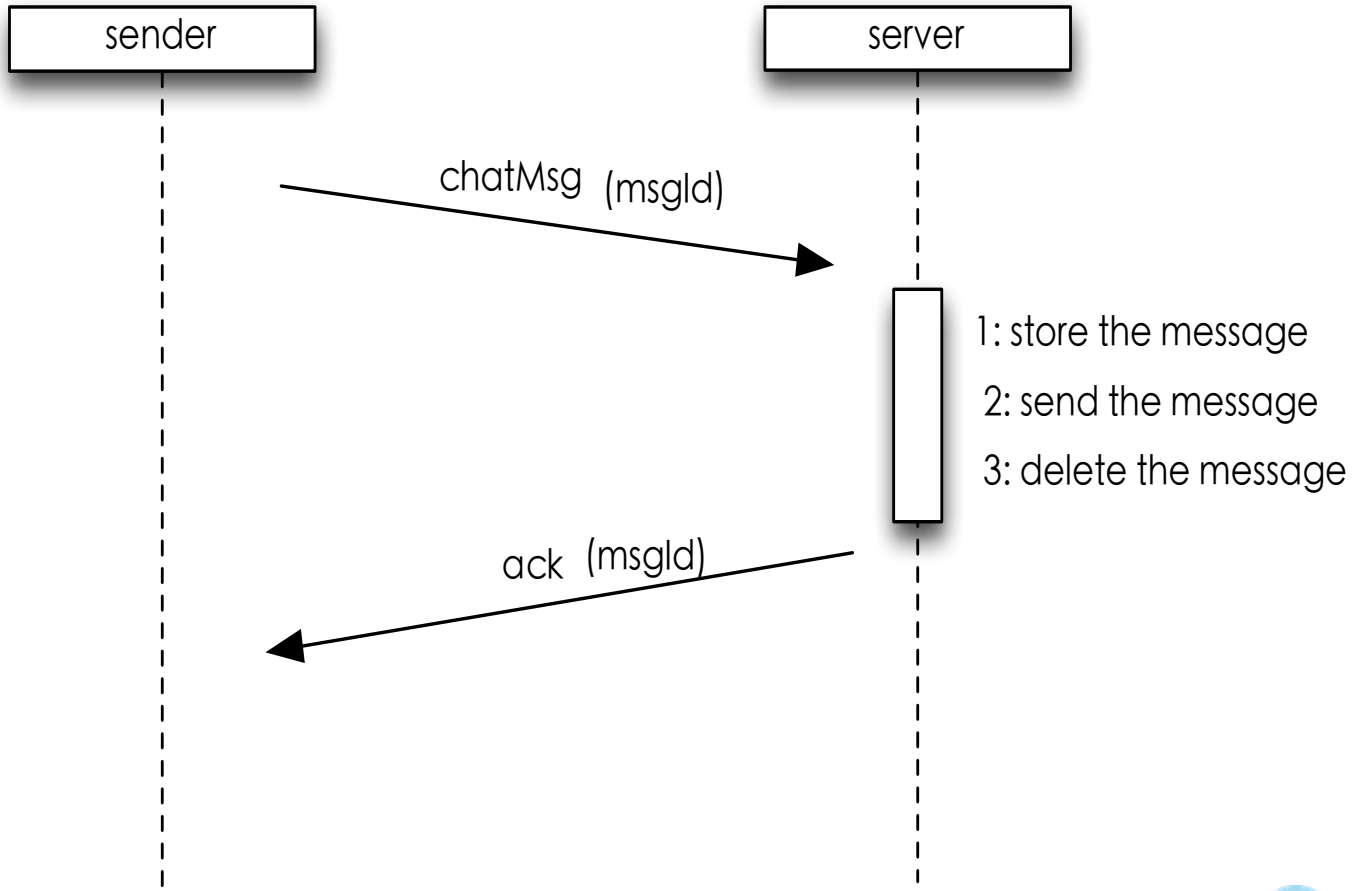
- 1：传输协议选取
- 2：传输格式选取
- 3：协议设计
- 4：架构设计

发送端协议设计



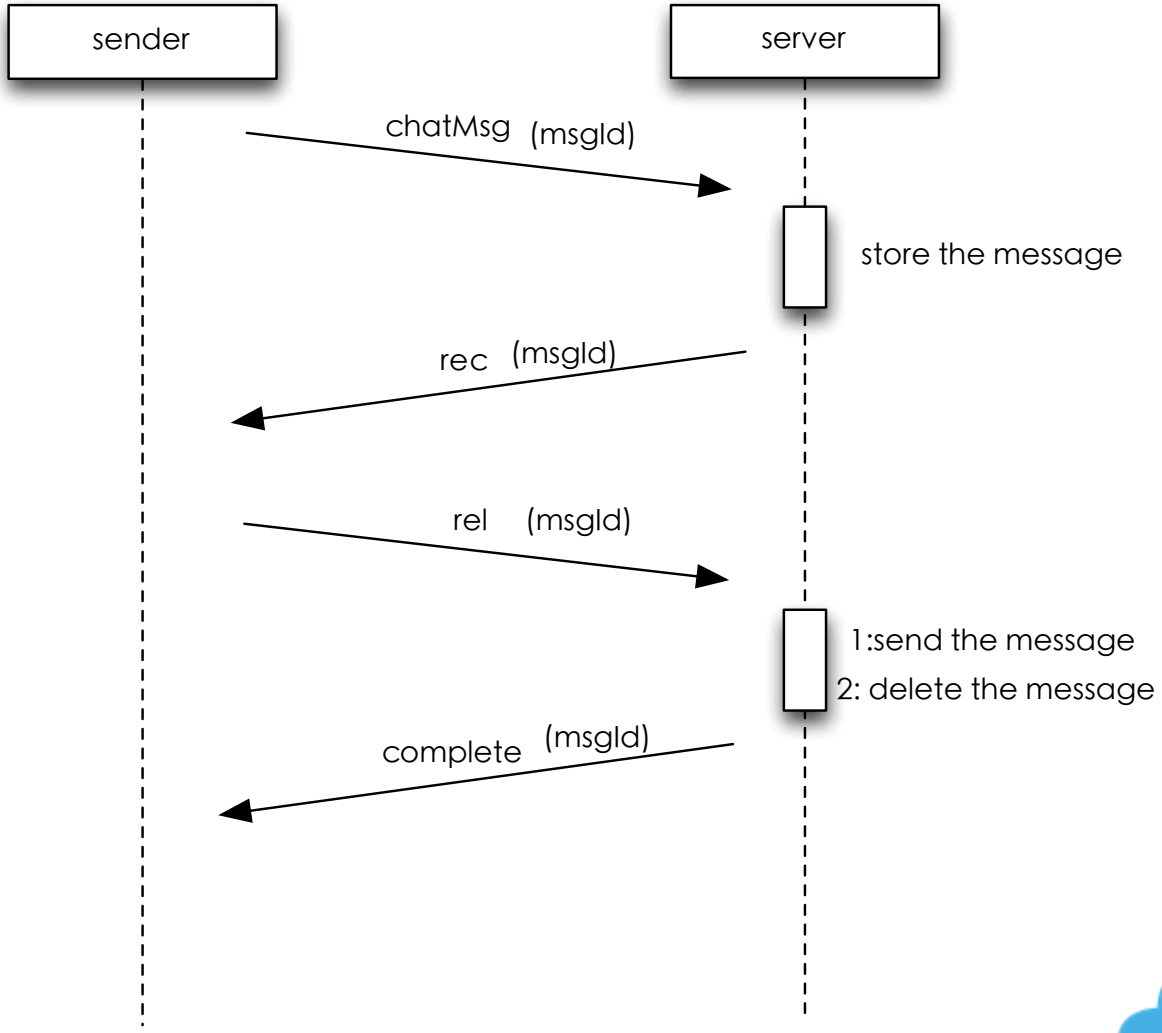
QOS=0

发送端协议设计



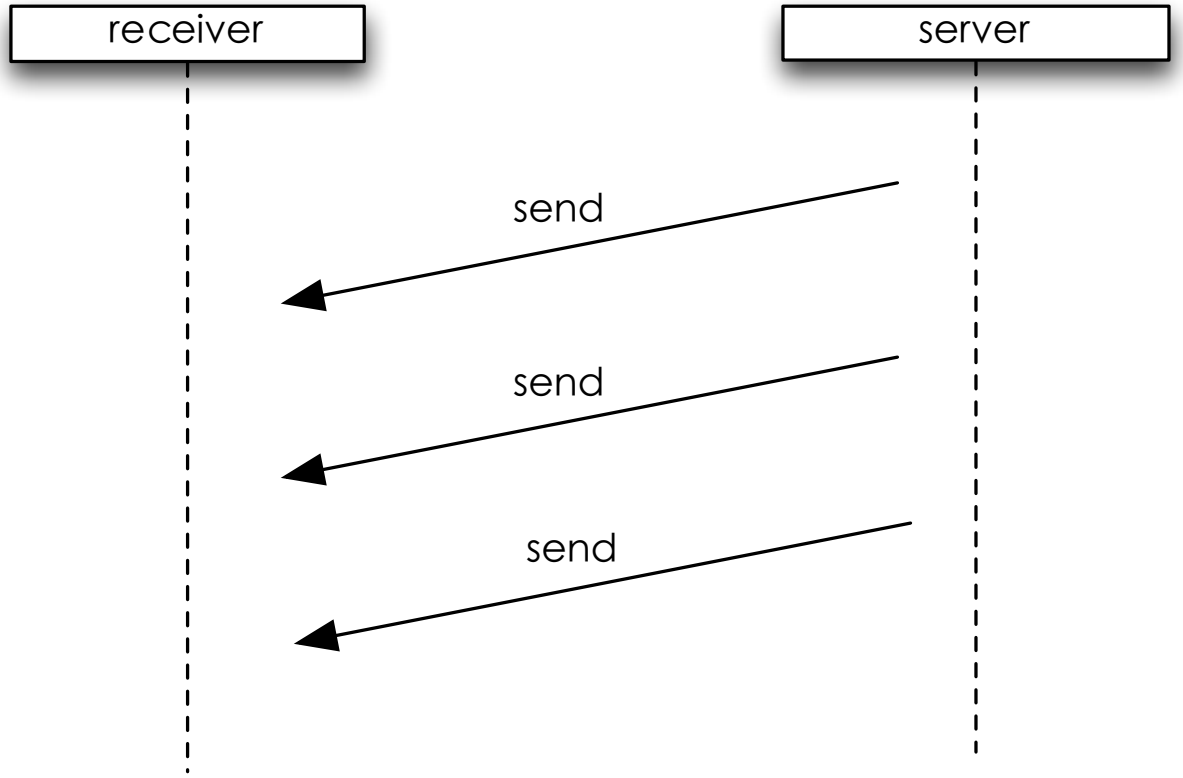
QOS=1

发送端协议设计

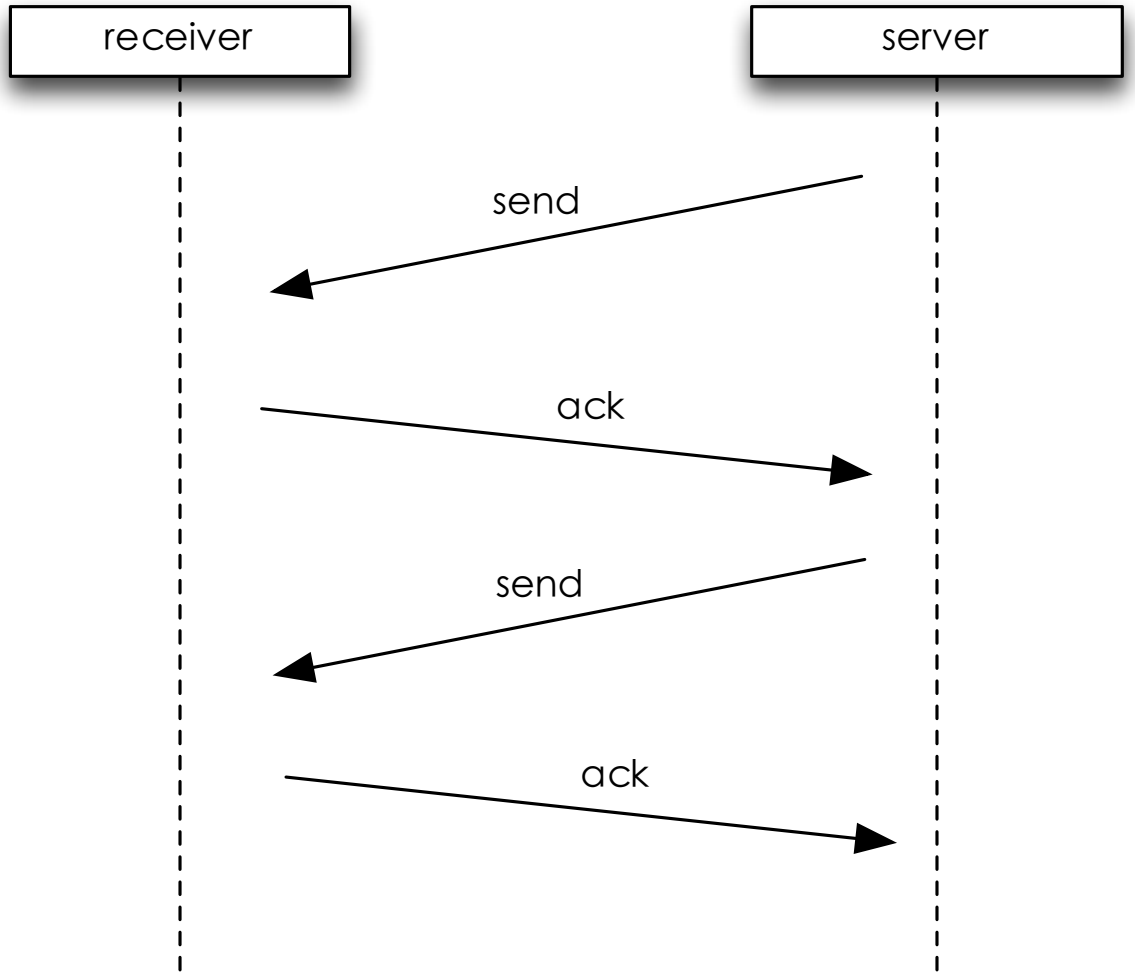


QOS=2

