Dive into NSQ

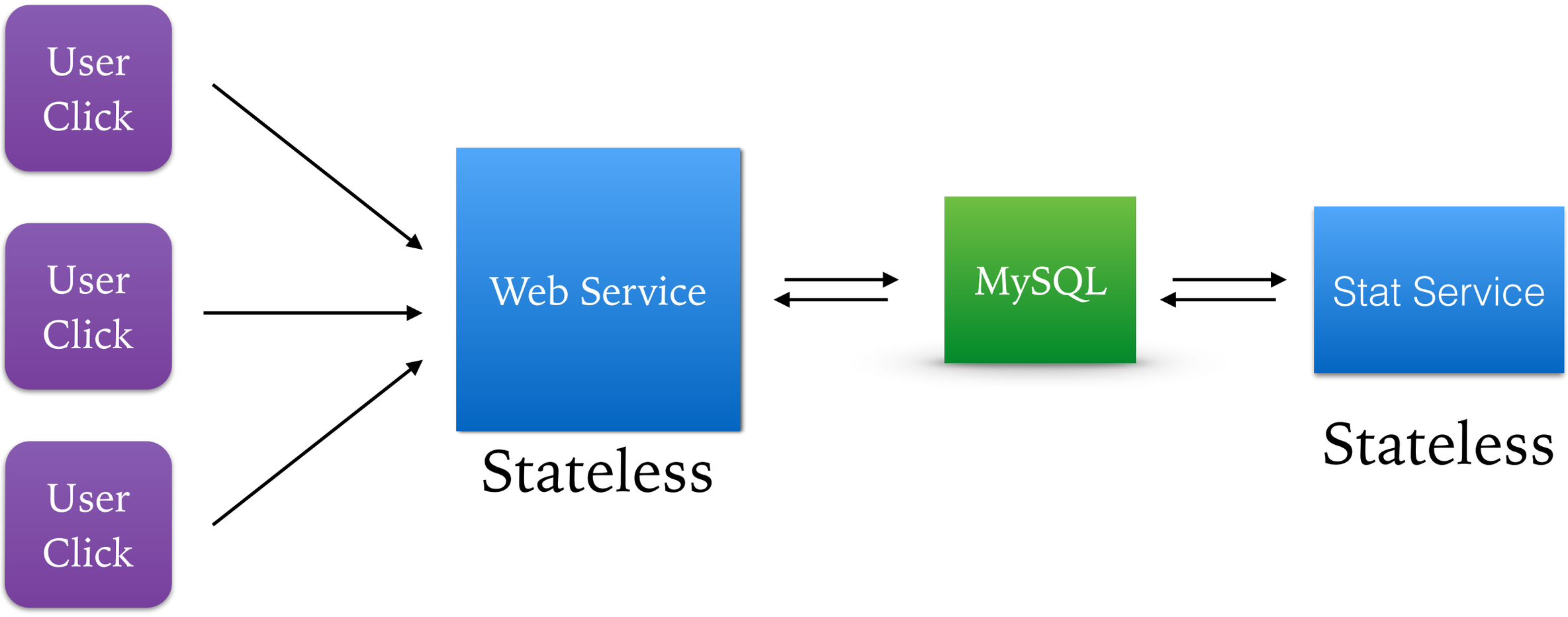
舜飞 - 陈冶

消息队列服务

- 面向跨进程/跨服务器通讯的组件
- 异步通信，将可并行化处理的同步操作解耦

使用案例

广告点击数统计



使用案例

广告点击数统计

1. MySQL 超载

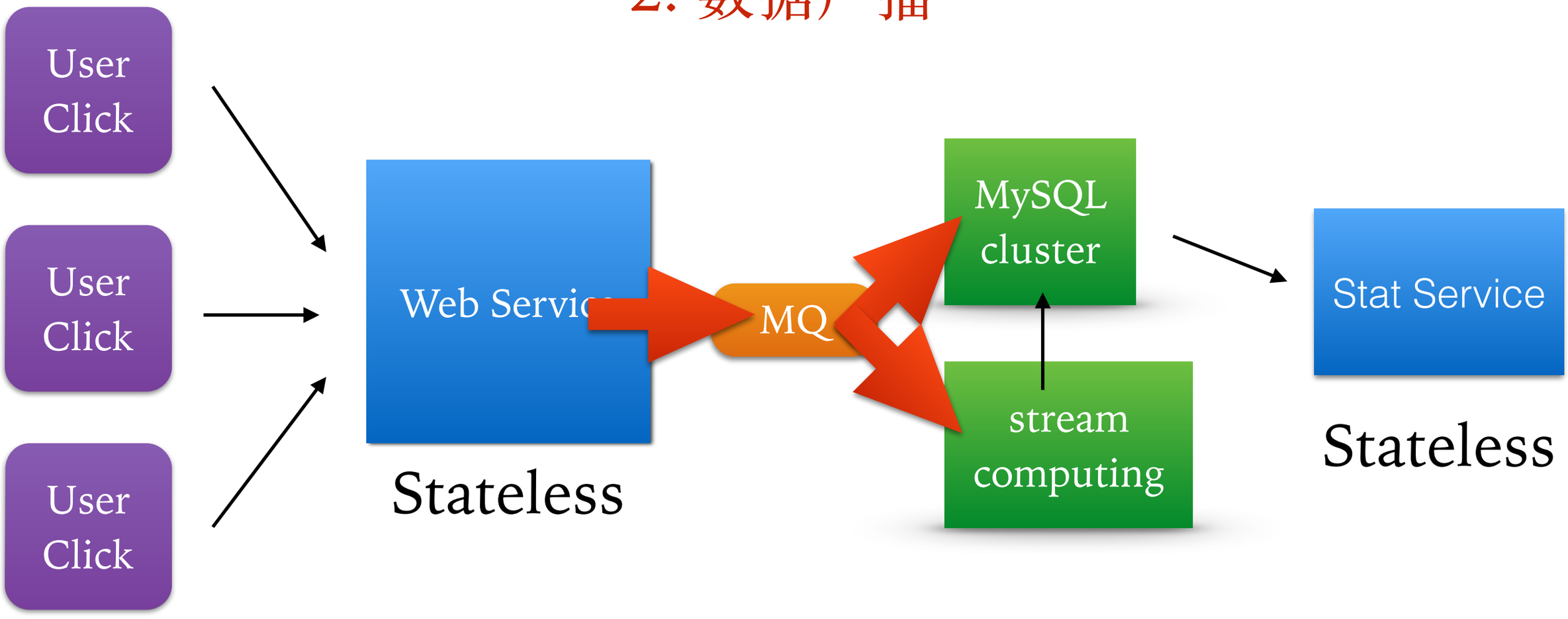
使用需求

- 数据缓冲，提高可用性，缓冲服务故障

使用案例

广告点击数统计

2. 数据广播



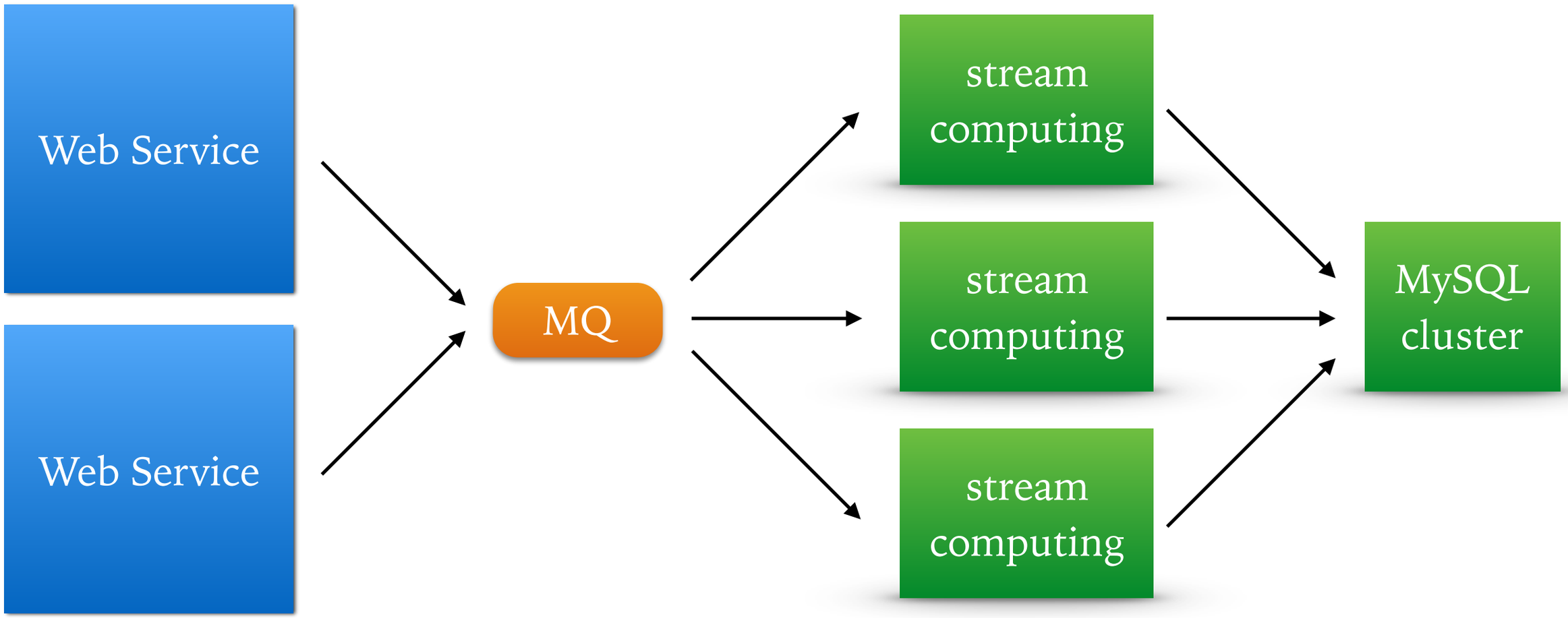
使用需求

- 数据缓冲，提高可用性，缓冲服务故障
- 数据广播，分发给多个服务

使用案例

广告点击数统计

3. 负载均衡



