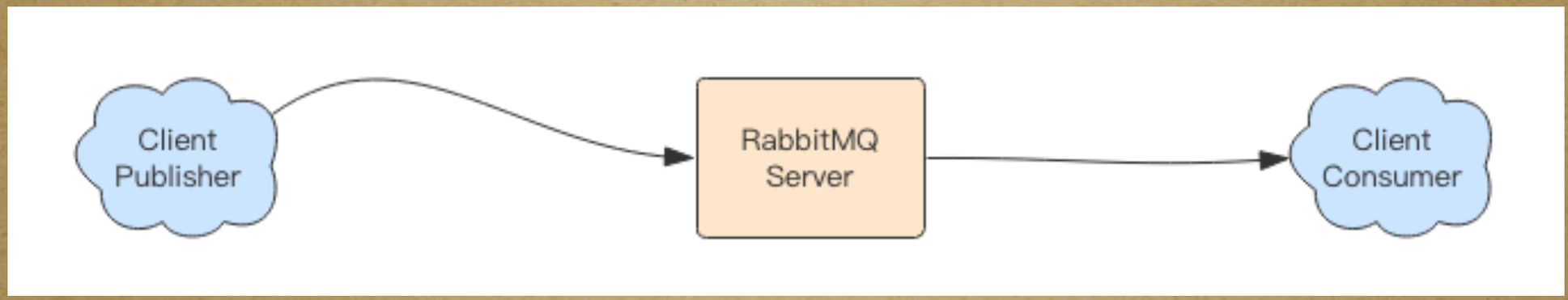
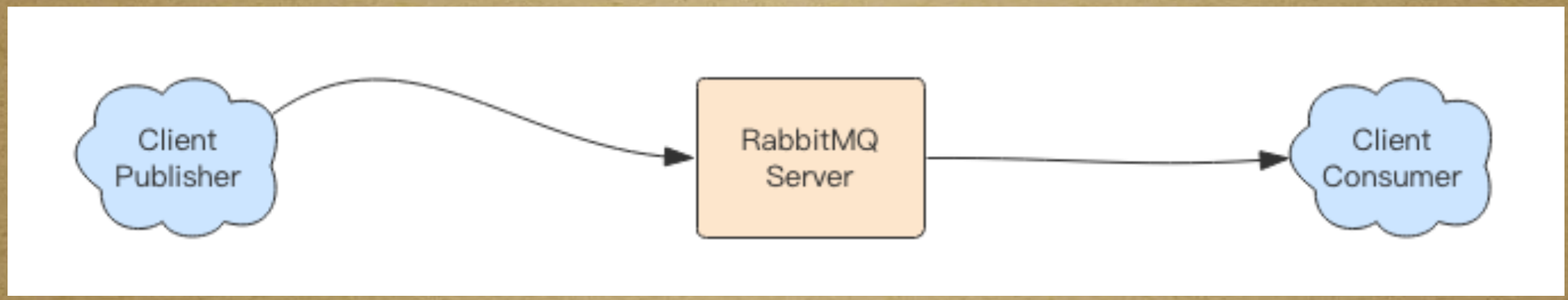
RabbitMQ 特性