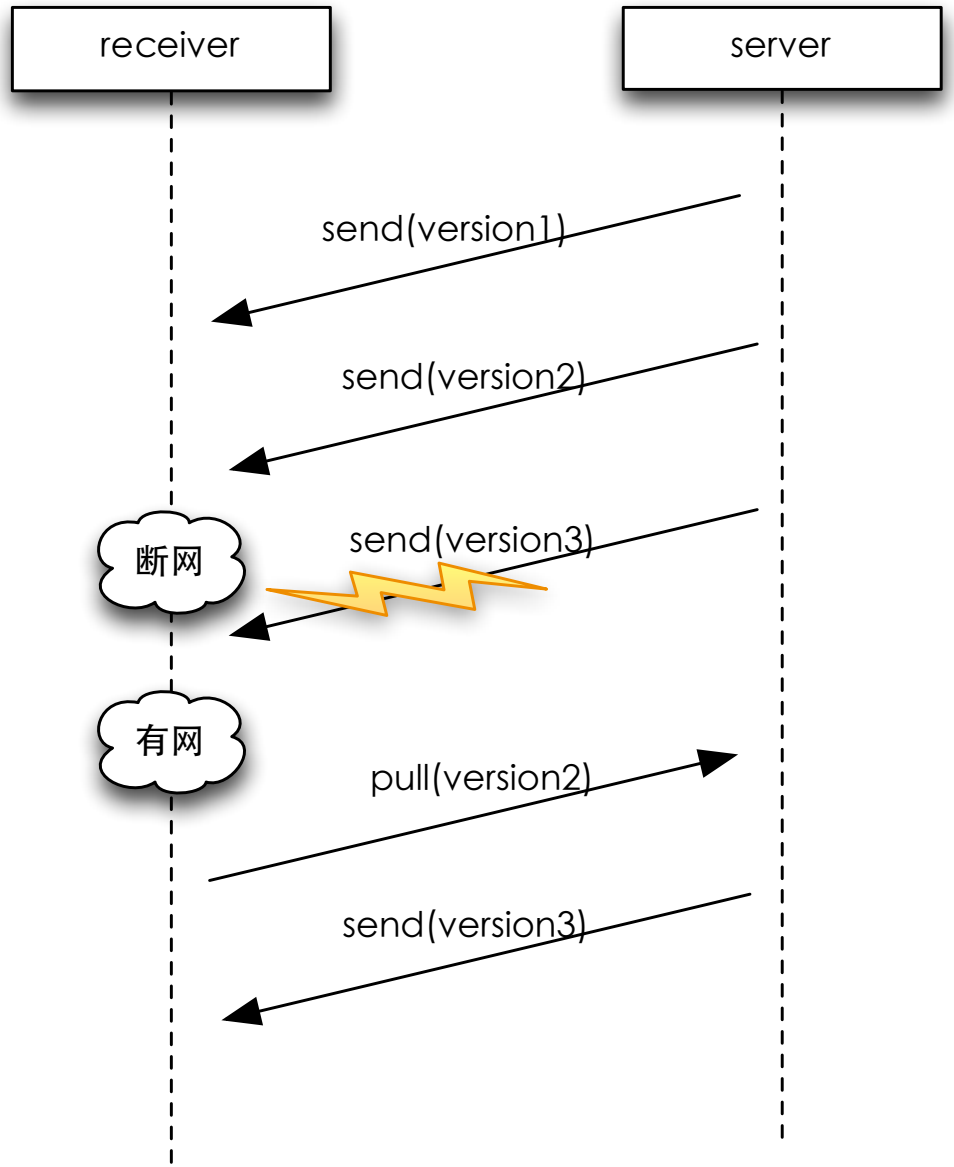
接收端协议设计



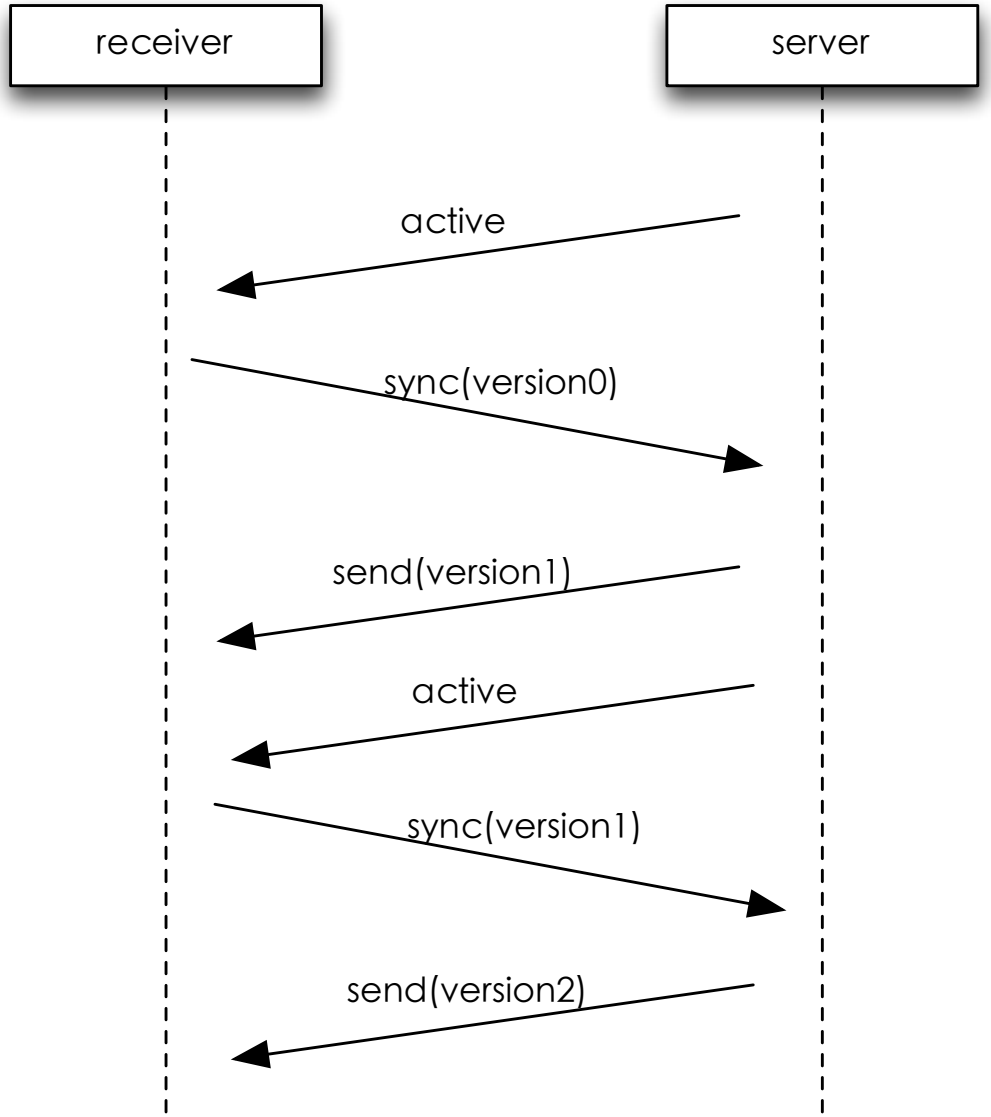
接收端协议设计



接收端协议设计



接收端协议设计



目录

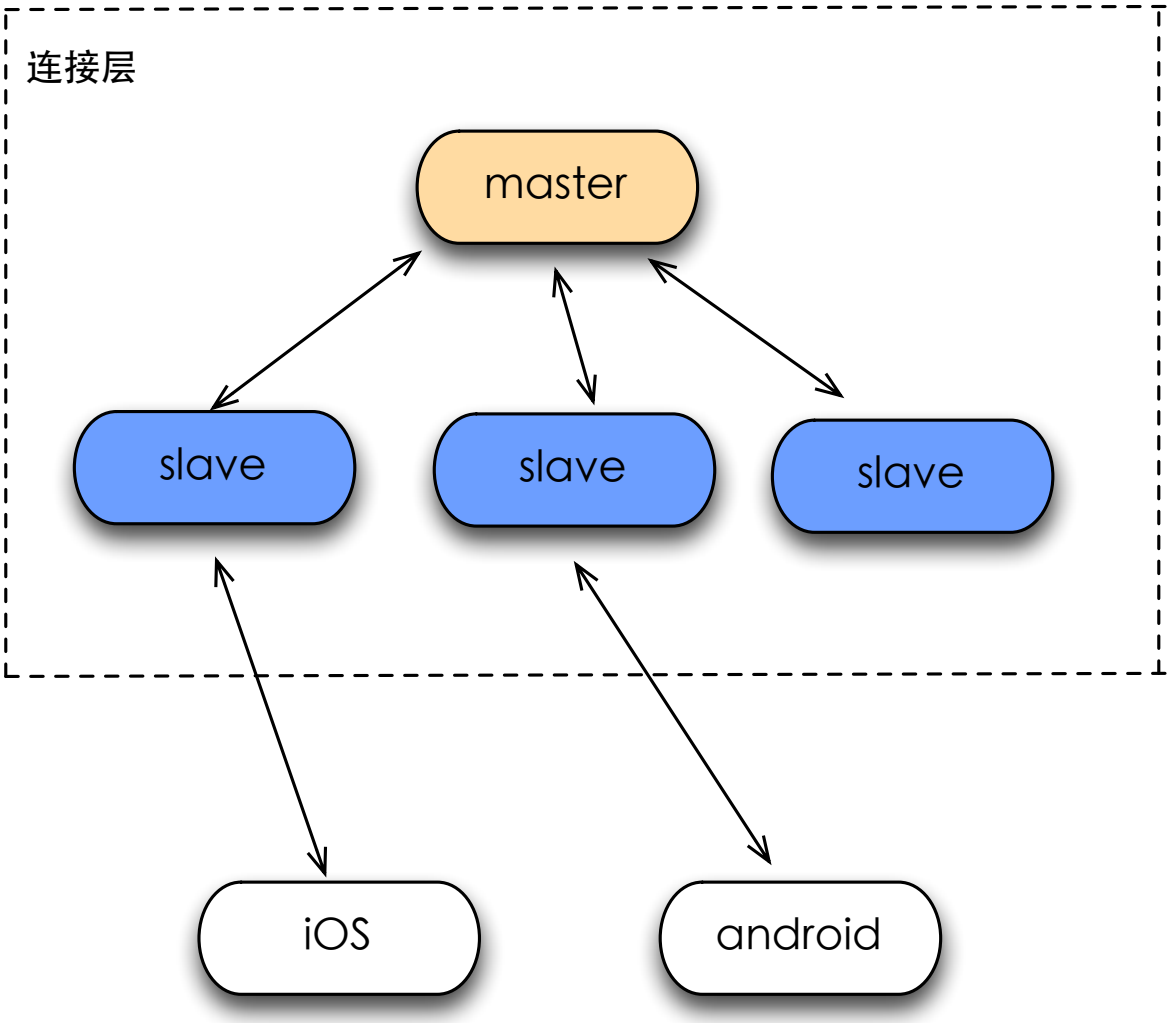
- 1：传输协议选取
- 2：传输格式选取
- 3：协议设计
- 4：架构设计

架构设计

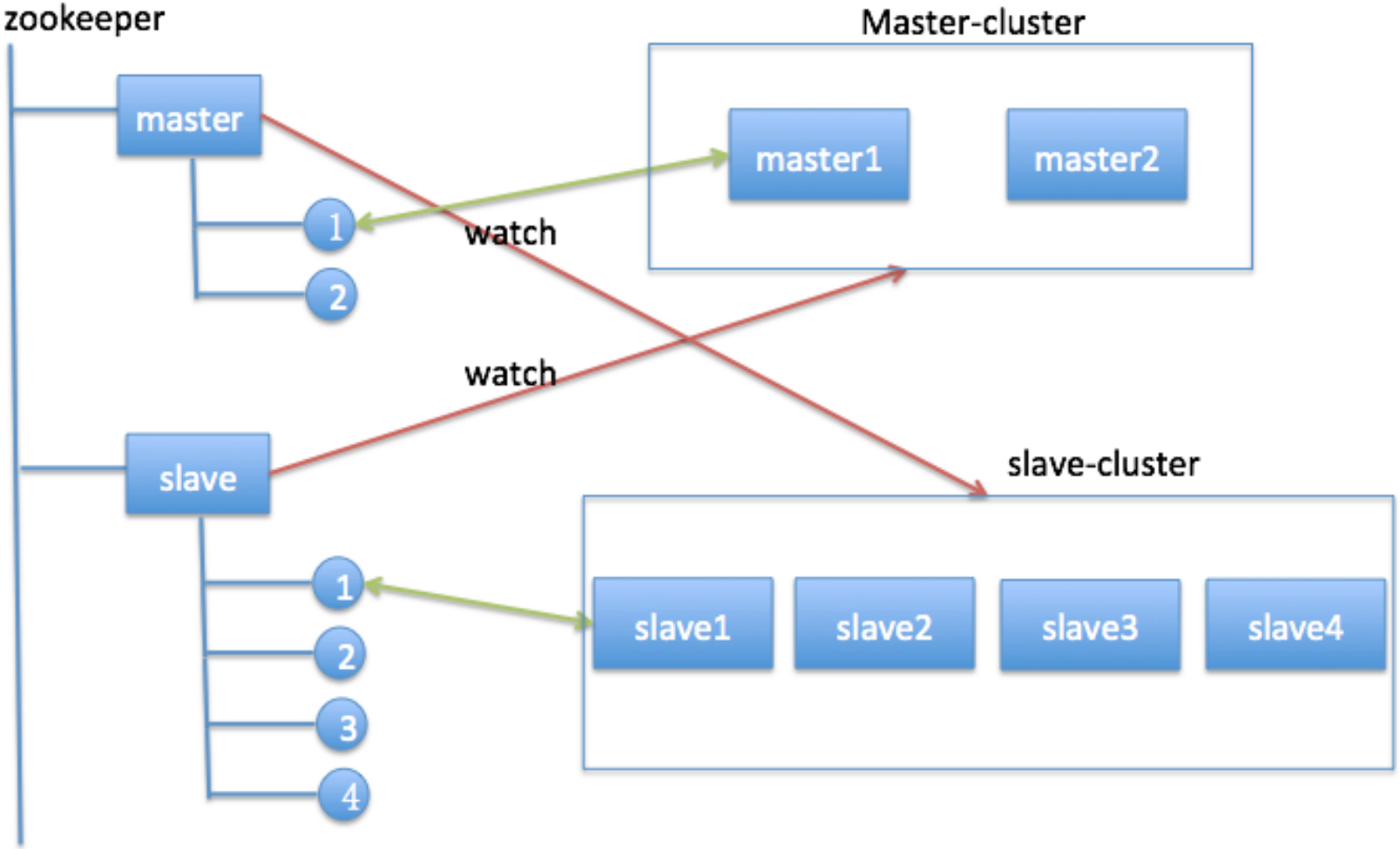
连接层

- 1：逻辑简单
- 2：可扩展
- 3：负载均衡