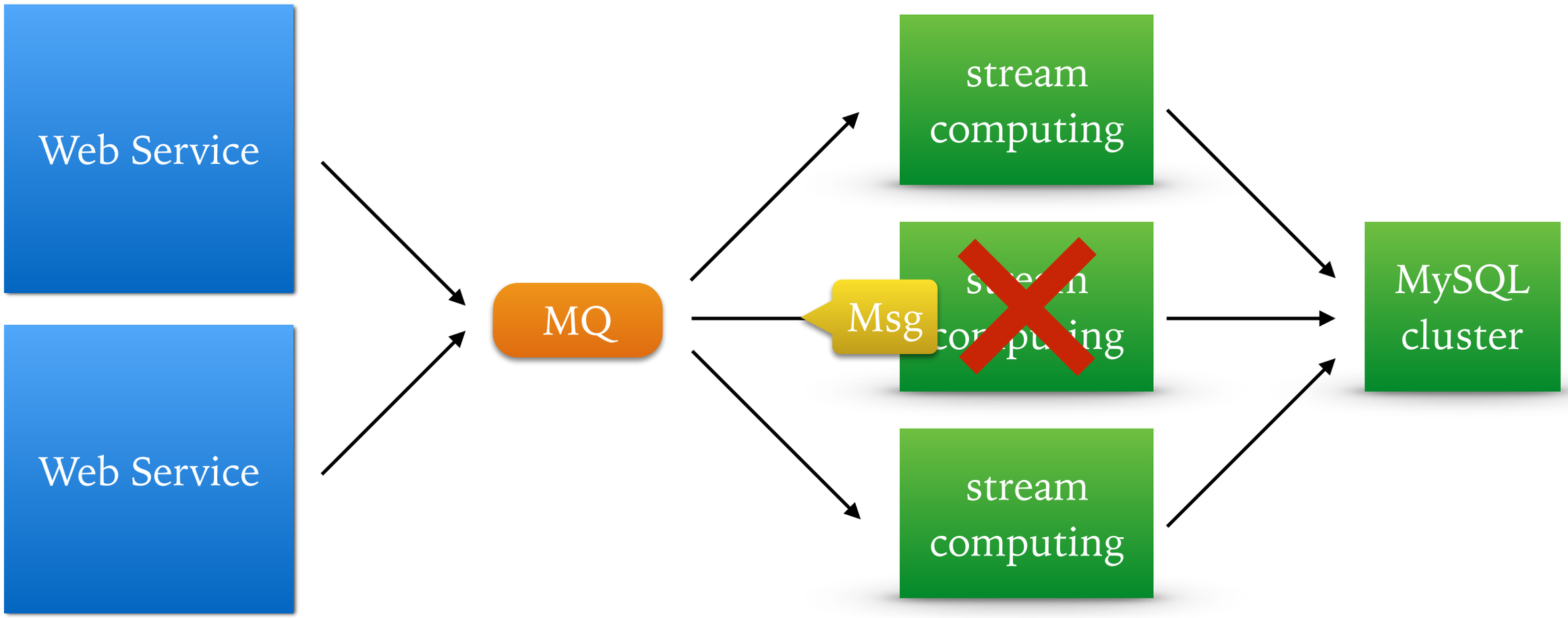
使用需求

- 数据缓冲，提高可用性，缓冲服务故障
- 数据广播，分发给多个服务
- 负载均衡，提高消费者的扩展性

使用案例

广告点击数统计

4. 正确消费确认



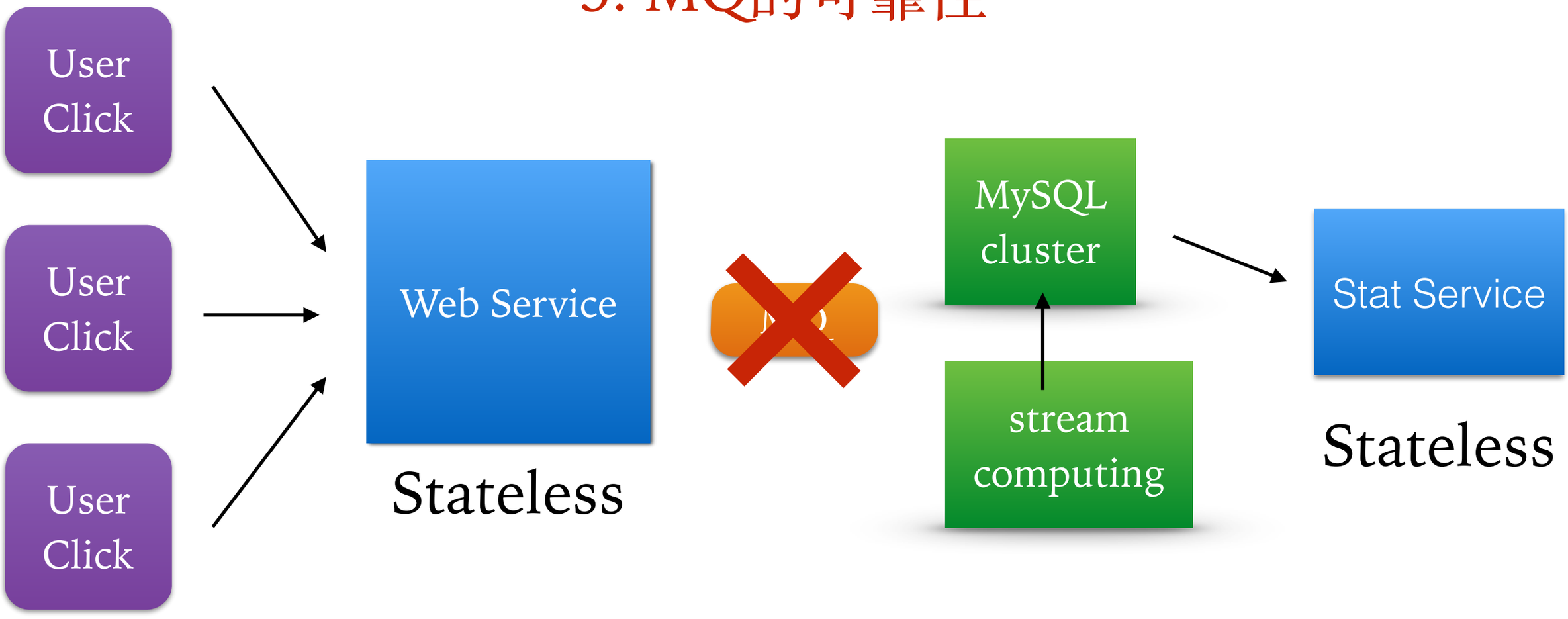
使用需求

- 数据缓冲，提高可用性，缓冲服务故障
- 数据广播，分发给多个服务
- 负载均衡，提高消费者的扩展性
- 消费反馈，确保消息不丢失

使用案例

广告点击数统计

5. MQ的可靠性



使用需求

- 数据缓冲，提高可用性，缓冲服务故障
- 数据广播，分发给多个服务
- 负载均衡，提高消费者的扩展性
- 消费反馈，确保消息不丢失
- MQ：分布式部署，排除自身单点故障