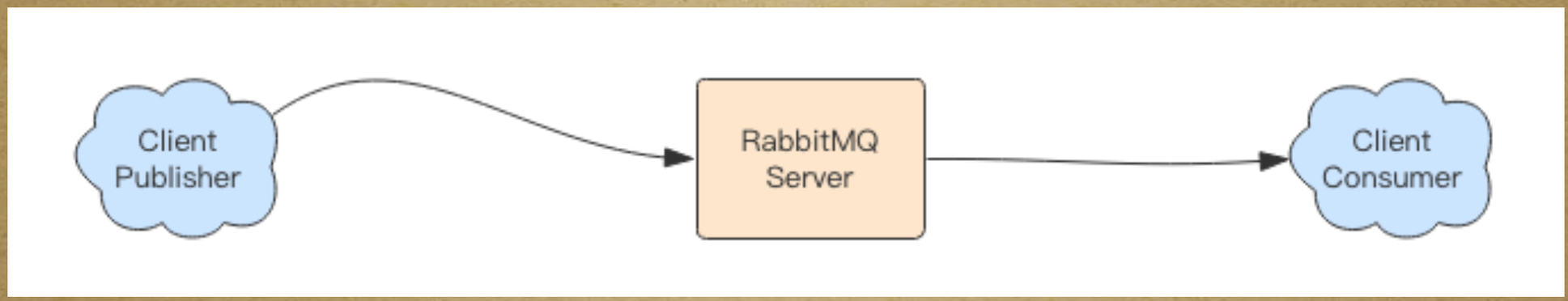


RabbitMQ 特性



1.消息的接受者（消费者）

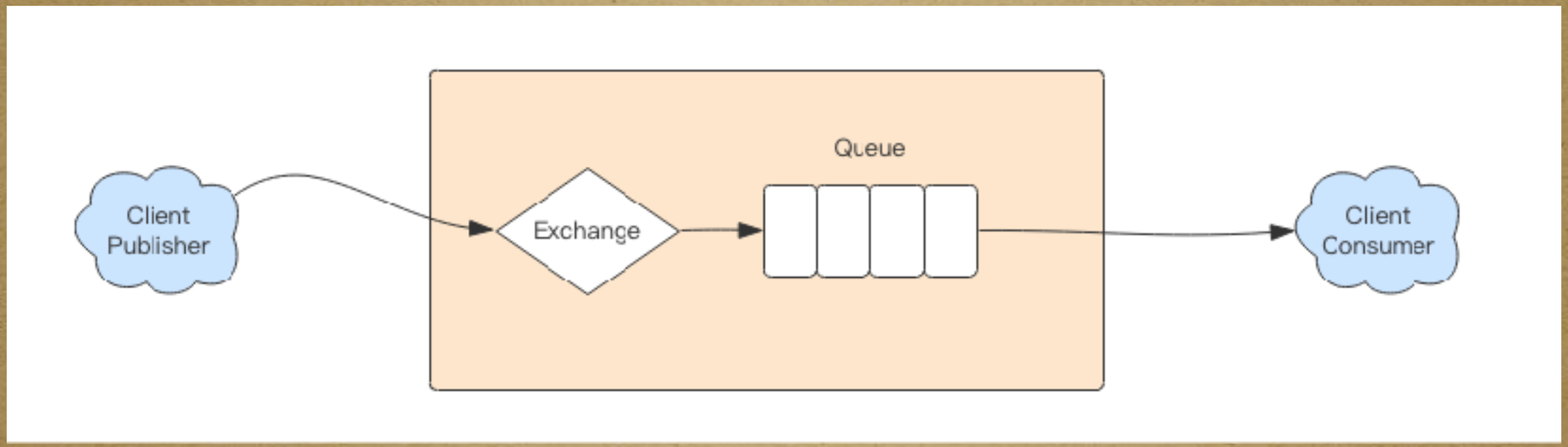
RabbitMQ 特性



1.消息的接受者（消费者）

2. 消息的发送者（生产者）

RabbitMQ 特性

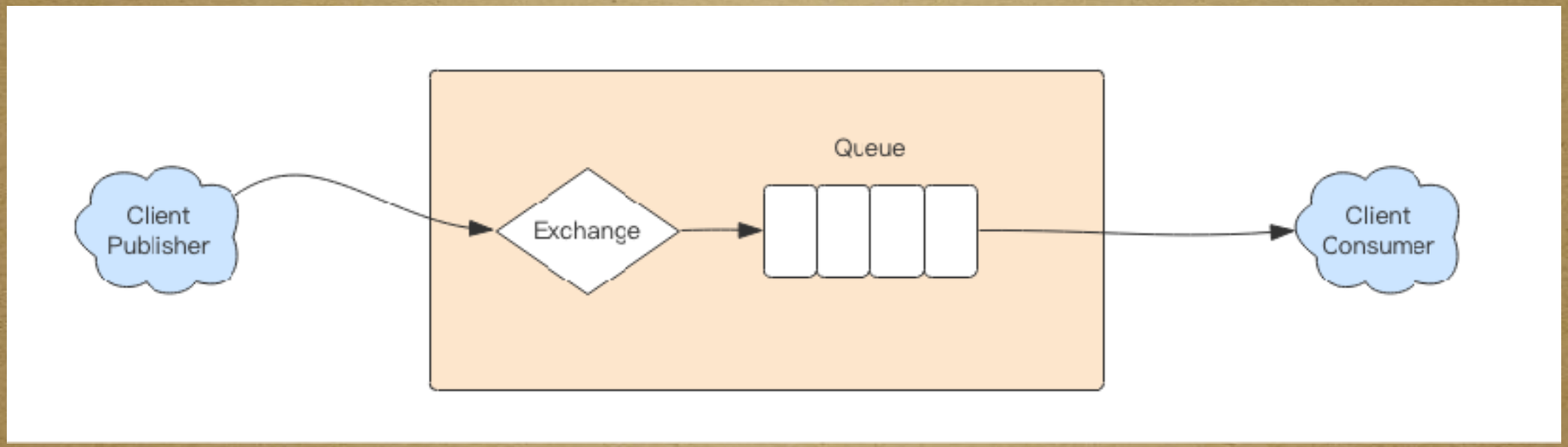


1.消息的接受者（消费者）

2. 消息的发送者（生产者）

3. 消息的仓储（持久化）

RabbitMQ 特性



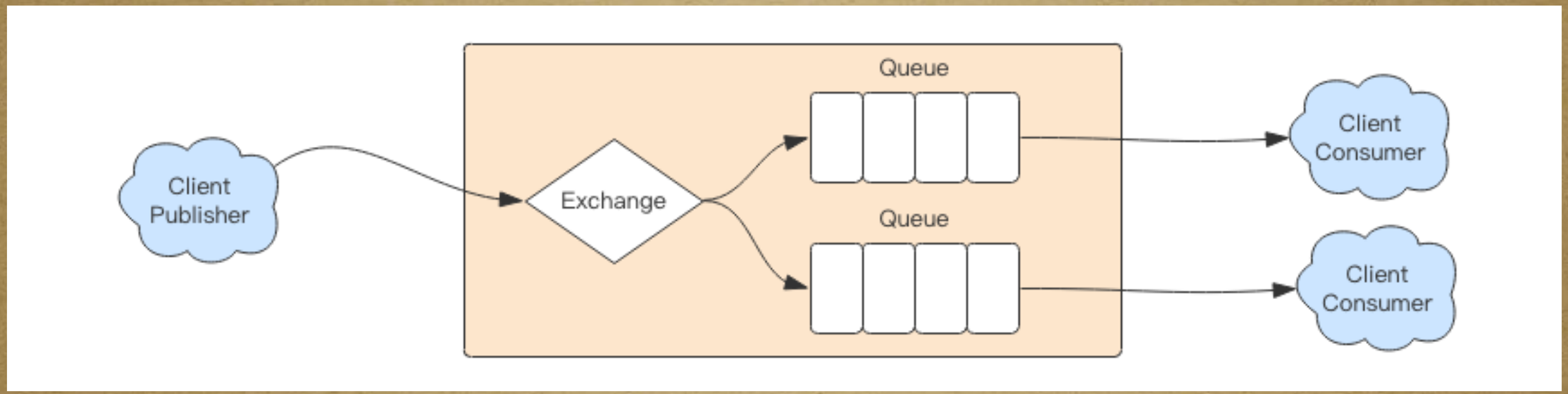
1.消息的接受者（消费者）

2. 消息的发送者（生产者）

3. 消息的仓储（持久化）

4. 延后传递（堆积）