- 4：高并发
- 5：高可用

连接层



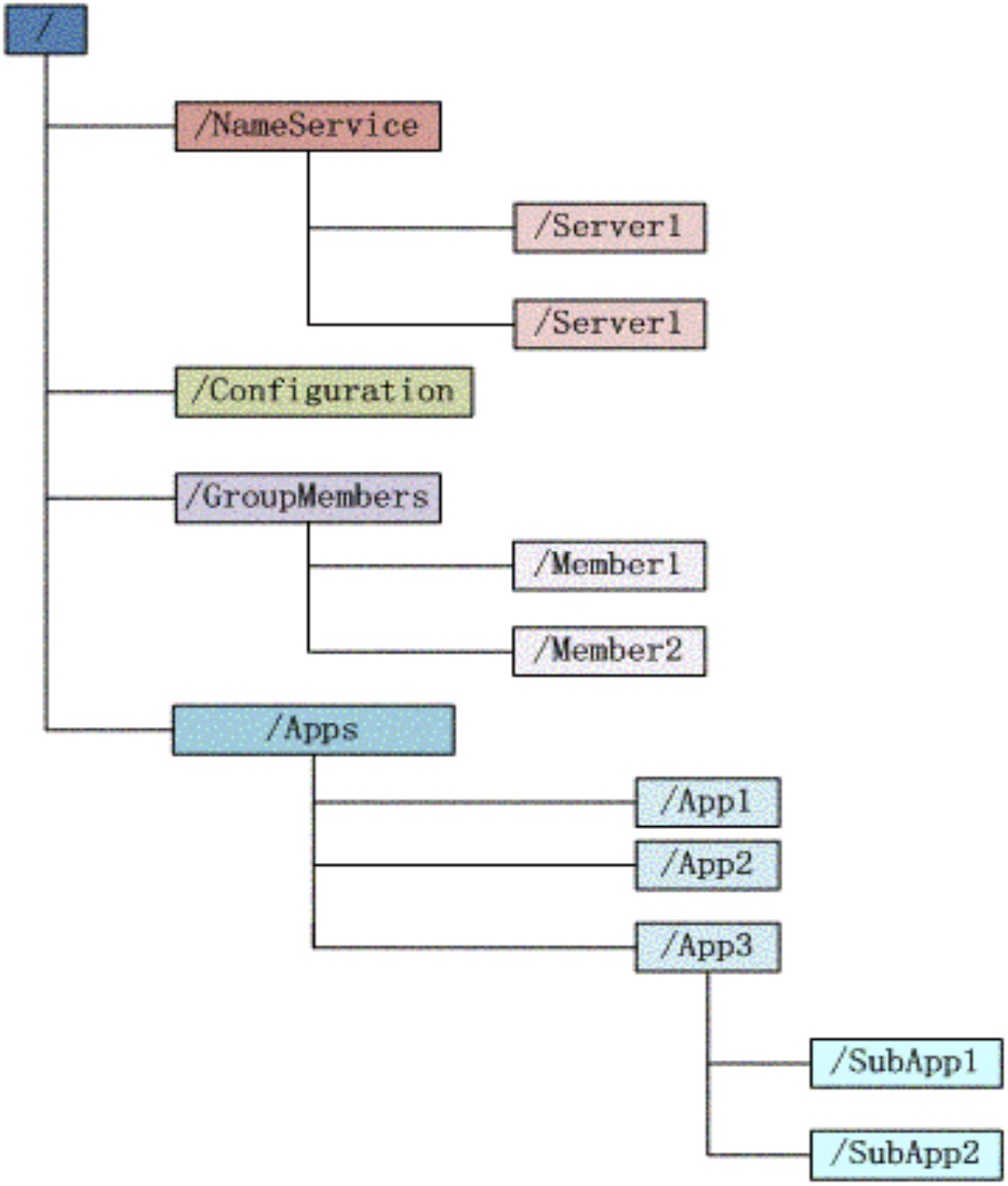
连接层 改进



红线代表watch；绿线代表心跳；

zookeeper

定义：
Zookeeper 作为一个分布式的服务框架，主要用来解决分布式集群中应用系统的一致性问题，它能提供基于类似于文件系统的目录节点树方式的数据存储，但是 Zookeeper 并不是用来专门存储数据的，它的作用主要是用来维护和监控你存储的数据的状态变化。通过监控这些数据状态的变化，从而可以达到基于数据的集群管理