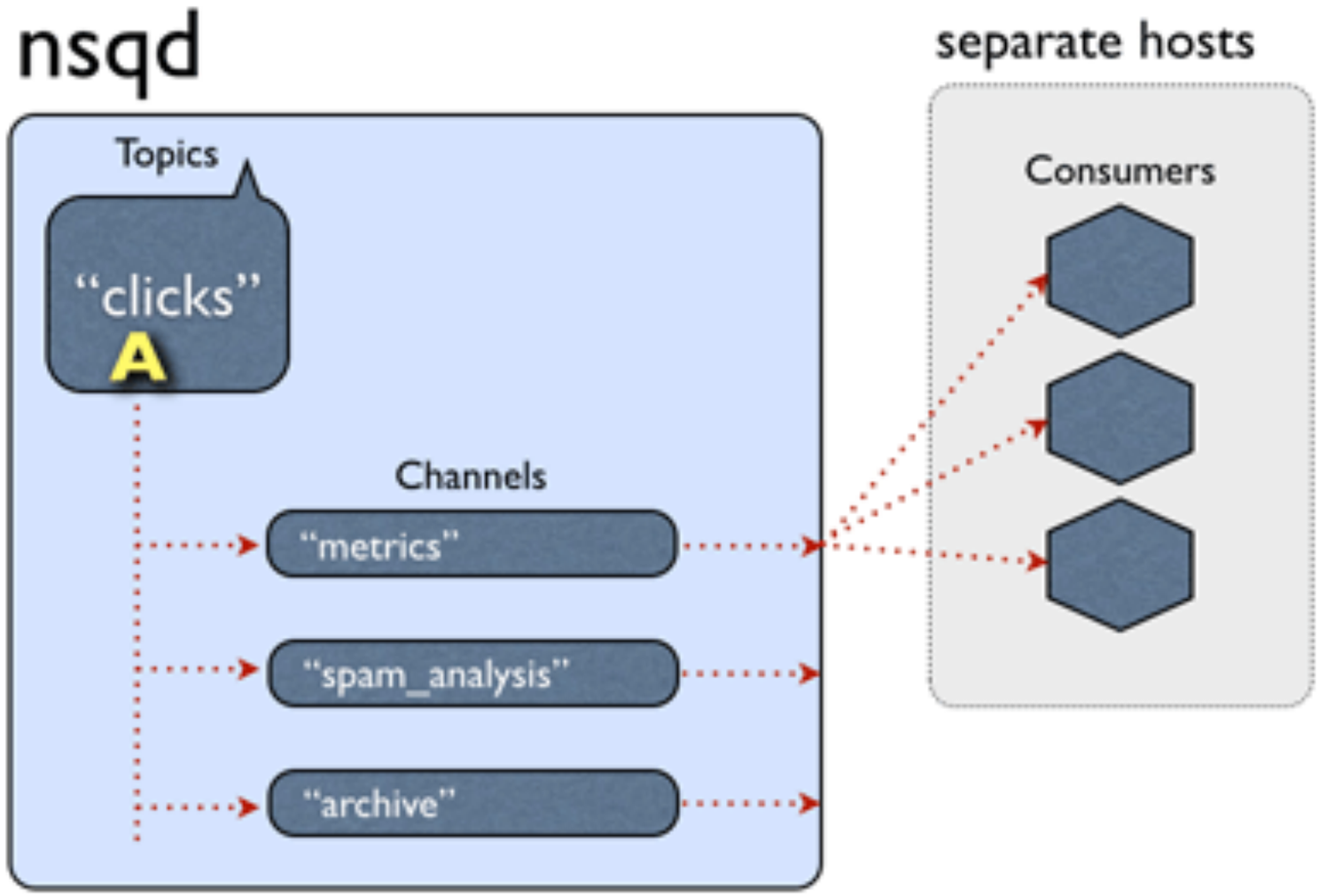
使用需求

- 数据缓冲，提高可用性，缓冲服务故障
- 数据广播，分发给多个服务
- 负载均衡，提高消费者的扩展性
- 消费反馈，确保消息不丢失
- MQ：分布式部署，排除自身单点故障
- MQ：具备横向扩展性，排除性能瓶颈