RabbitMQ 特性



1.消息的接受者（消费者）

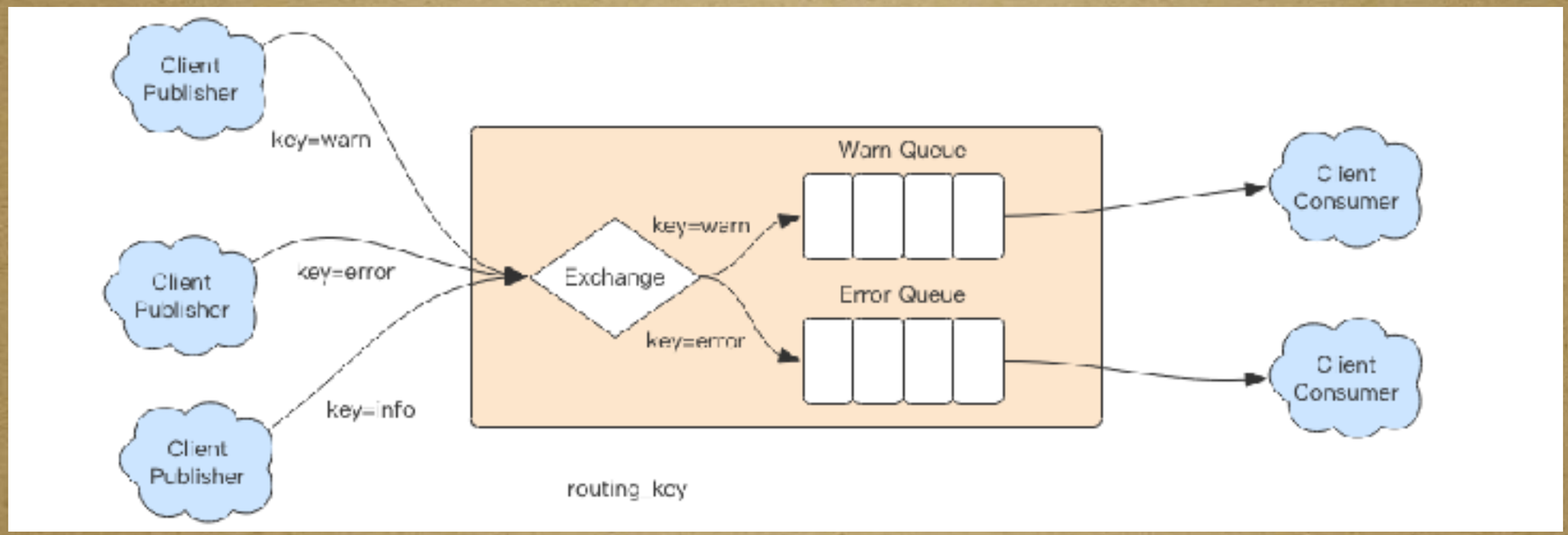
2. 消息的发送者（生产者）

3. 消息的仓储（持久化）

4. 延后传递（堆积）

5. 复制（广播）

RabbitMQ 特性



1.消息的接受者（消费者）

2. 消息的发送者（生产者）

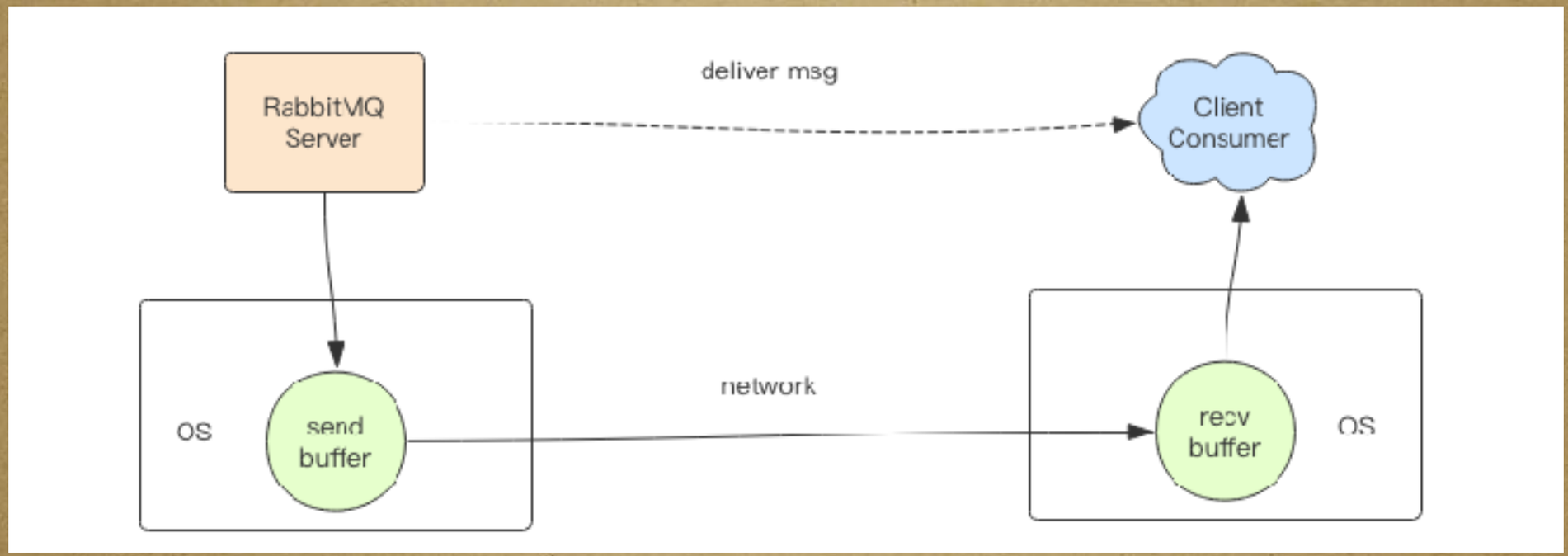
3. 消息的仓储（持久化）

4. 延后传递（堆积）

5. 复制（广播）

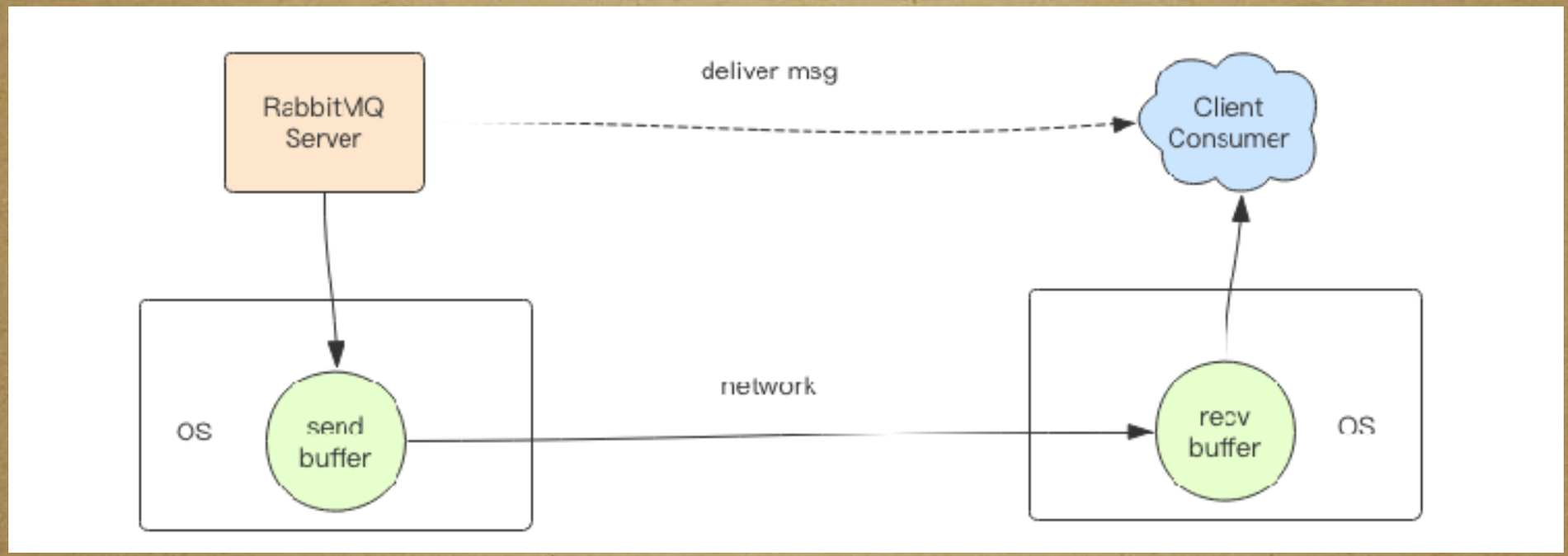
6. 分炼（分类路由）

消息不可靠



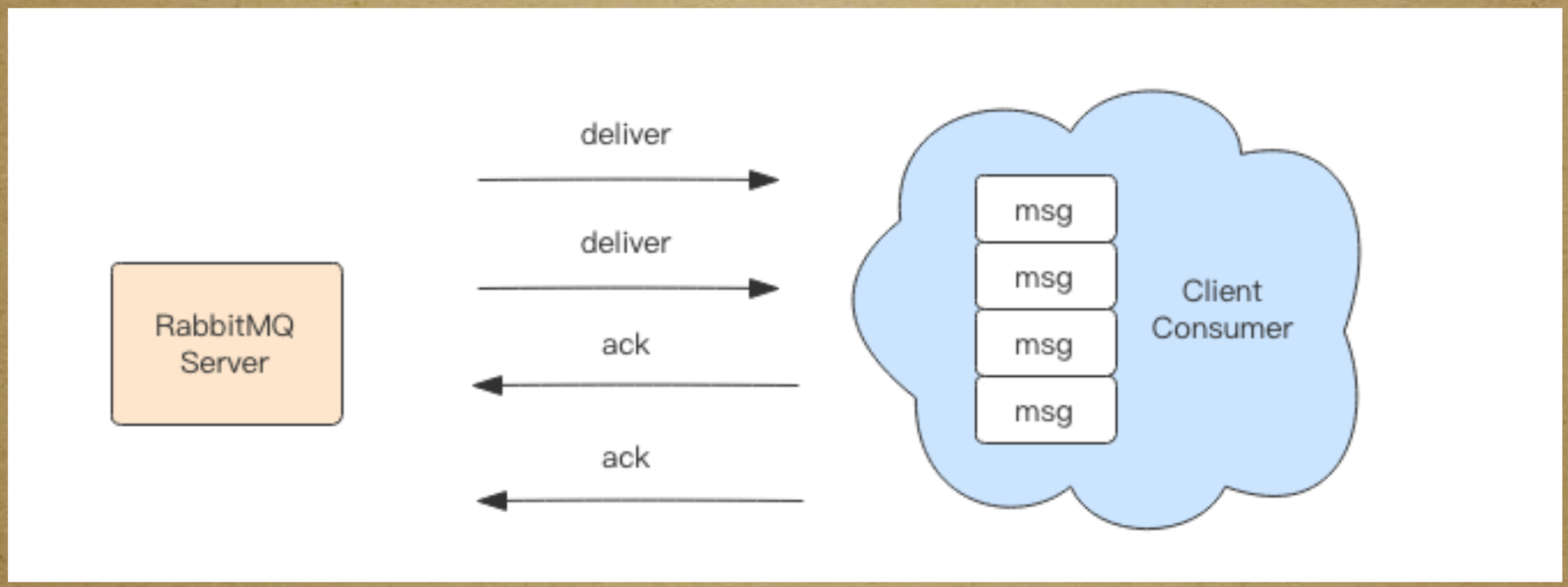
socket.write(msg)