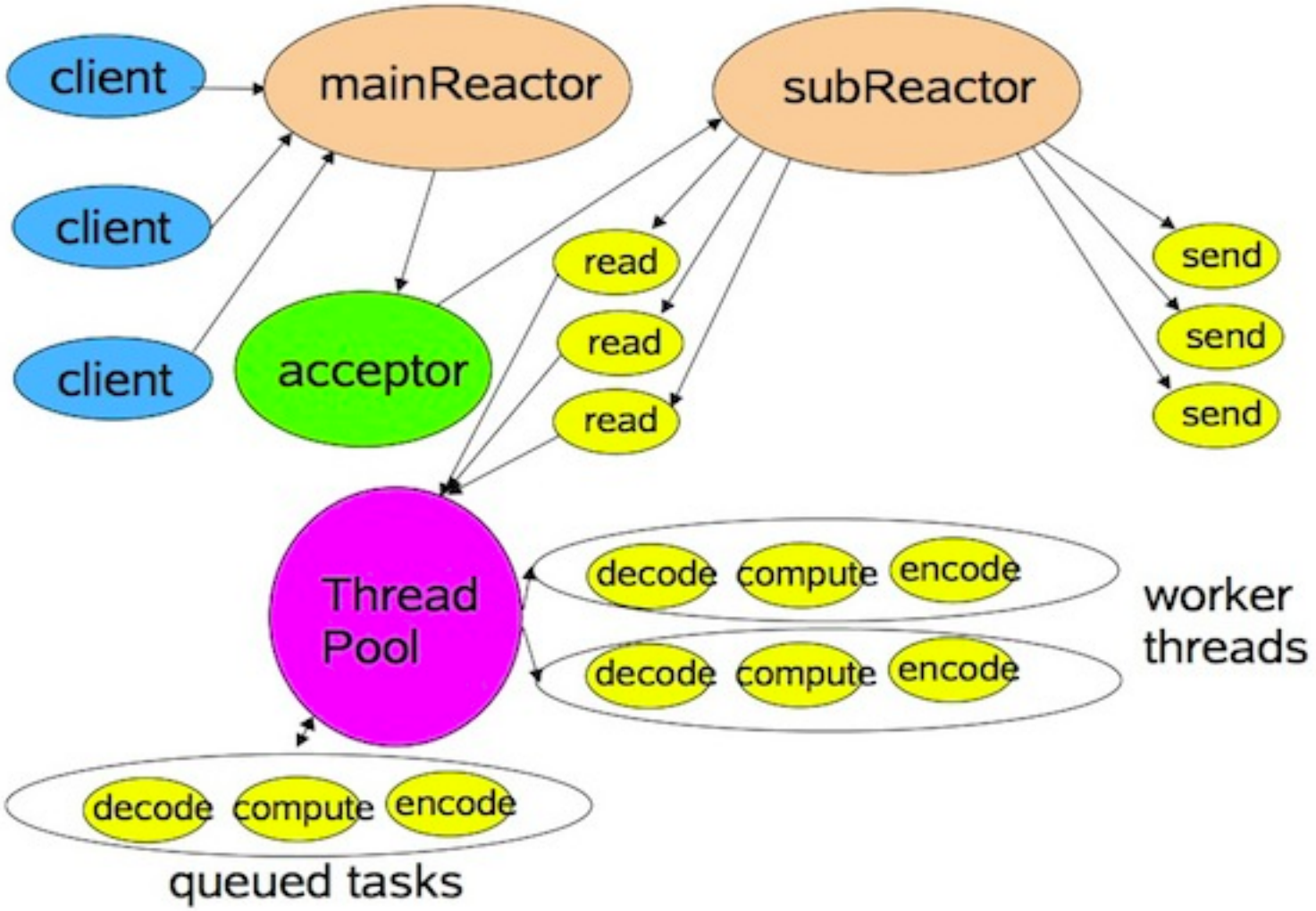
zk应用场景

- 统一命名服务
- 配置服务
- 集群管理
- 分布式锁

网络通信框架

- Netty介绍
  - 异步
  - 事件驱动

Reactor模式



百万连接？

- 连接层如何支持百万连接？
  - 1）io多路复用
  - 2）正常关闭连接 避免close_wait 句柄泄露
  - 3）io线程不要加入过重的业务处理逻辑
  - 4）合理的心跳周期
  - 5）合理设置接收和发送缓冲区
  - 6）使用针对缓冲区的内存池
  - 7）tcp参数优化