NSQ 是怎么面对这些问题的

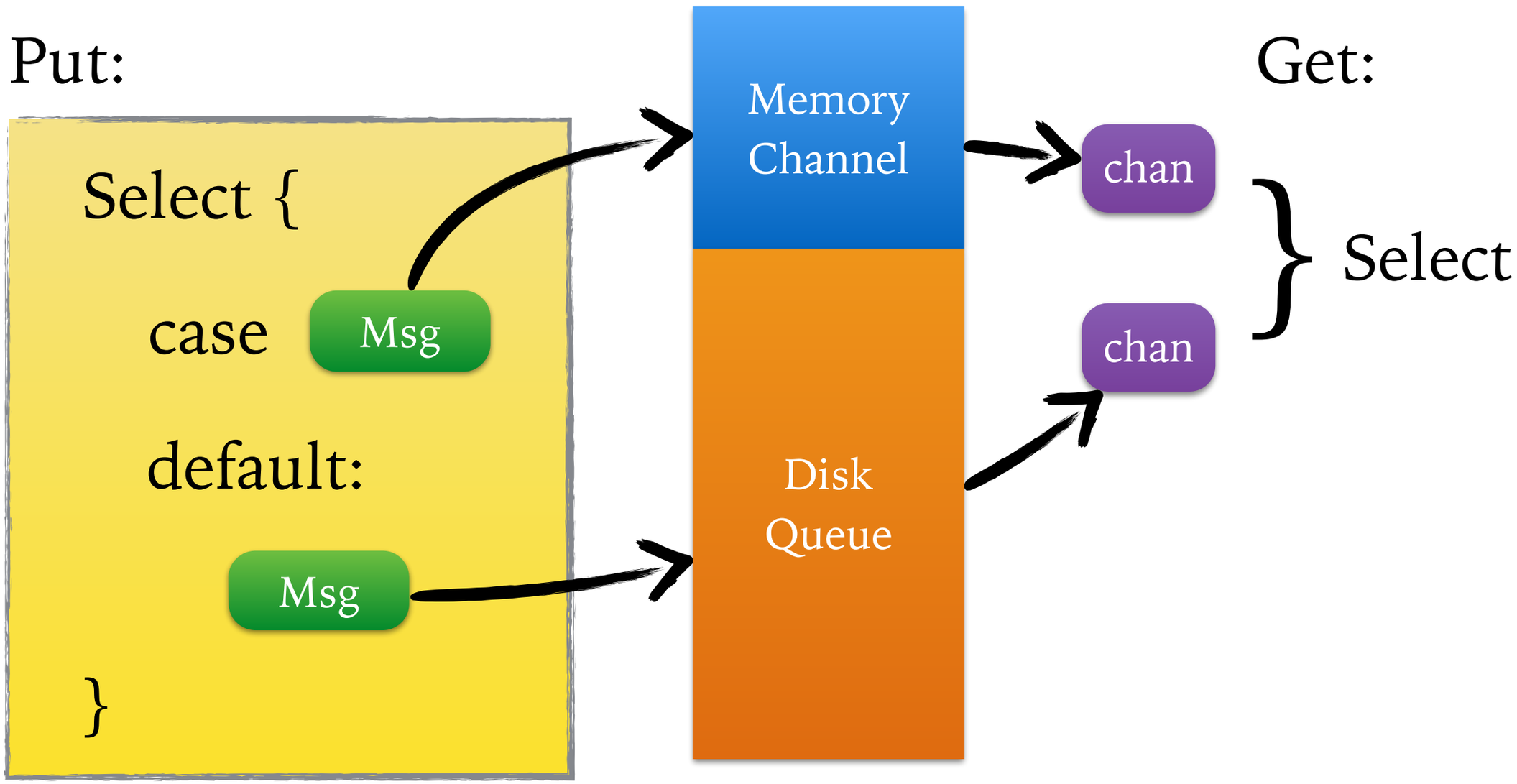
NSQ概念

- Topic
- Channel



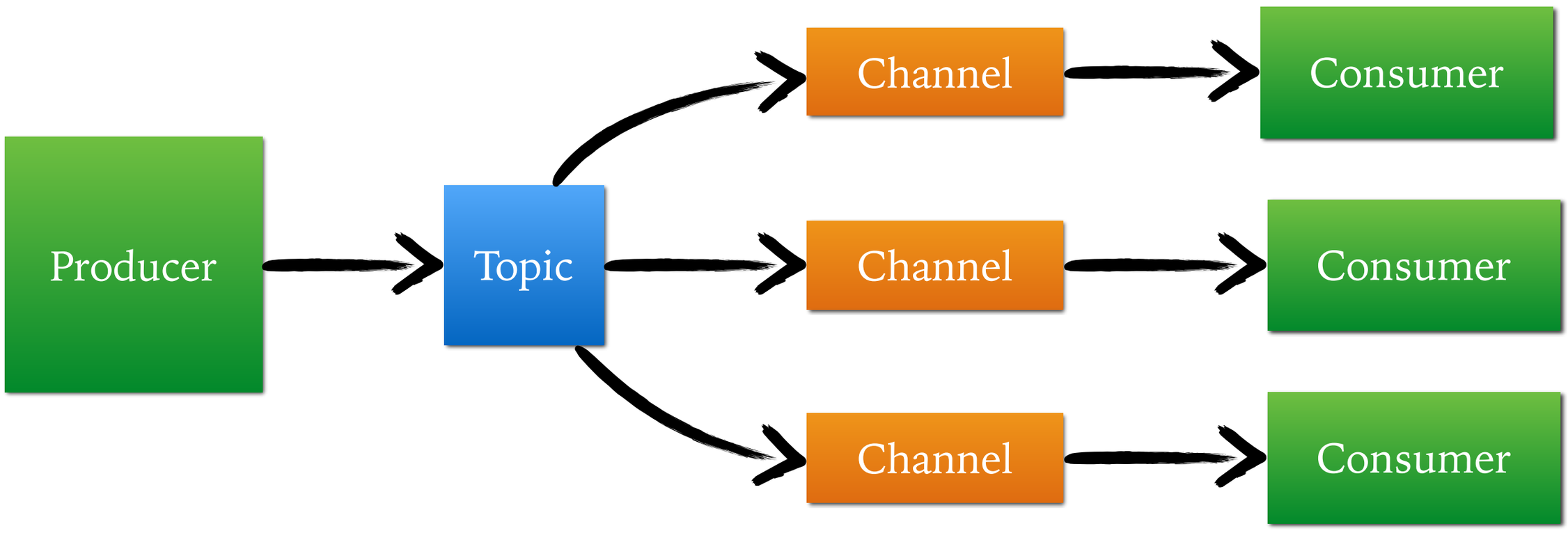
数据缓冲

- 内存队列和磁盘队列混合使用



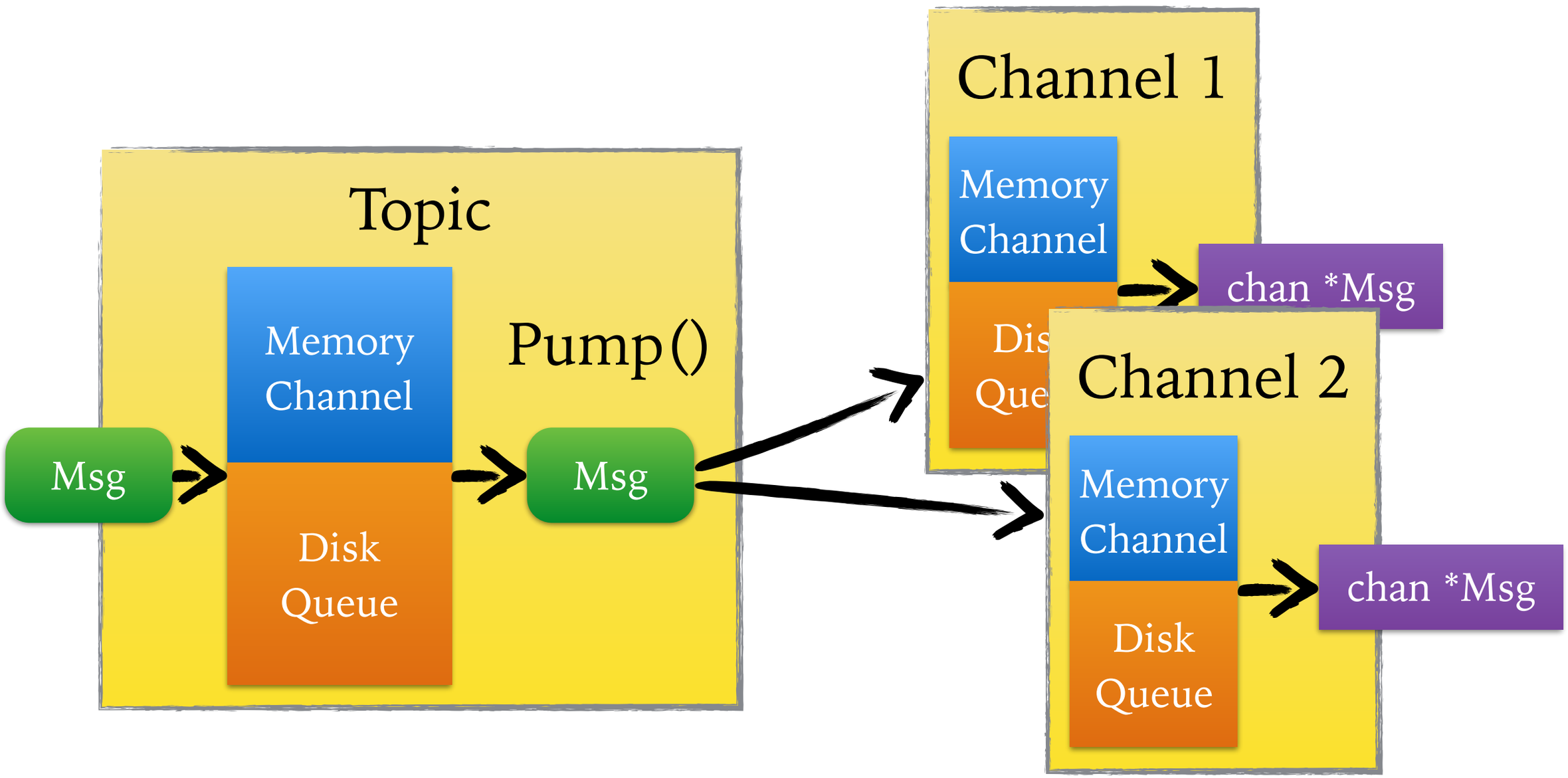
数据广播

Topic 异步把消息广播给 Channel



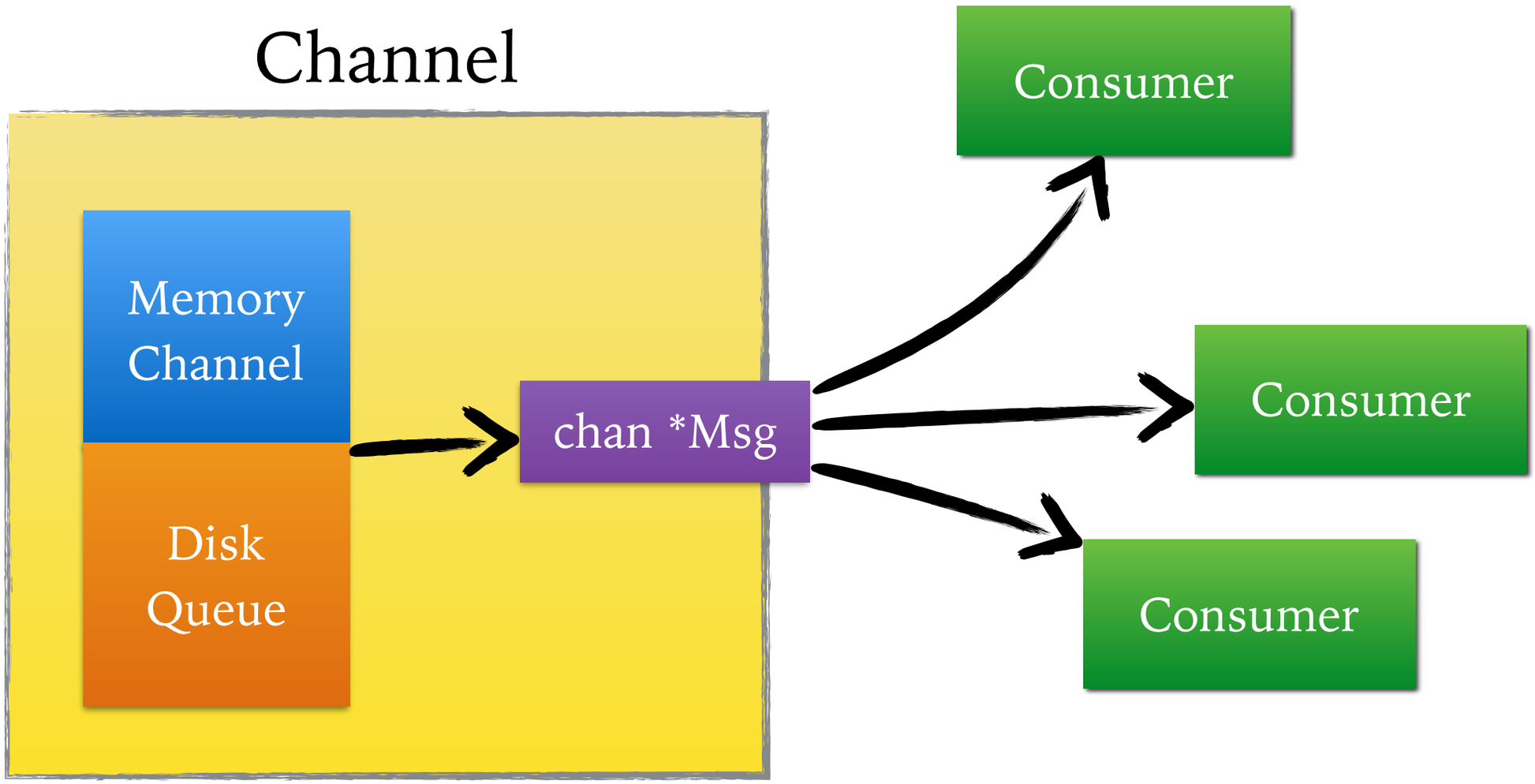
数据广播

Topic 自身拥有 buffer 的能力

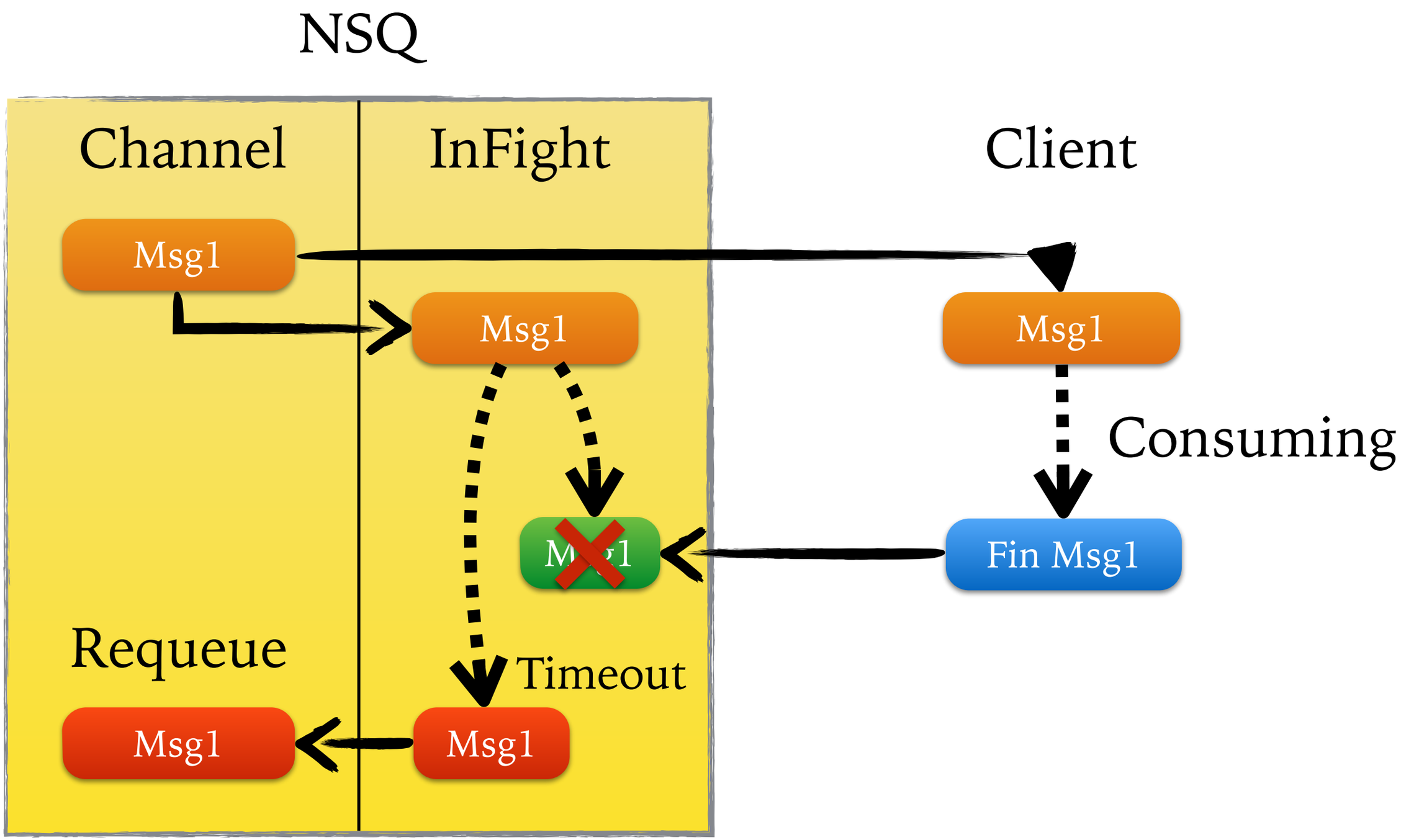