消息不可靠



1. 网络故障
2. 宕机
2. kill -9

消息可靠



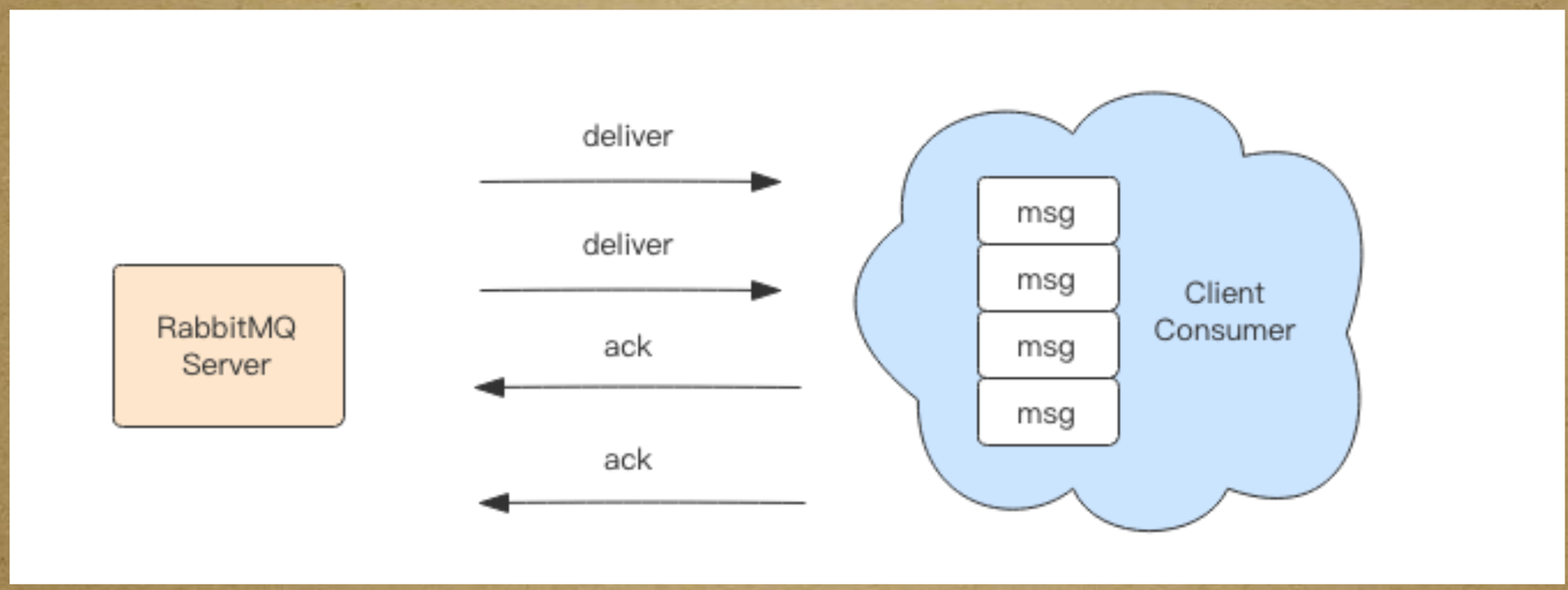
1.server deliver后不删除

2. client 收到消息处理后回复ack

3. server 收到ack后删除消息

4.没收到ack消息(关闭后)重新投递

消息可靠



1.server deliver后不删除

2. client 收到消息处理后回复ack

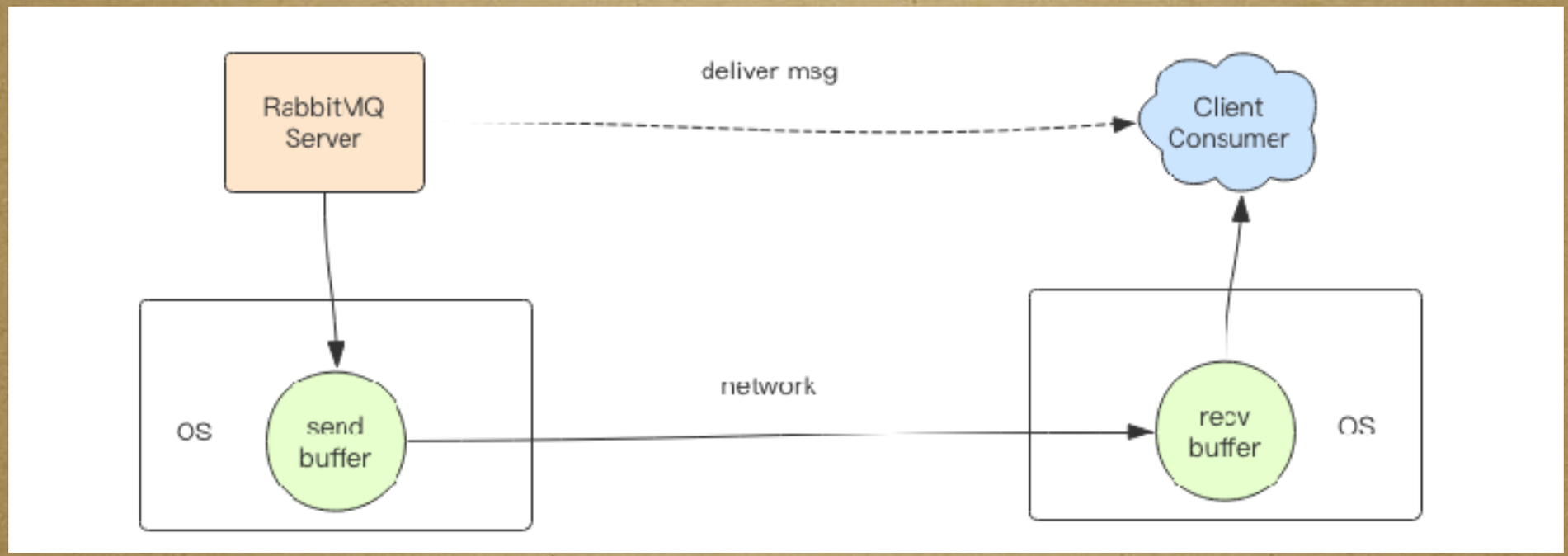
3. server 收到ack后删除消息

4.没收到ack消息(关闭后)重新投递

5.ack丢失会导致消息重复处理

6.去重(幂等)由业务系统自己考虑

Auto Ack vs Manual Ack



1.AutoAck 投后即删(no ack)