异常处理

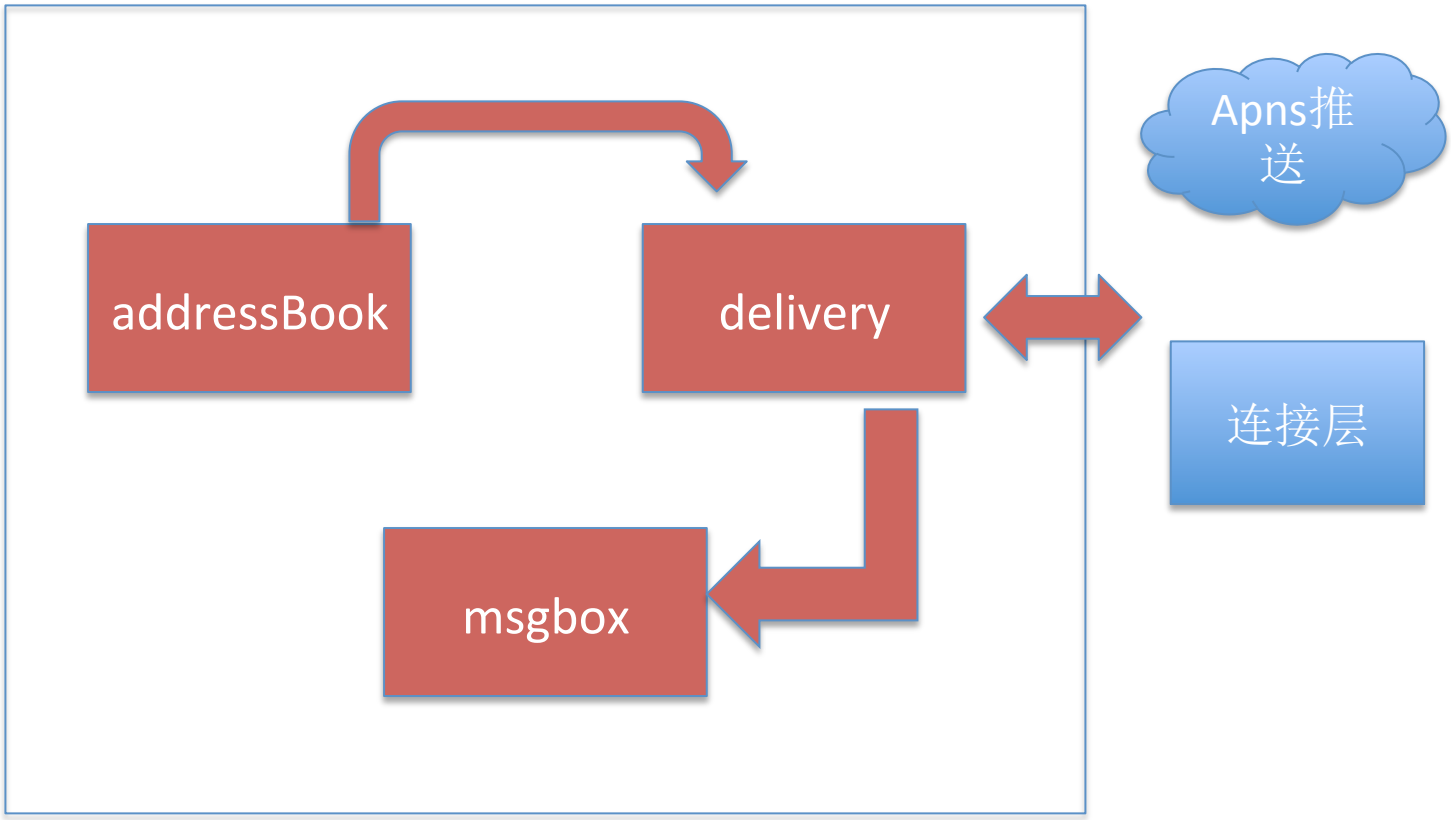
- 用户进入无网环境？
  - 进入无网环境和断掉wifi的区别？
  - slave如何感知?
  - IM-sdk如何感知？