负载均衡

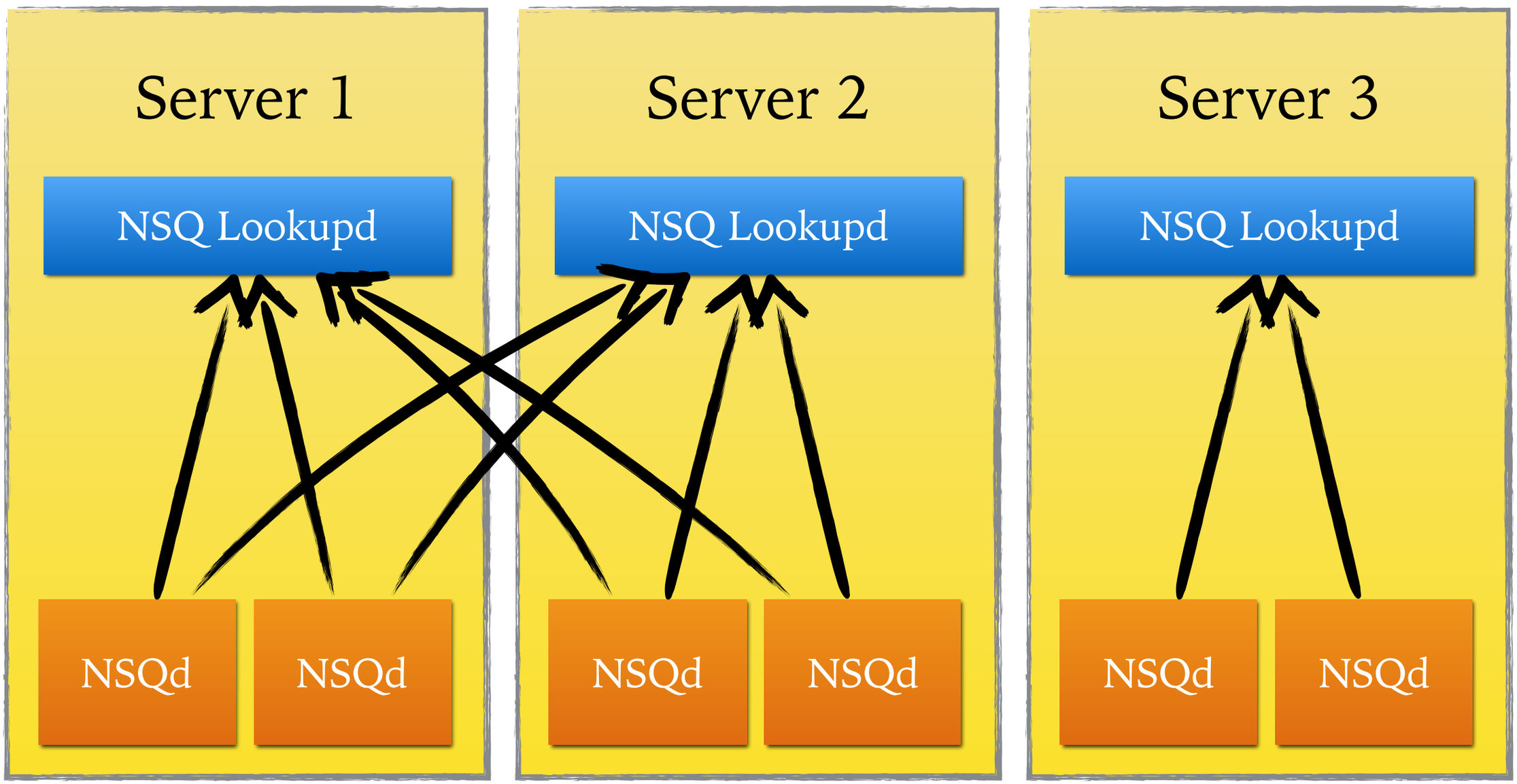
多个 Consumer 同时通过同一个 chan 读取



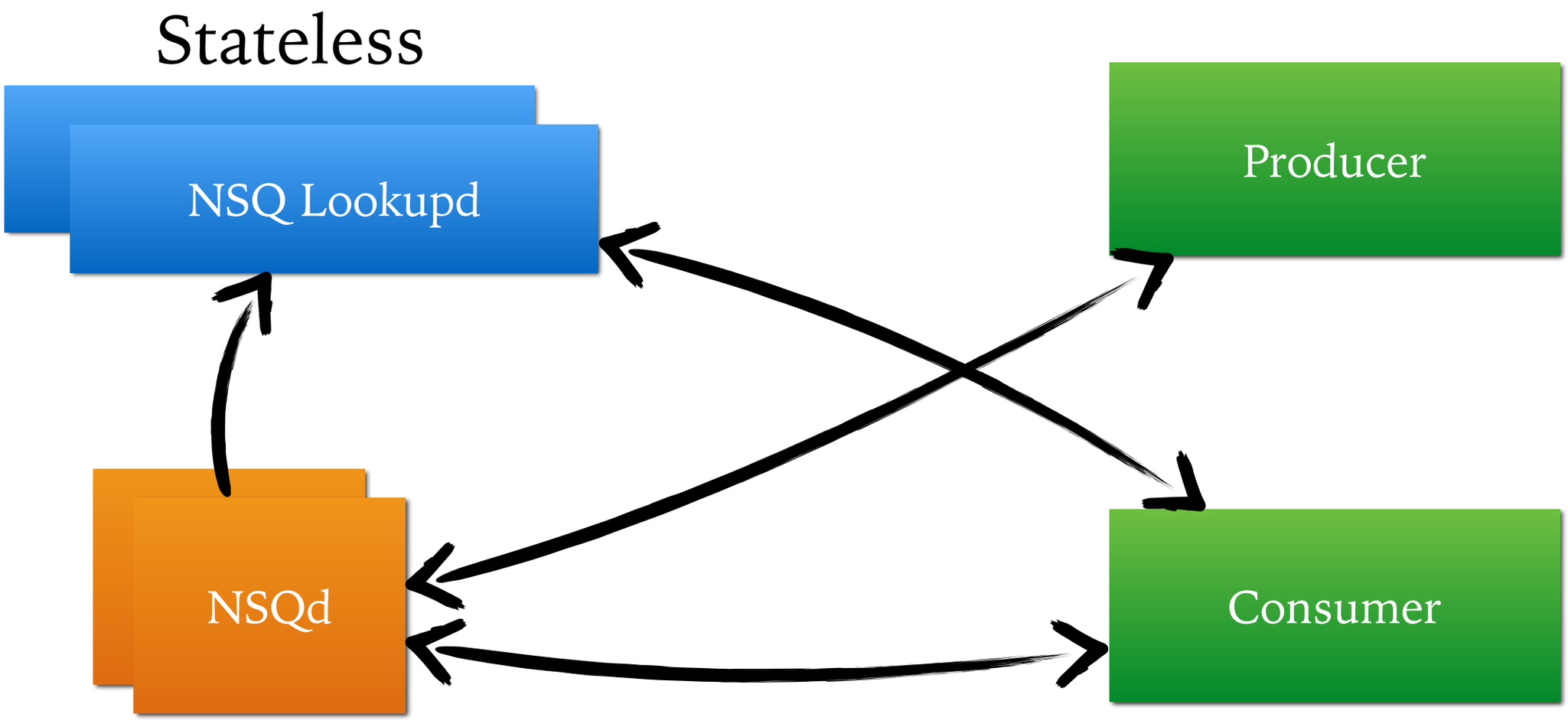
消费反馈


NSQ
Channel
InFight
Client
Msg1
Msg1
Msg1
Consuming
Fin Msg1
Requeue
Timeout
Msg1
Msg1

分布式架构


Server 1
NSQ Lookupd
NSQd
NSQd
Server 2
NSQ Lookupd
NSQd
NSQd
Server 3
NSQ Lookupd
NSQd
NSQd