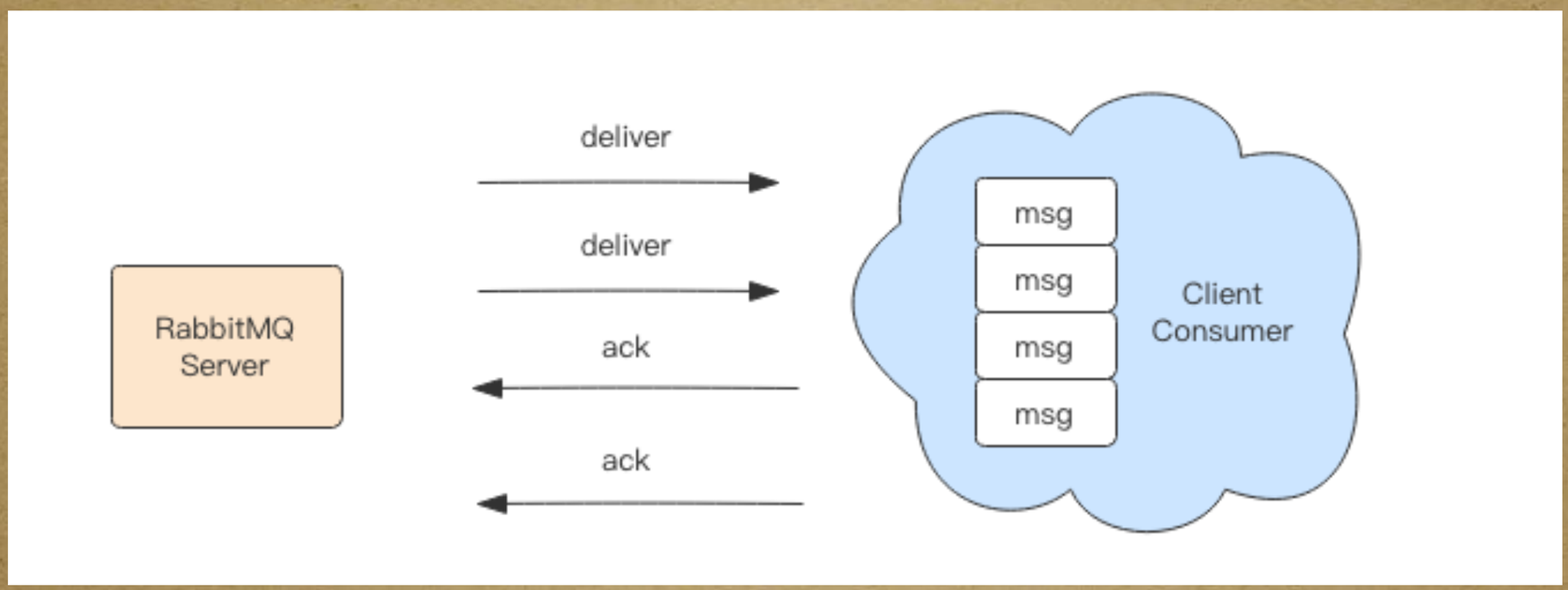
2. 网络故障，消息丢失

3.消费慢，投递风暴

4.缓冲区堆积，Server 写不动

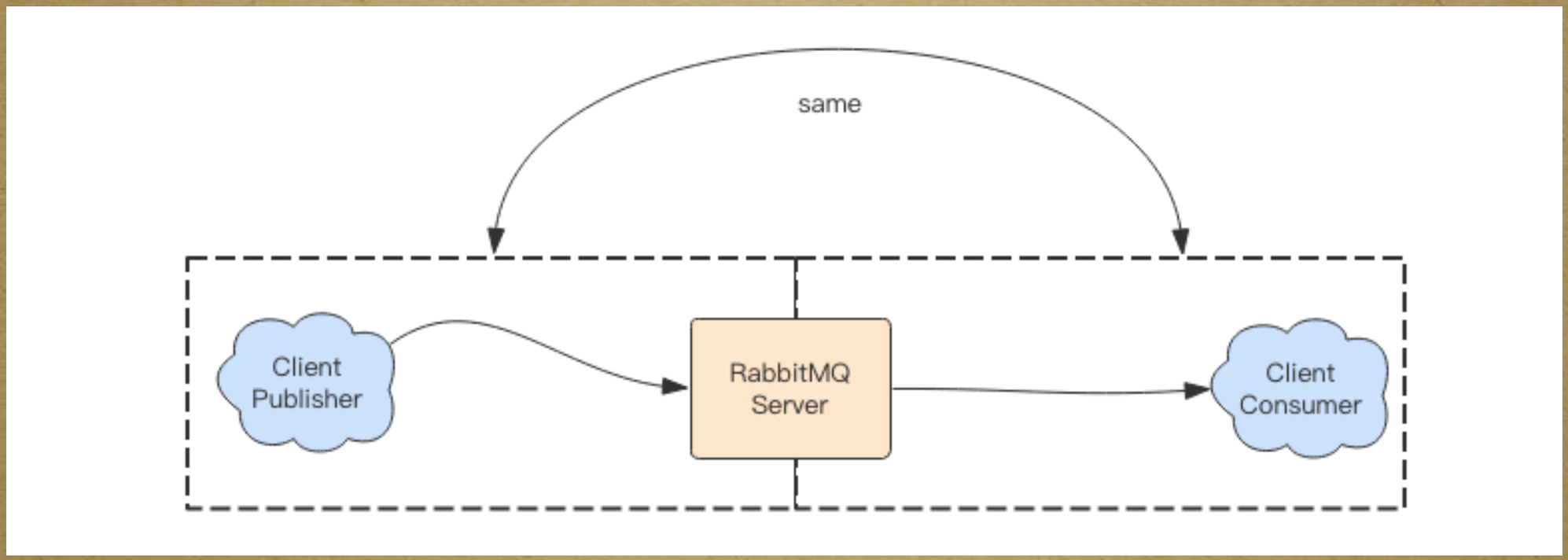
5.链接被Server强制关闭

Auto Ack vs Manual Ack



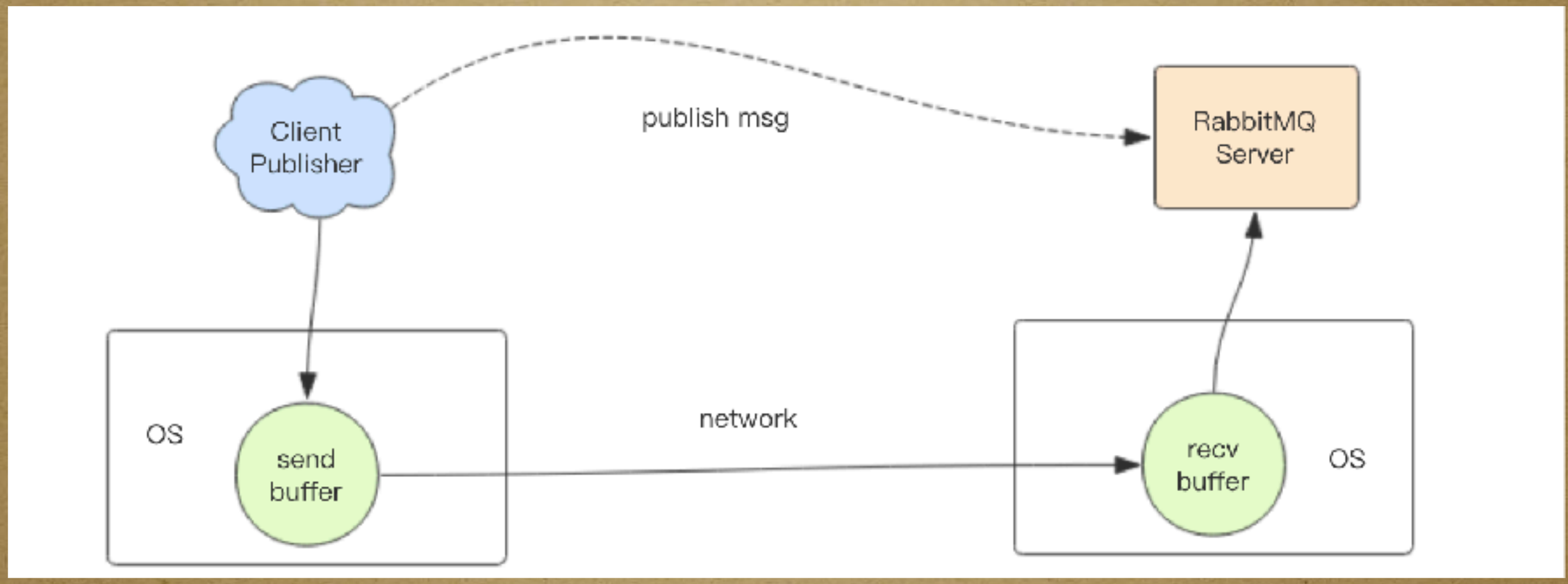
1. Manual Ack 善解人意(照顾客户端)
2. 客户端PrefetchCount 能力参数
3. Deliver 有限个消息
4. Ack 一个，Deliver 一个