- Slave挂掉？

逻辑层

- addressBook（组服务）
- msgBox
- delivery

逻辑层

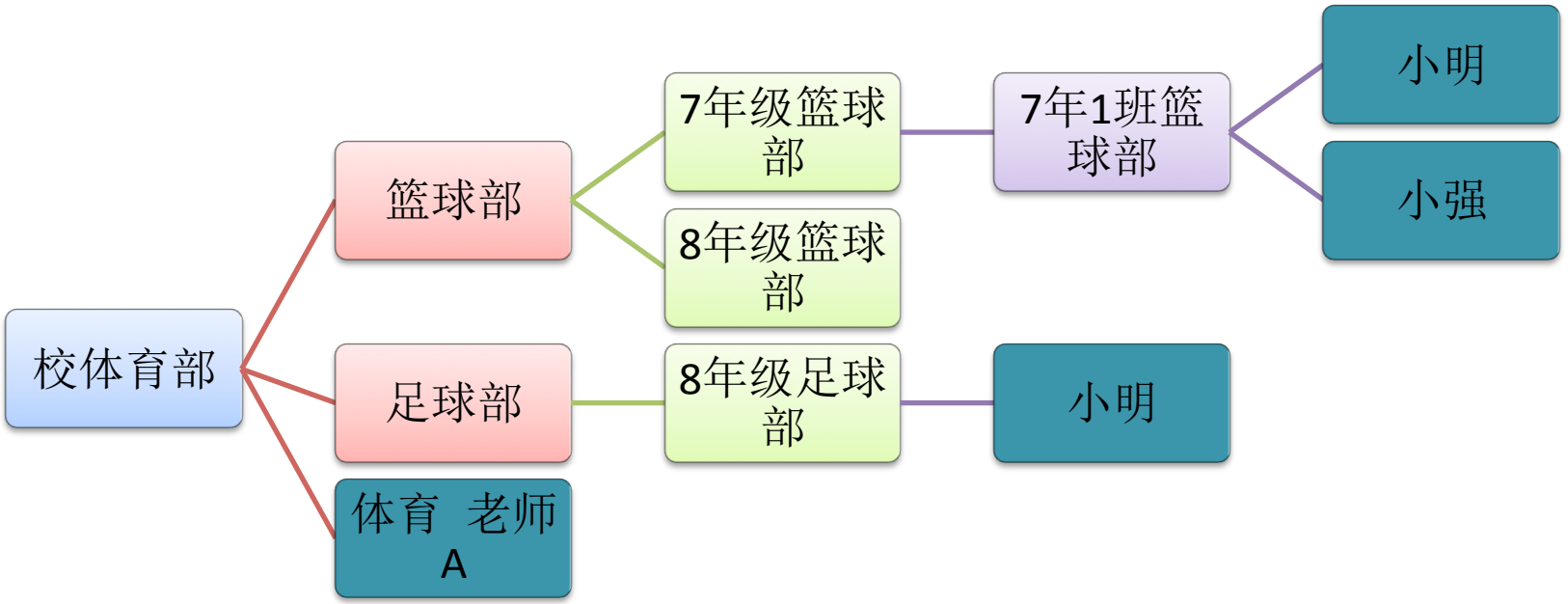


addressBook（组服务）

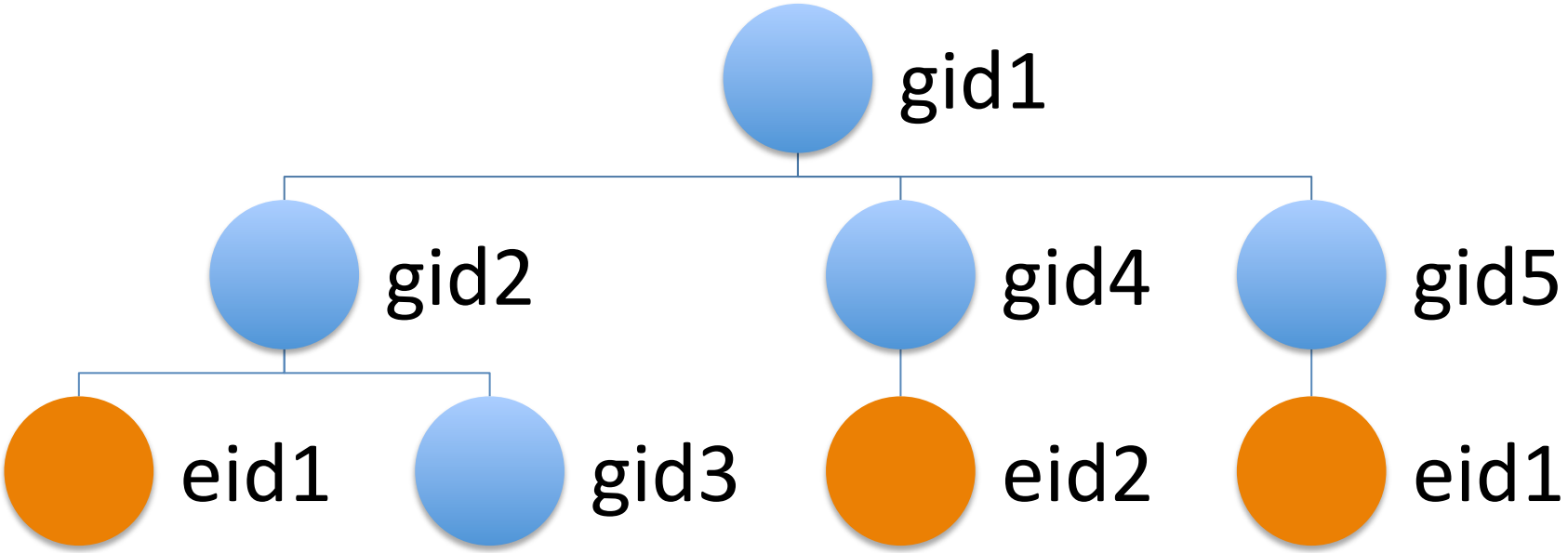
- 通讯录
- 群聊
- 单位组织架构
- 兴趣小组

- 组服务，有向无环图

addressBook（组服务）

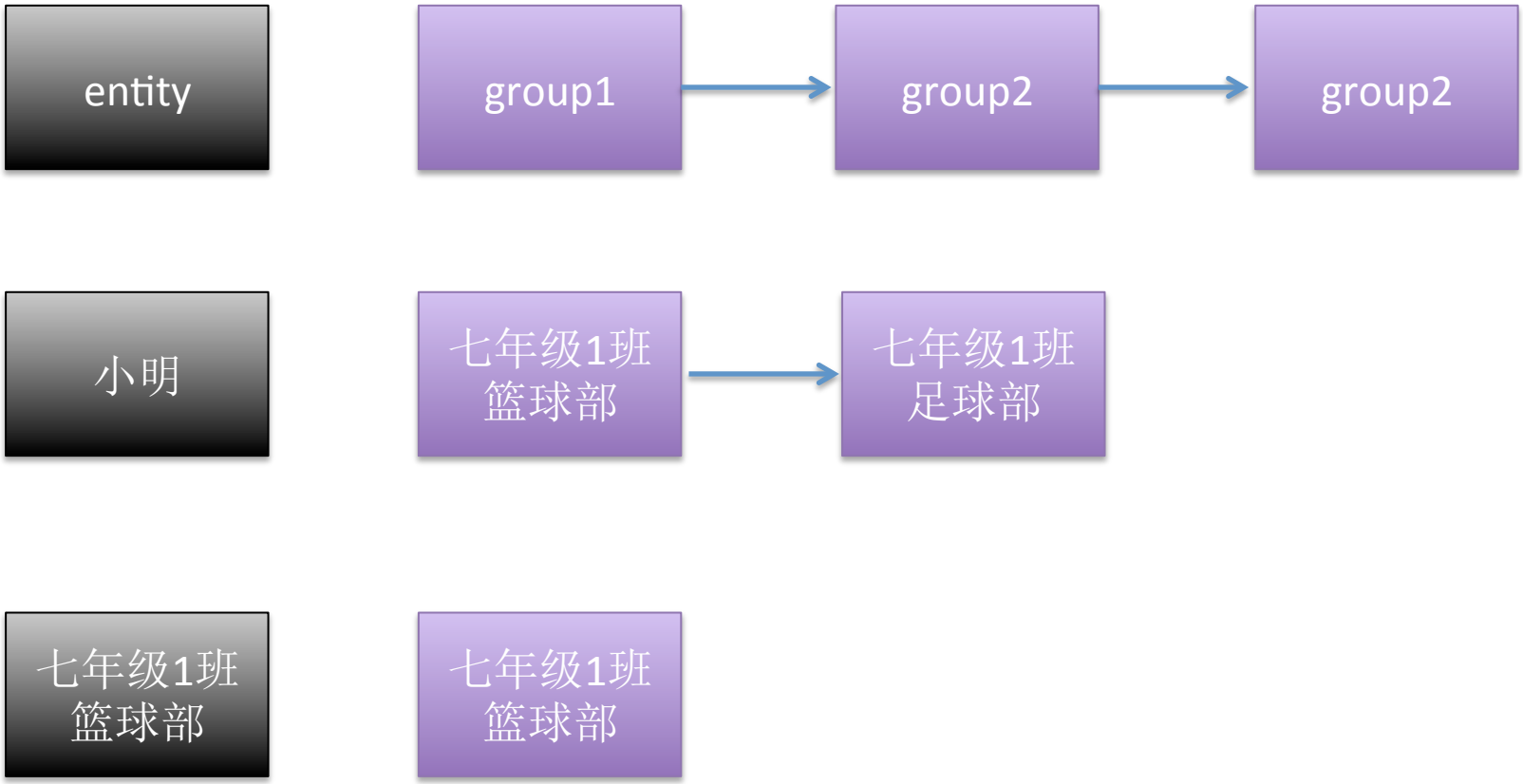


模型抽象



1：树形结构(多叉)
2：有向无环
3：每个叶子节点都是entity

倒排

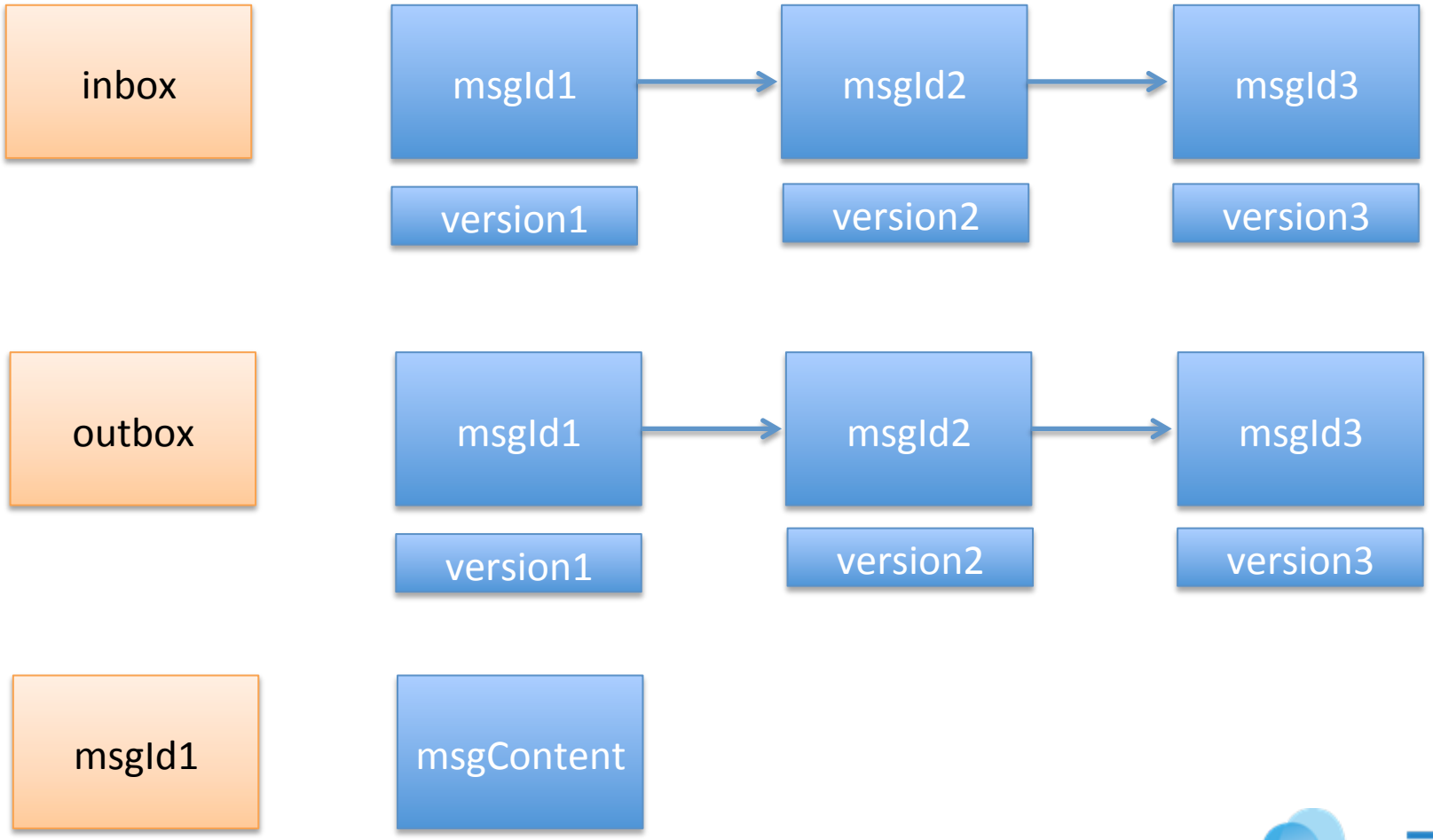


遍历算法

- 查找某个组内的人？
  - DFS or BFS
  - 去重

- 查找某个人在哪些组内？
  - 根据倒排反向查找 DFS or BFS
  - 去重

msgbox



msgbox

- 主要接口介绍：基于版本查找
  - 参数：fromVersion，toVersion，Offset，size，sortType，boxType

比如：有如下10条消息[msg_id:version]：

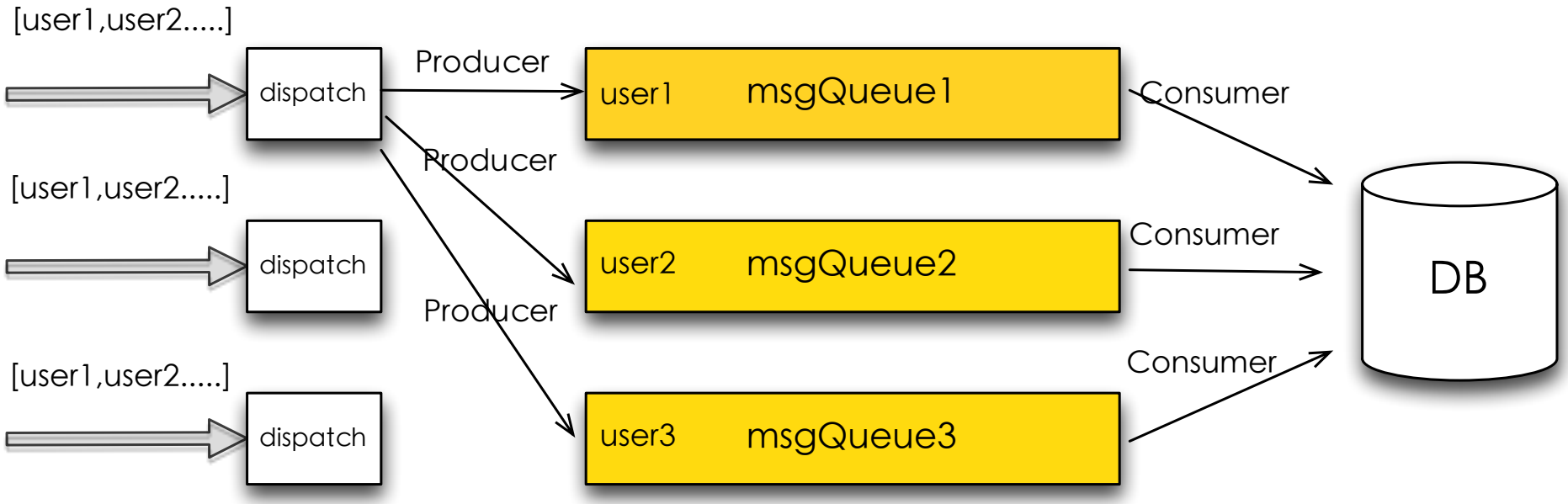