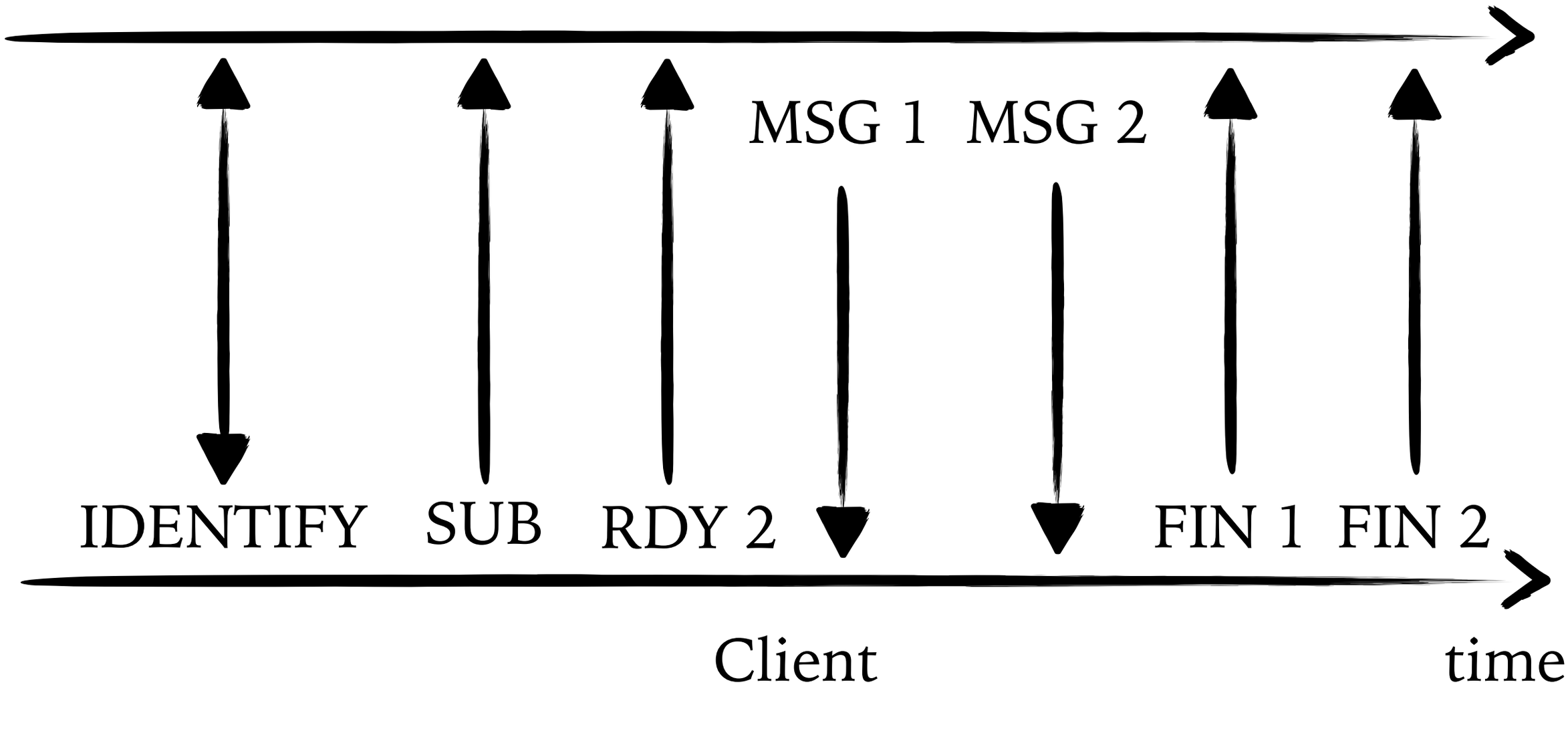
分布式架构



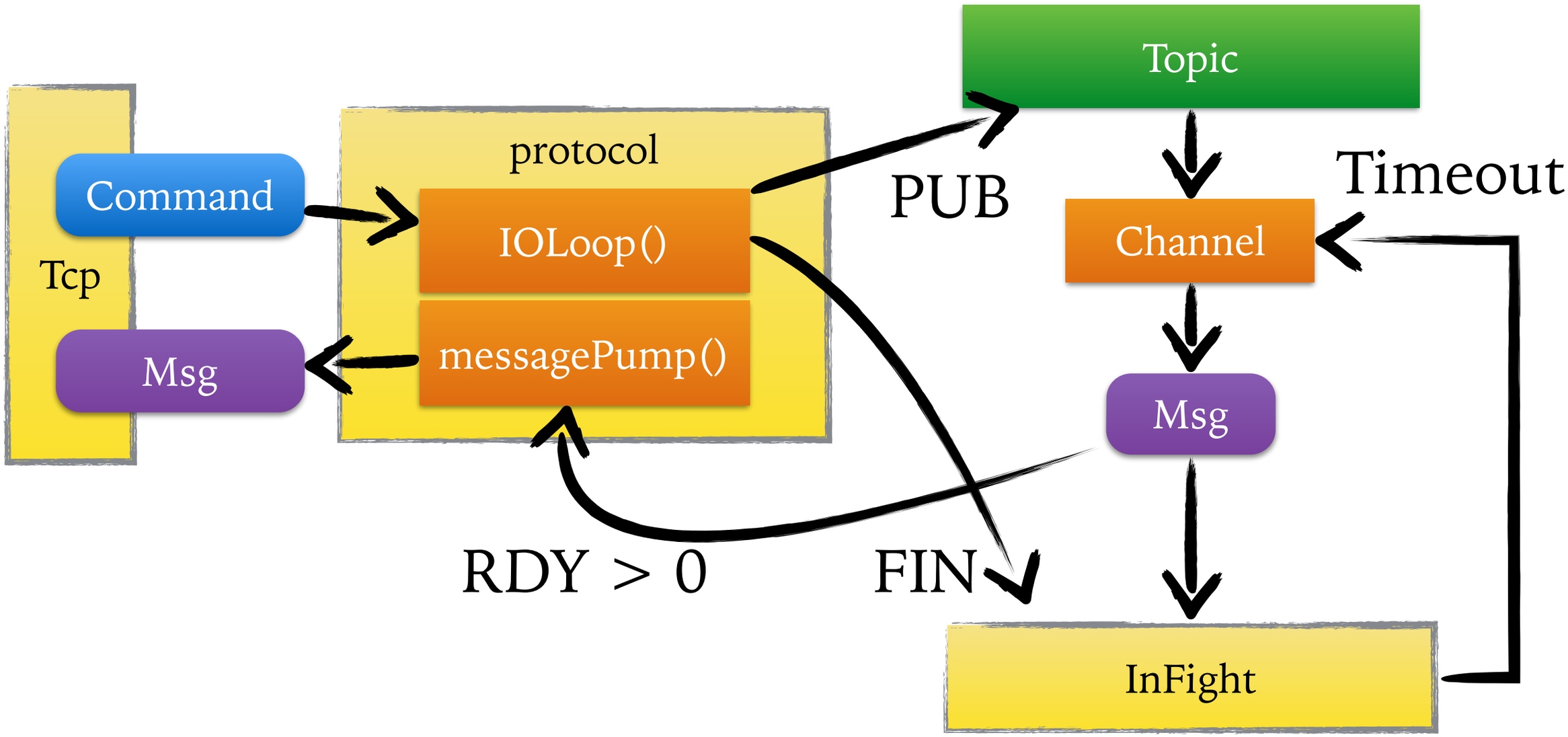
内部逻辑

协议解析

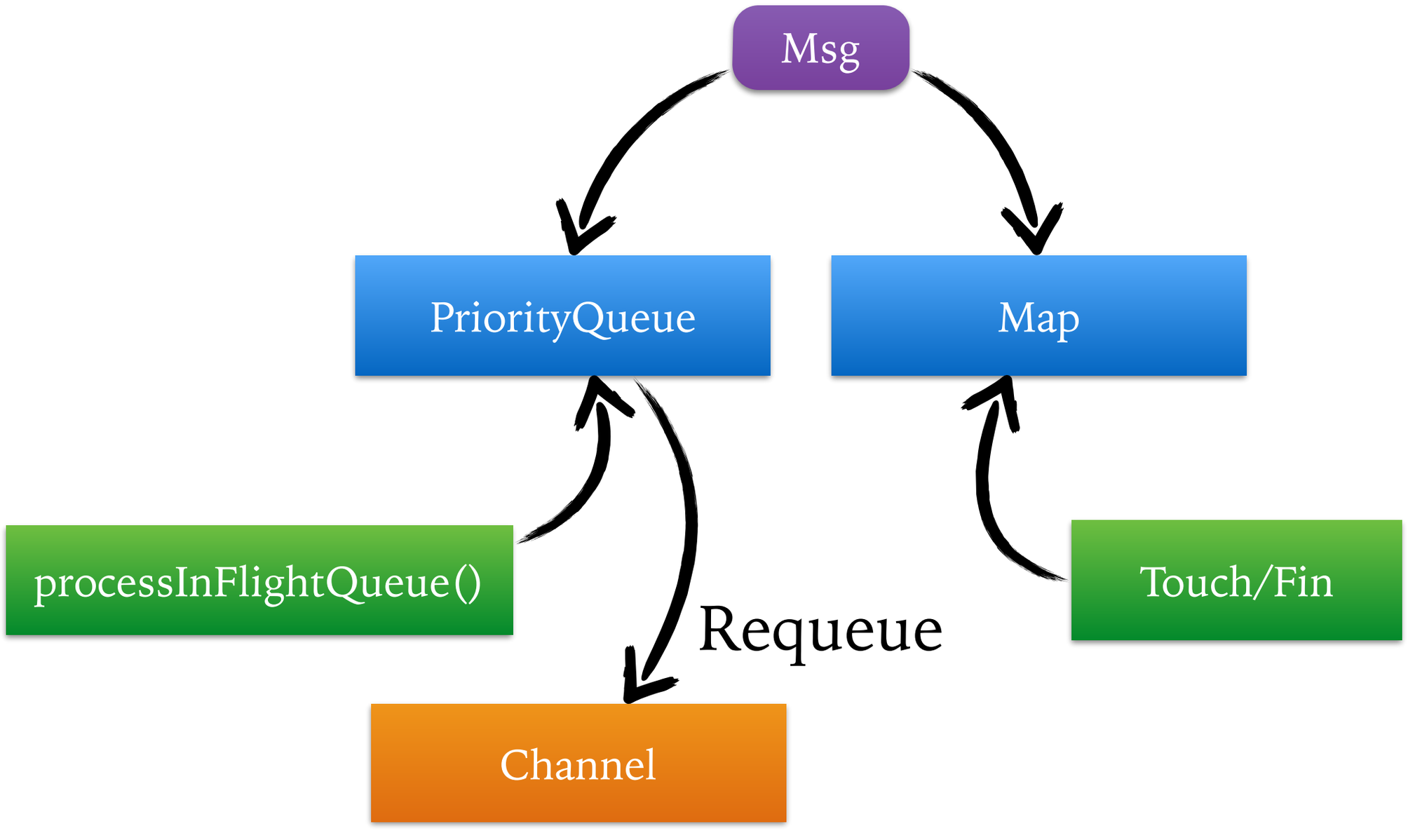
消费消息



消息架构


Topic
protocol
Command
IOLoop()
PUB
Timeout
Channel
Tcp
Msg
messagePump()
Msg
RDY > 0
FIN
InFight