生产者消息可靠性



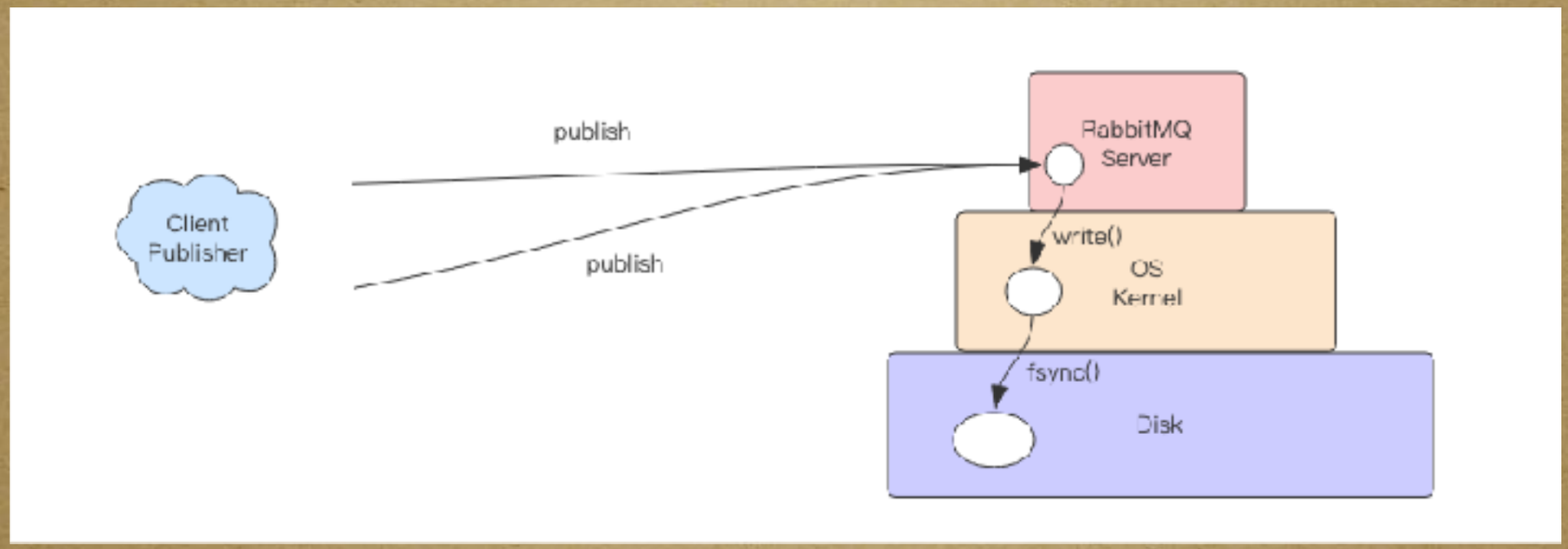
消费端的问题生产端也会有

生产者消息可靠性



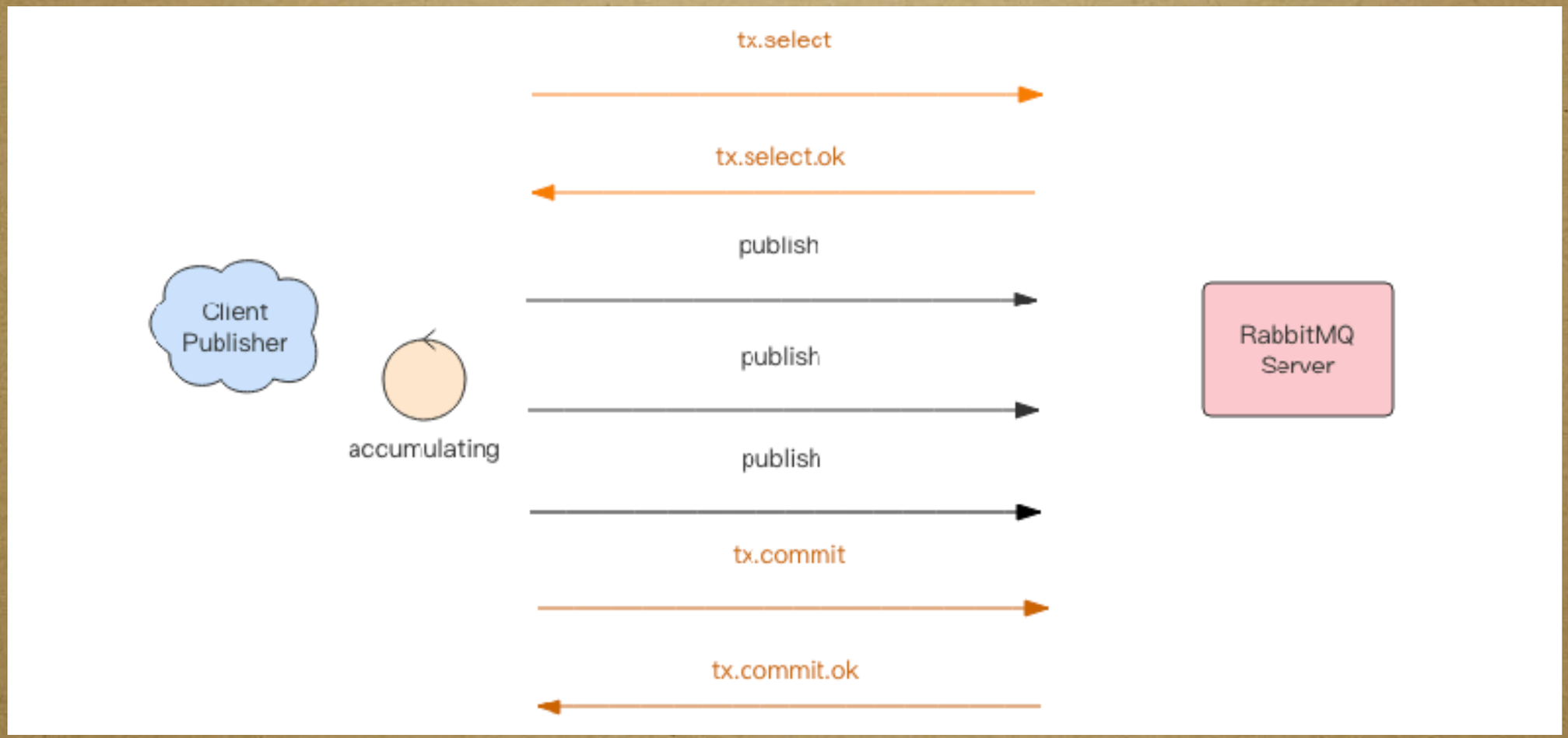
生产的消息也会丢

生产者消息可靠性



1. fsync 几百毫秒一次
2. Server 宕机
3. Redis AOF

生产者事务

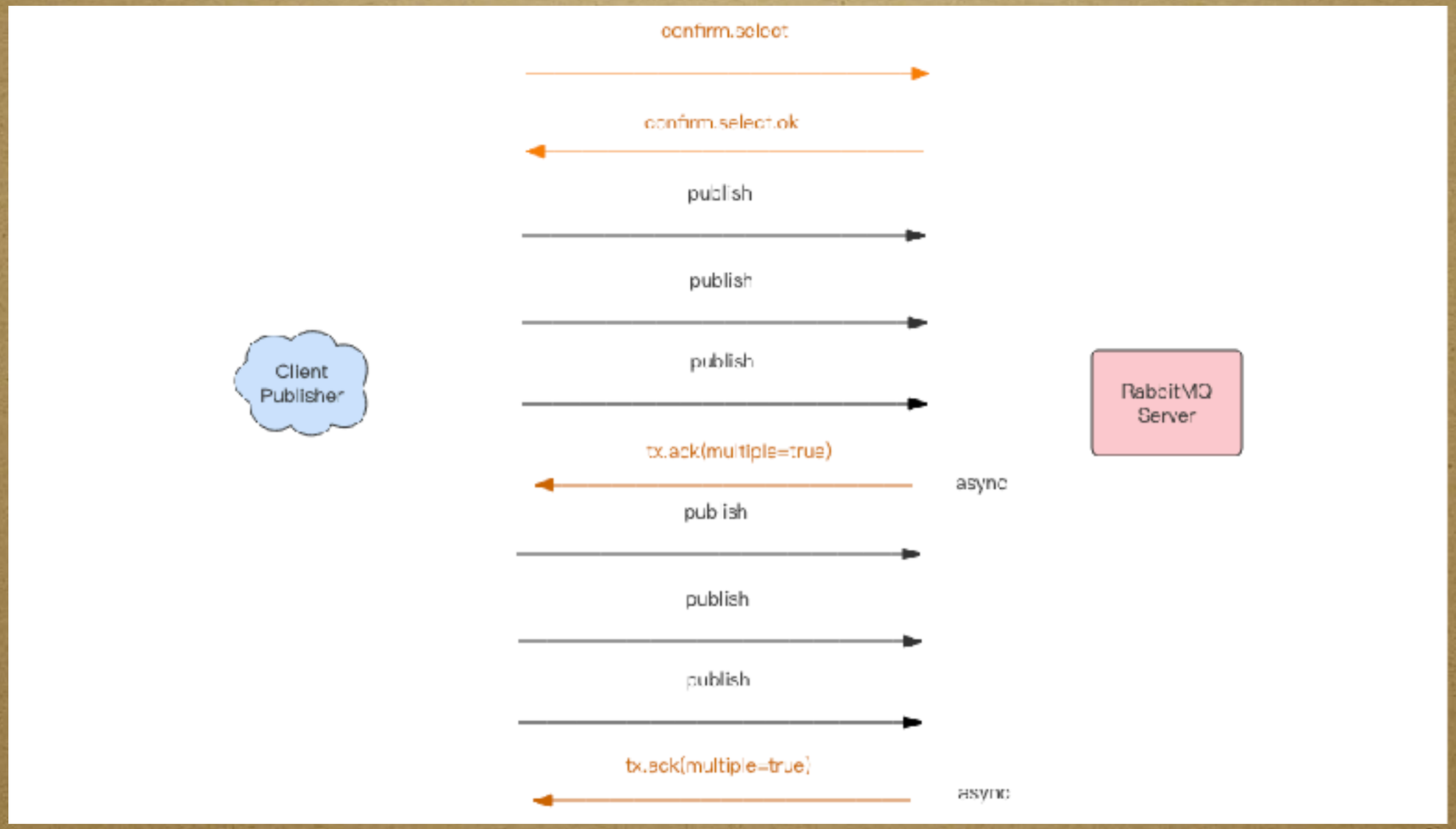


1. 用 select 和 commit 包裹 publish