CHAT: [0:7, 1:8, 2:9, 3:10, 4:11, 5:12, 6:13, 7:14, 8:15, 9:16]

input: from_version:9, to_version:15, sort_type:desc, offset:2, size:3, box_type:CHAT

output: [5:12,4:11,3:10]

msgBox线程模型

- 生产者/消费者



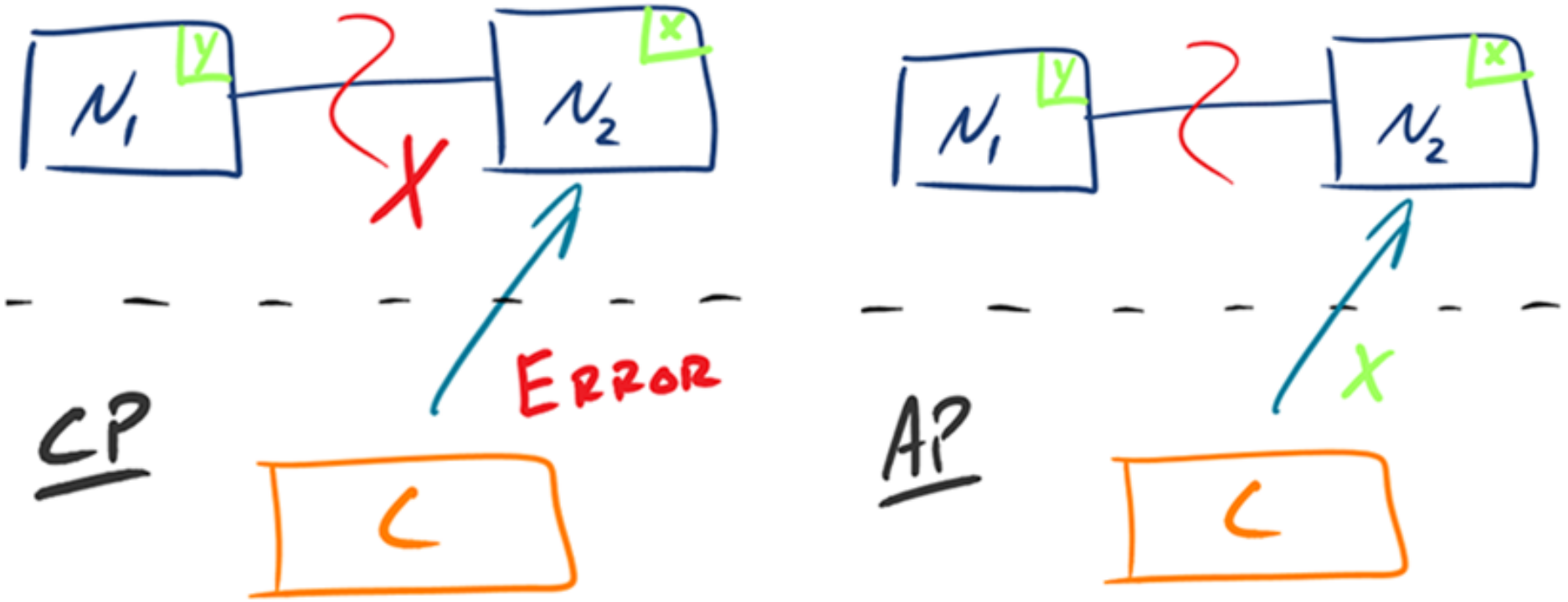
Delivery模块

- 串联各个模块
  - 从连接层master中查找路由信息
  - 从addressBook中查找组成员
  - 向msgbox中插入消息
  - 向apns发推送

CAP理论

- 针对分布式系统
- Consistency
- Avaliability
- Partition Tolerance
- 三者只能取其二，不可能同时满足

CAP理论

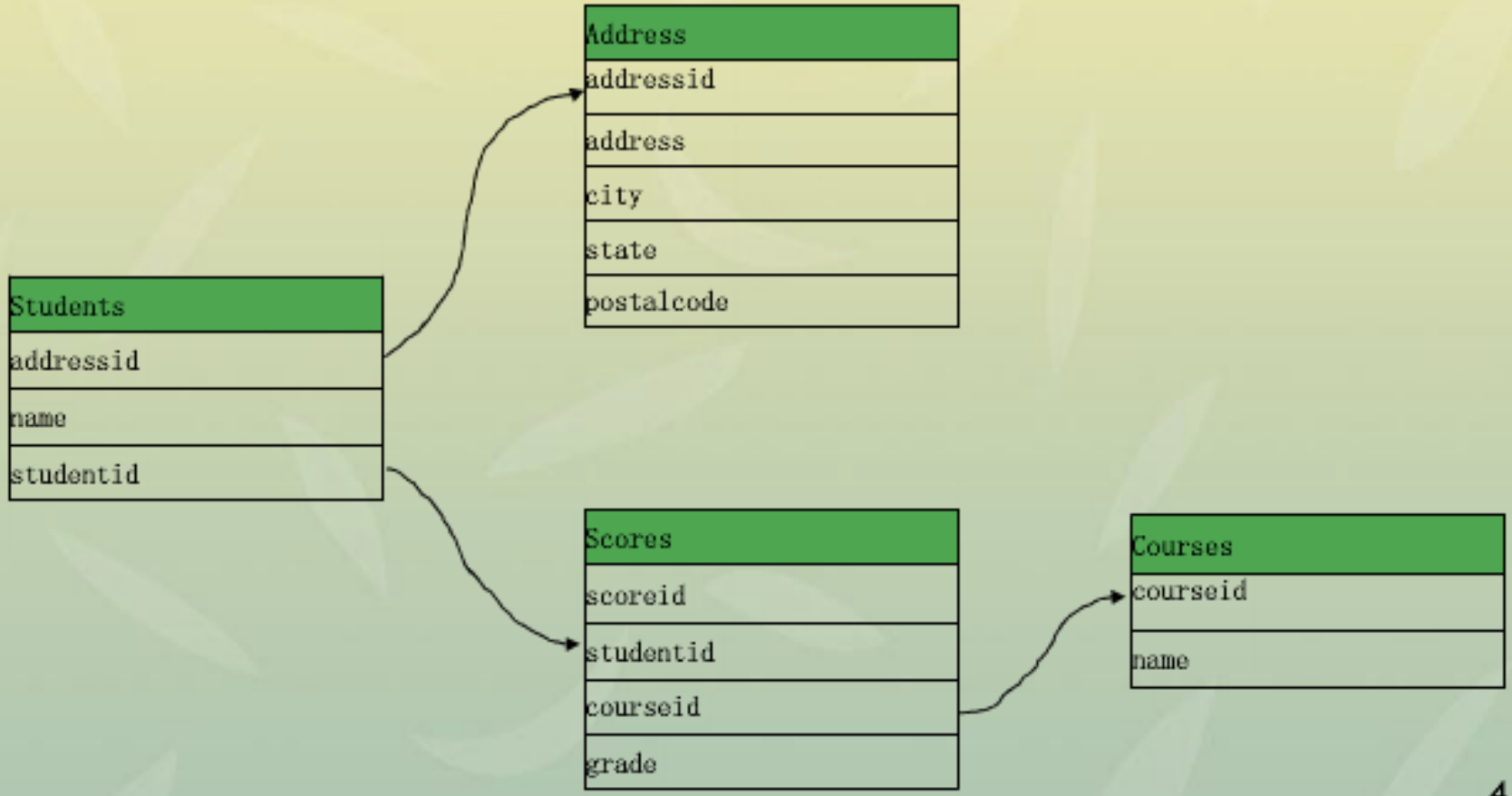


存储层

- MongoDB：
  - Schema-free
  - Auto-sharding
  - Json
  - Low latency
  - CP & AP
- Redis
  - Key-value