InFight


Msg
PriorityQueue
Map
processInFlightQueue()
Requeue
Touch/Fin
Channel

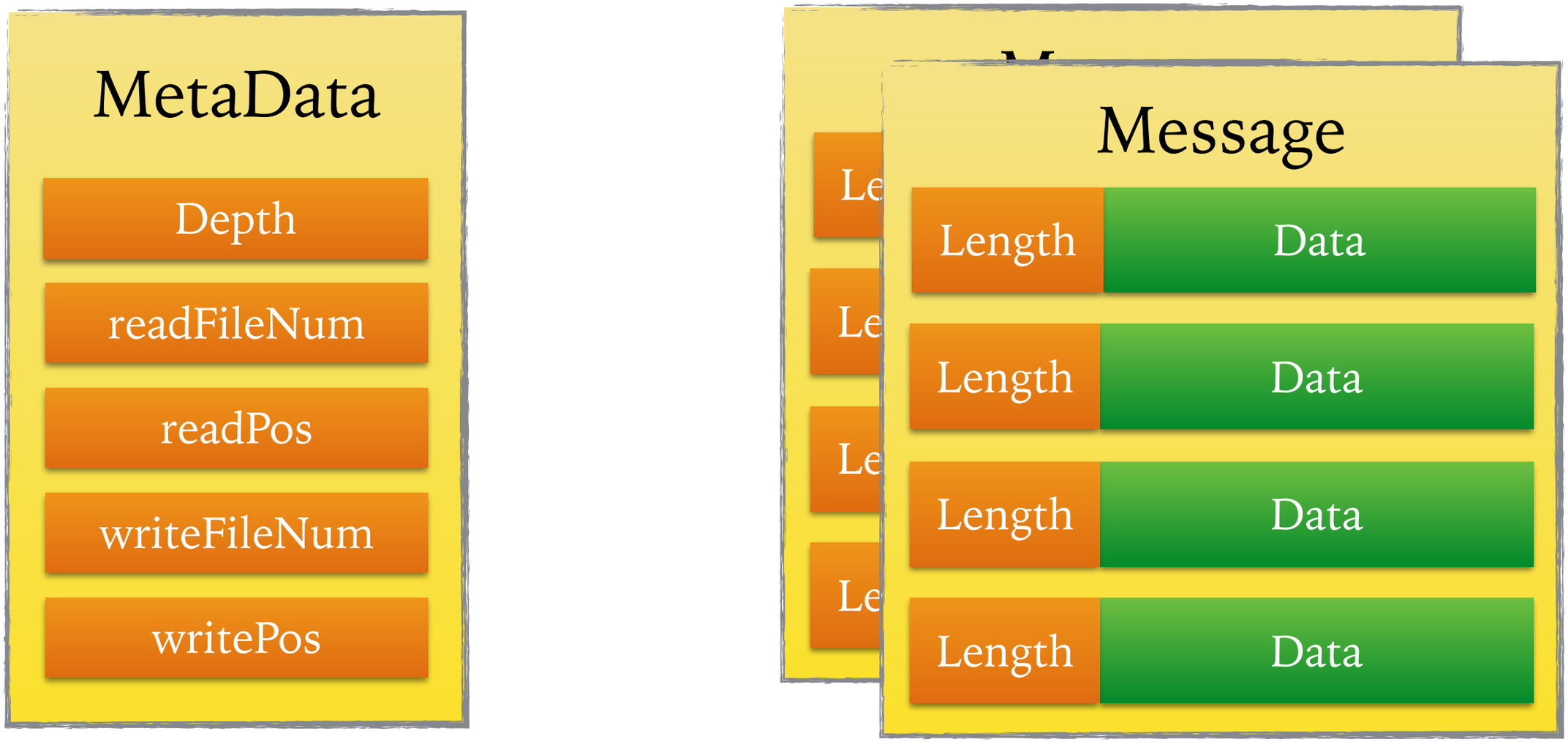
DiskQueue

接口

```
type BackendQueue interface {
    Put([]byte) error
    ReadChan() chan []byte
    Close() error
    Delete() error
    Depth() int64
    Empty() error
}
```