2.commit要等到fsync才返回，奇慢！

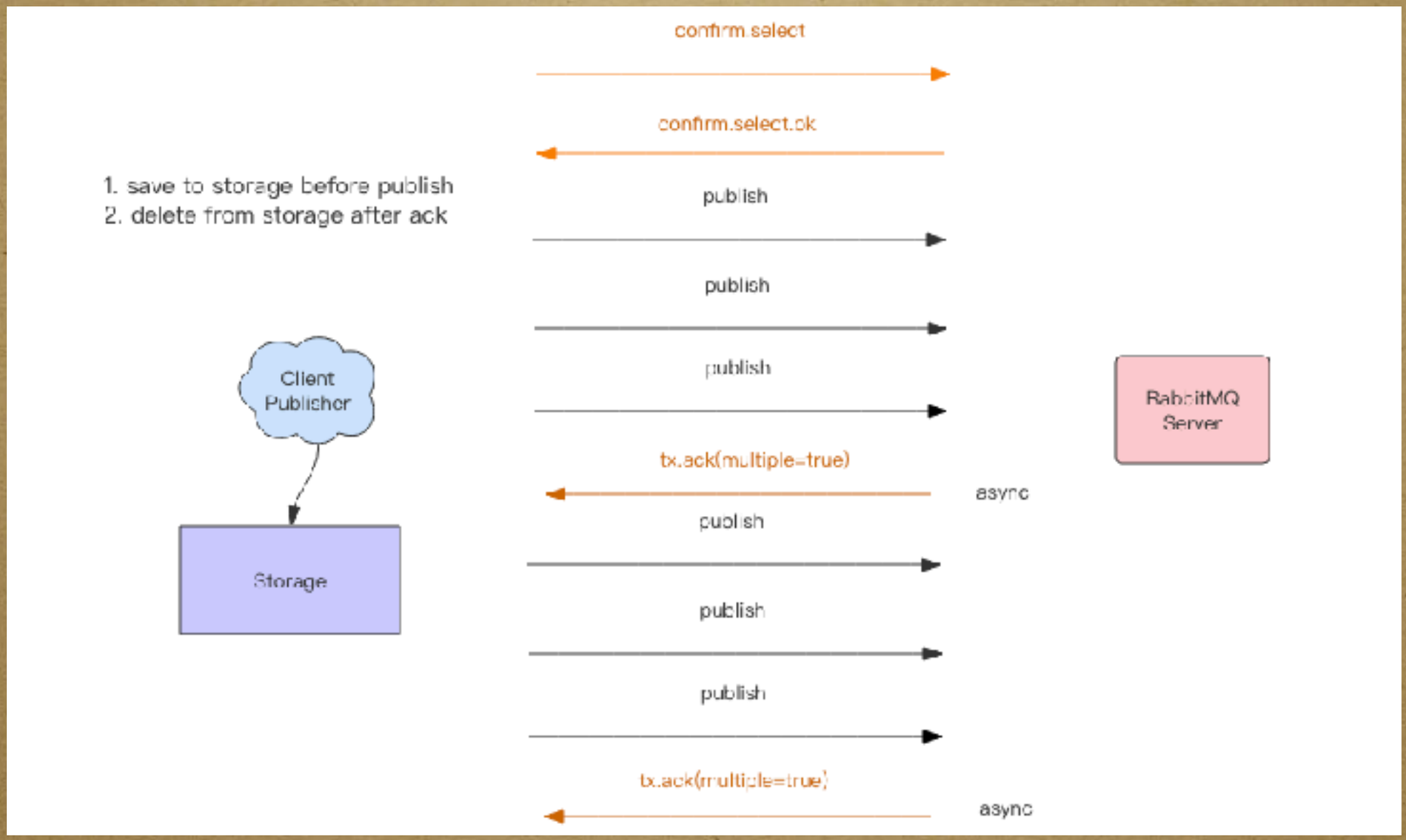
3.批量 publish

生产者确认



1. 等价于 消费者 ack（fsync）
2. 生产端需要实现消息重发机制（难）
3. 没有重发机制的confirm没什么用(除了降速)

生产者确认(重发)



1. 存内存会丢

2. 存磁盘需要fsync（无状态变有状态）

3.存 redis 还会遇到网络故障(Redis也会丢)