mongoDB

- 关系数据库的表结构（学生、地址、成绩、科目）：

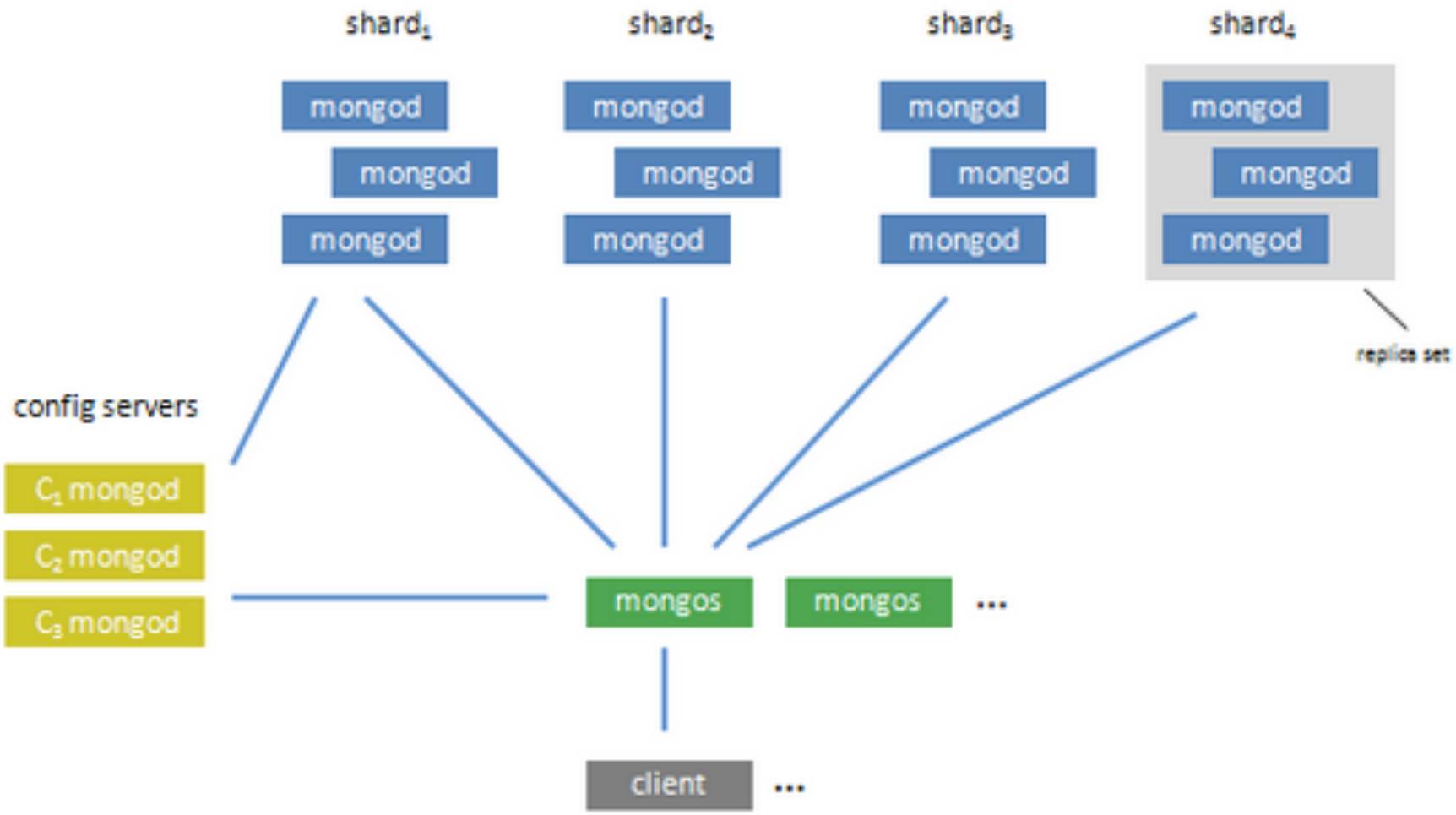


mongoDB

- NoSQL的表结构（学生、地址、成绩、科目）：

Students
_id : 007
name : "Jane"
address :
address : "123 Main St."
city : "New York"
state : "NY"
postalcode : "10014"
scores :
Biolgy : 4.0
English : 3.0

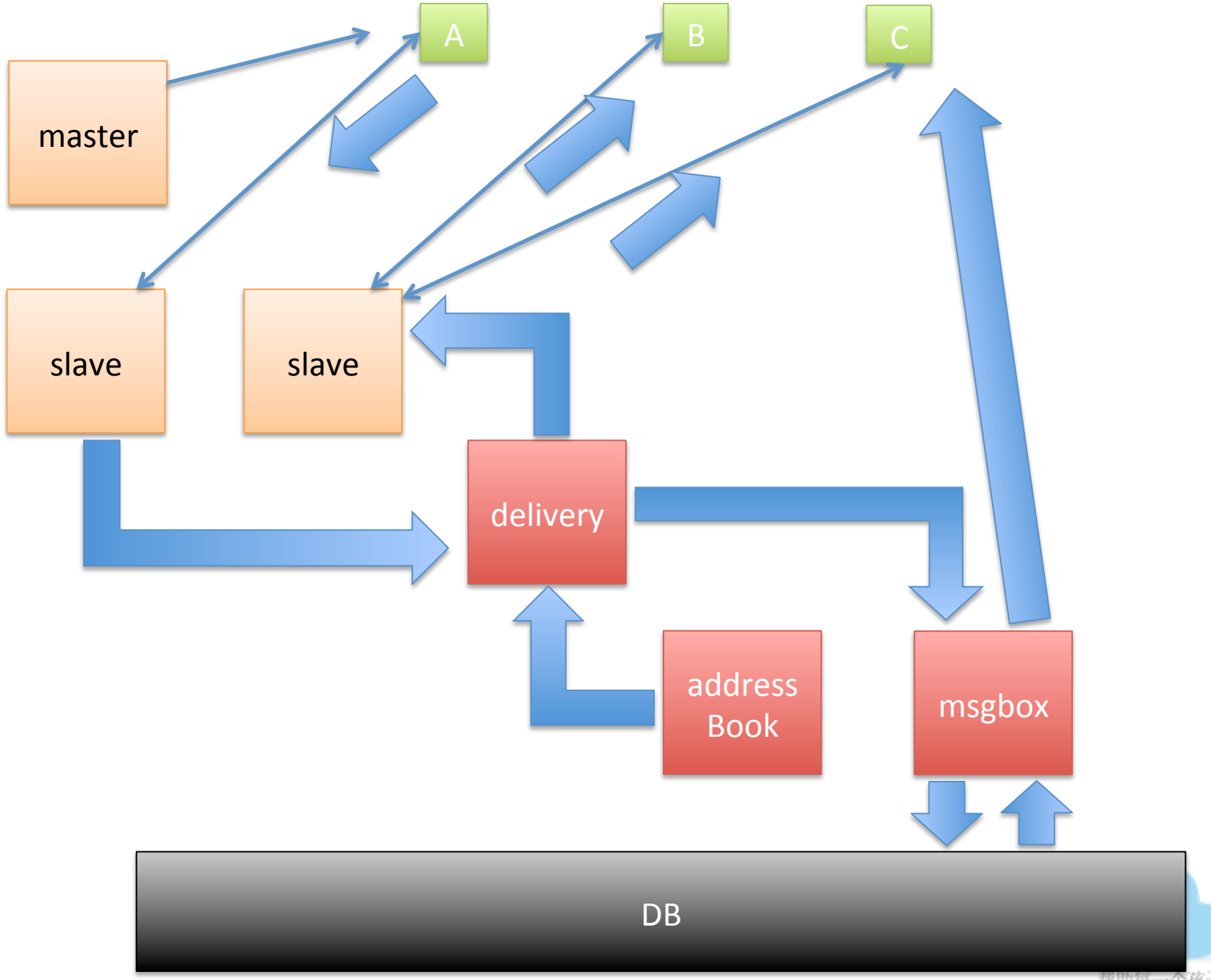
Mongo架构图


shard1
shard2
shard3
shard4
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
replica set
config servers
C1 mongod
C2 mongod
C3 mongod
mongos
mongos
...
client
...

存储层

- Redis
  - Slave和user的mapping关系<userId, slaveId>
  - Slave连接的user count. <slaveId, userCount>

- mongoDB
  - Inbox:[list….]
  - Outbox:[list….]



性能优化

- 线程池
- 批处理（高吞吐和低延迟不可兼得？）
- http连接池
- DB表级锁->行级锁
- 测试消息穿透所有服务，并记录下各个阶段的时间，寻找热点

性能

- 单聊性能100ms左右
- 最大吞吐量：单机1000/s，无丢失消息

- 性能热点：
  - 消息的持久化
- 消灭性能热点：
  - 逻辑层与存储层之间增加高性能缓存层

总结

- 调研 设计 开发 优化 2人 2月
- 目前内部接入中
- 1）生产者/消费者 模型的熟练掌握
- 2）性能调优工具jprofile，jconsole等的熟练掌握
- 3）对各类开源技术的熟练使用
- 4）带着高并发的思维去设计和coding

谢谢大家

欢迎加入云校！
用技术改变教育！
talent@iyunxiao.com