DiskQueue

磁盘结构

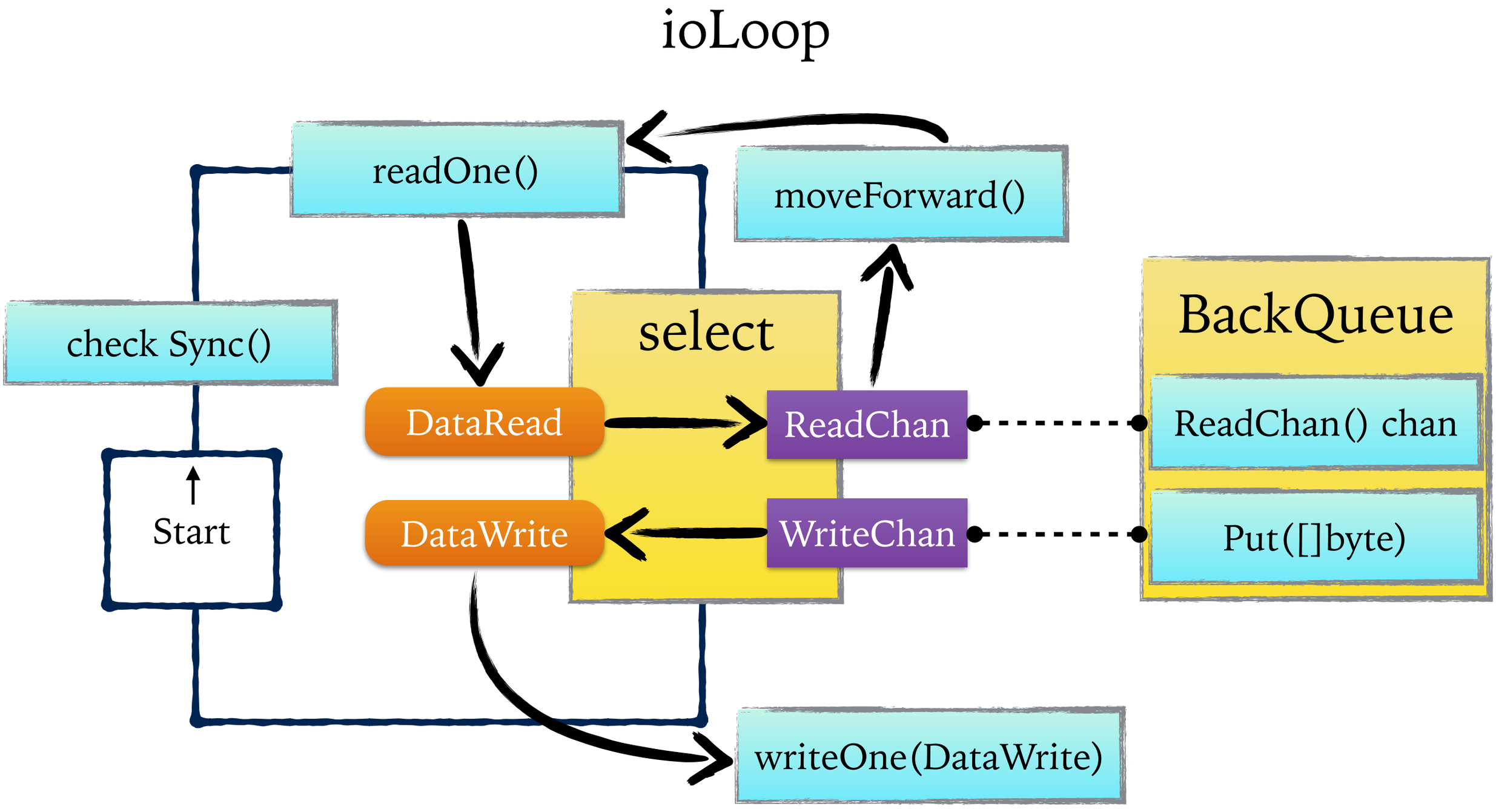


DiskQueue

磁盘多个数据文件的作用

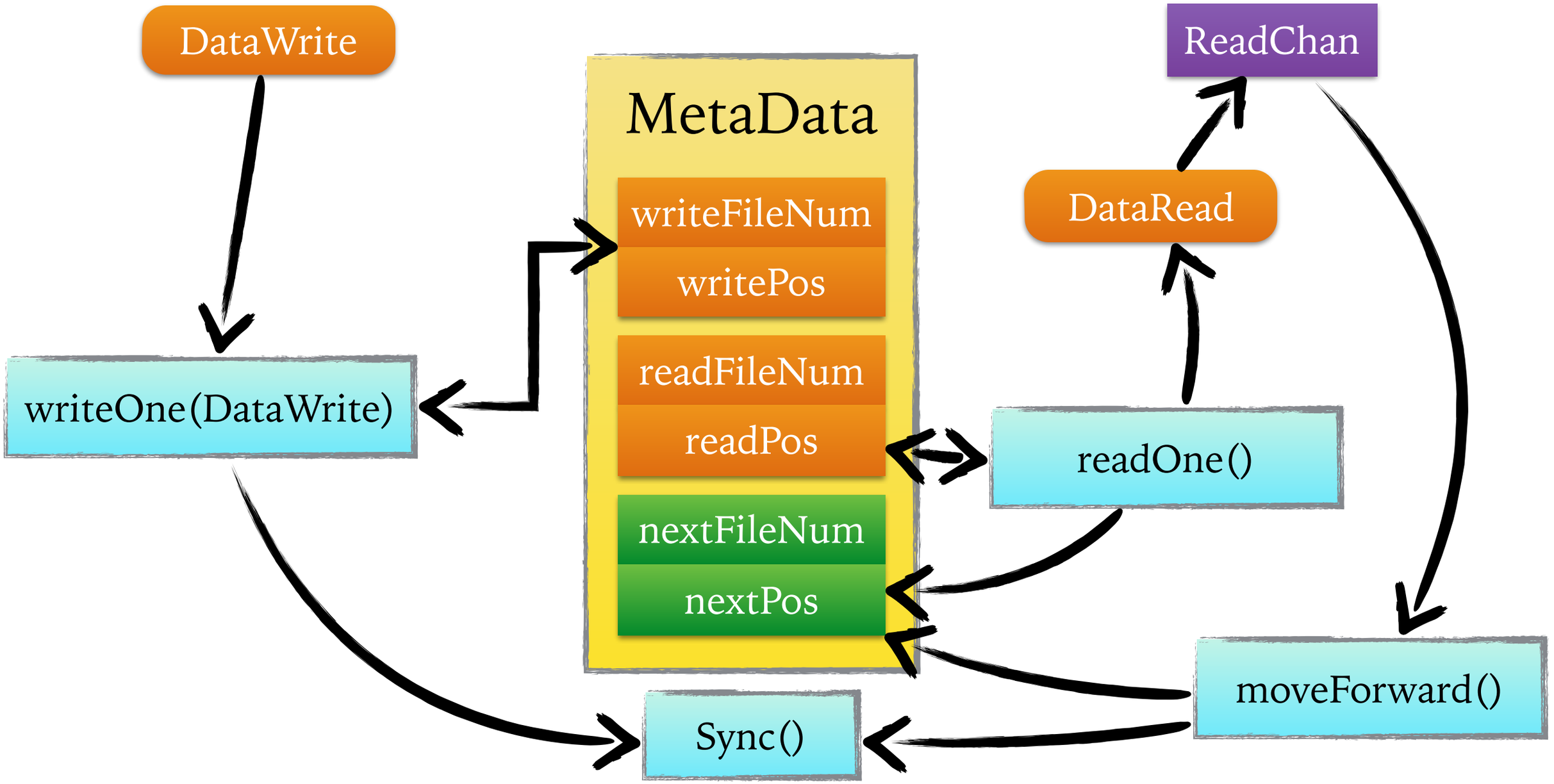
- 过期磁盘文件清理的最小单位
- 错误异常忽略的最小单位
- 文件过大会造成inode读取效率差

DiskQueue


ioLoop
readOne()
moveForward()
check Sync()
select
BackQueue
DataRead
ReadChan
ReadChan() chan
Start
DataWrite
WriteChan
Put([]byte)
writeOne(DataWrite)

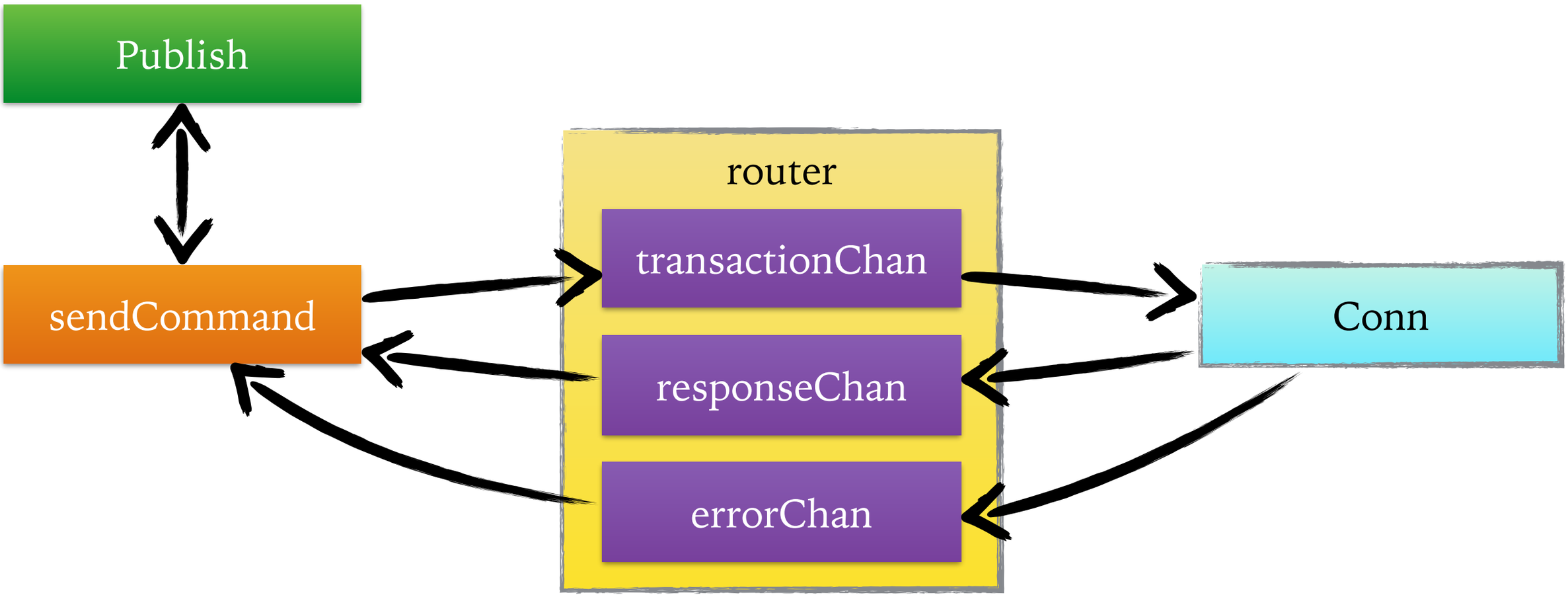
DiskQueue

Write/Read

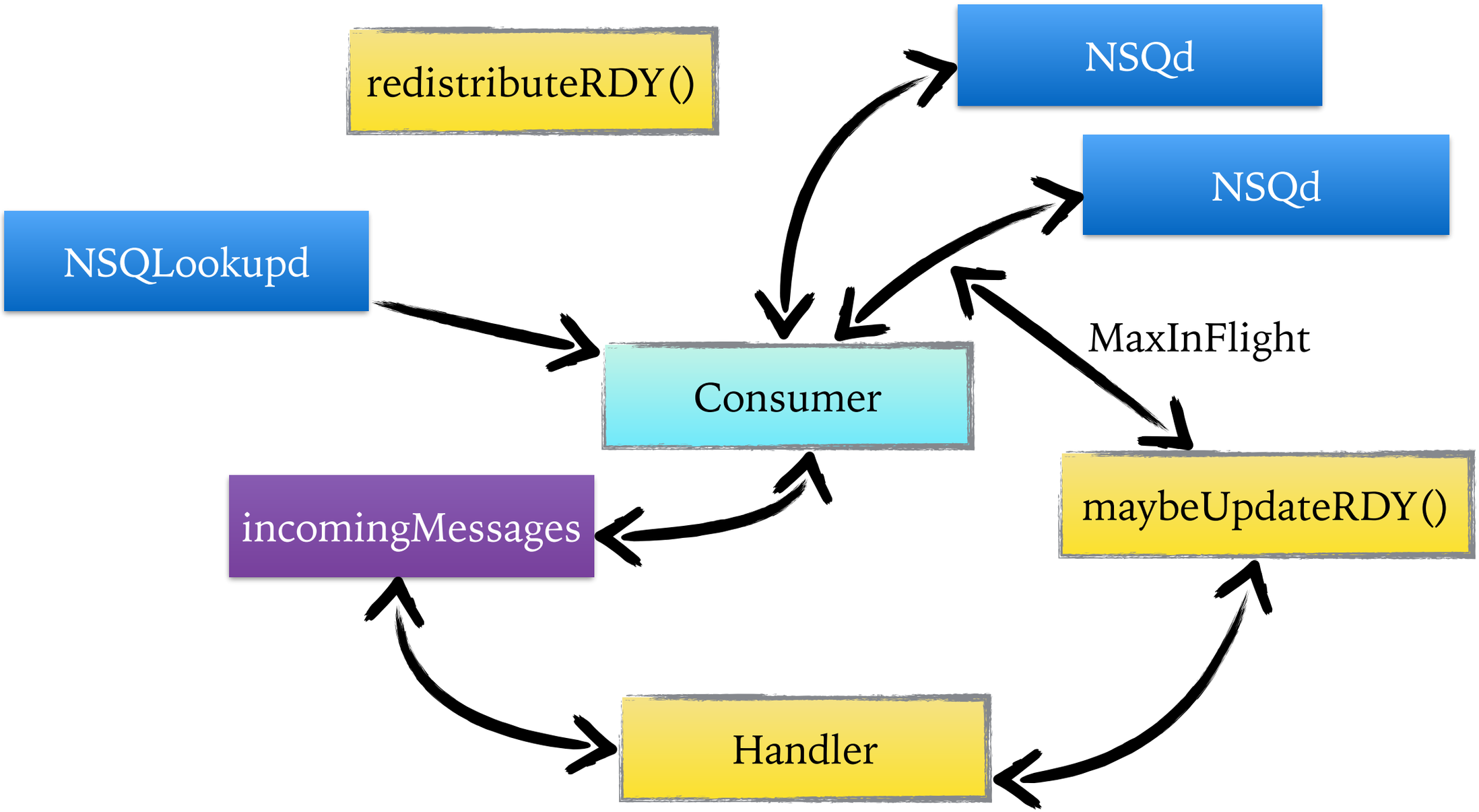


SDK

Producer


Publish
sendCommand
router
transactionChan
responseChan
errorChan
Conn

Consumer


redistributeRDY()
NSQd
NSQd
NSQLookupd
MaxInFlight
Consumer
maybeUpdateRDY()
incomingMessages
Handler

总结

- 将复杂度集中在服务端，客户端易用
- 数据无冗余
- 数据广播空间复杂度是 O(N)
- 内存队列与磁盘队列混用，时序性无法预估
- Topic -> Channel 消耗太高
- 大量Fin包导致系统调用/小数据包过多
- Producer 没有官方分布式解决方案

Q&A