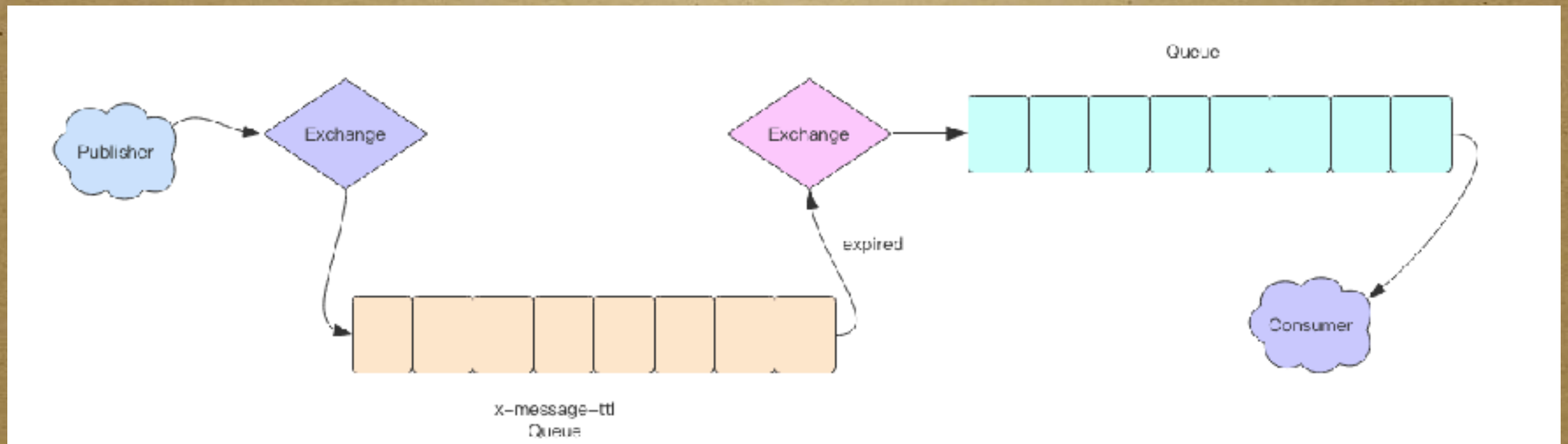
4.ack丢失，重发会导致消息重复

消息过期

Queue x-message-ttl 属性

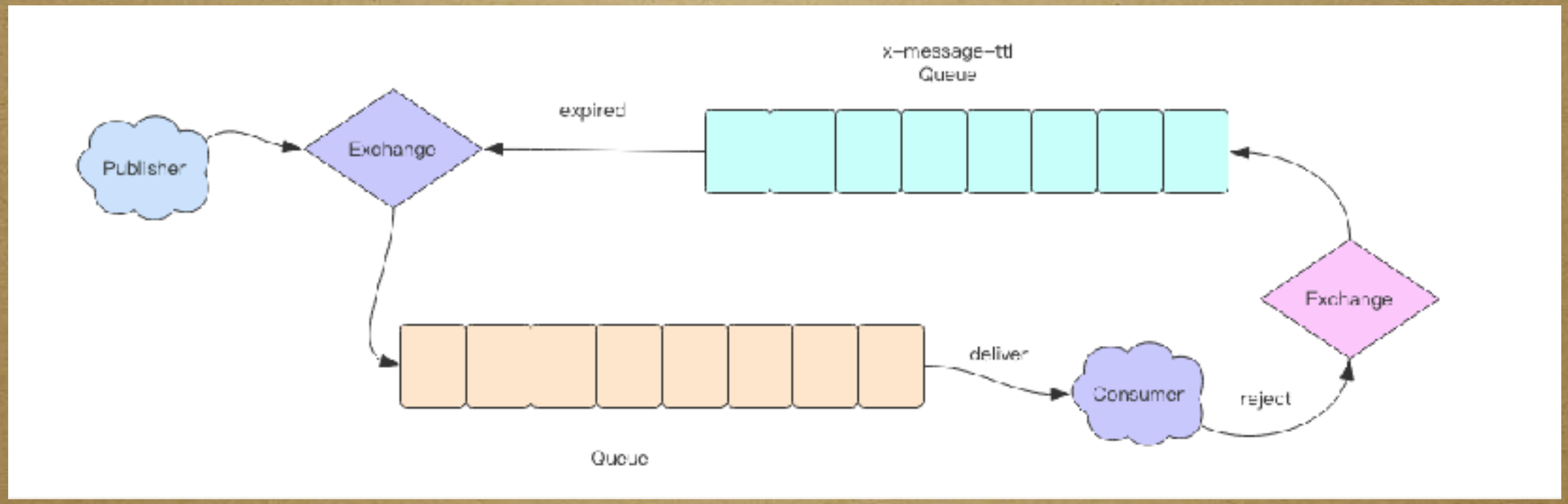
回收站

死信队列



1. 延时队列
2. 过期时间比较死，不灵活
3.不同的过期时间需要不同的过期队列

死信队列



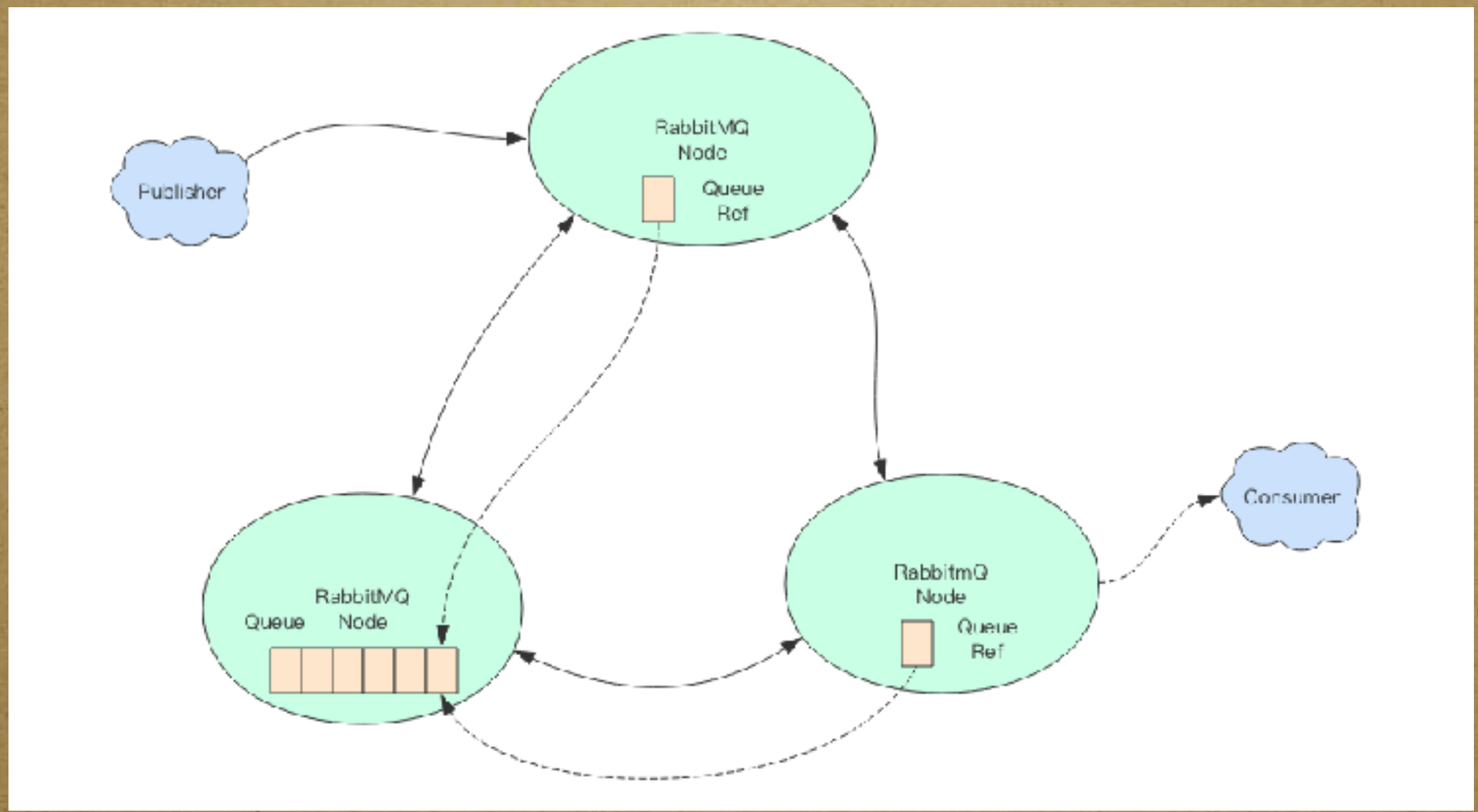
Retry Later(双重死信)

1. 消息处理异常 客户端reject 消息进入死信队列
    1. 死信队列里消息过期重新入队列

死信队列

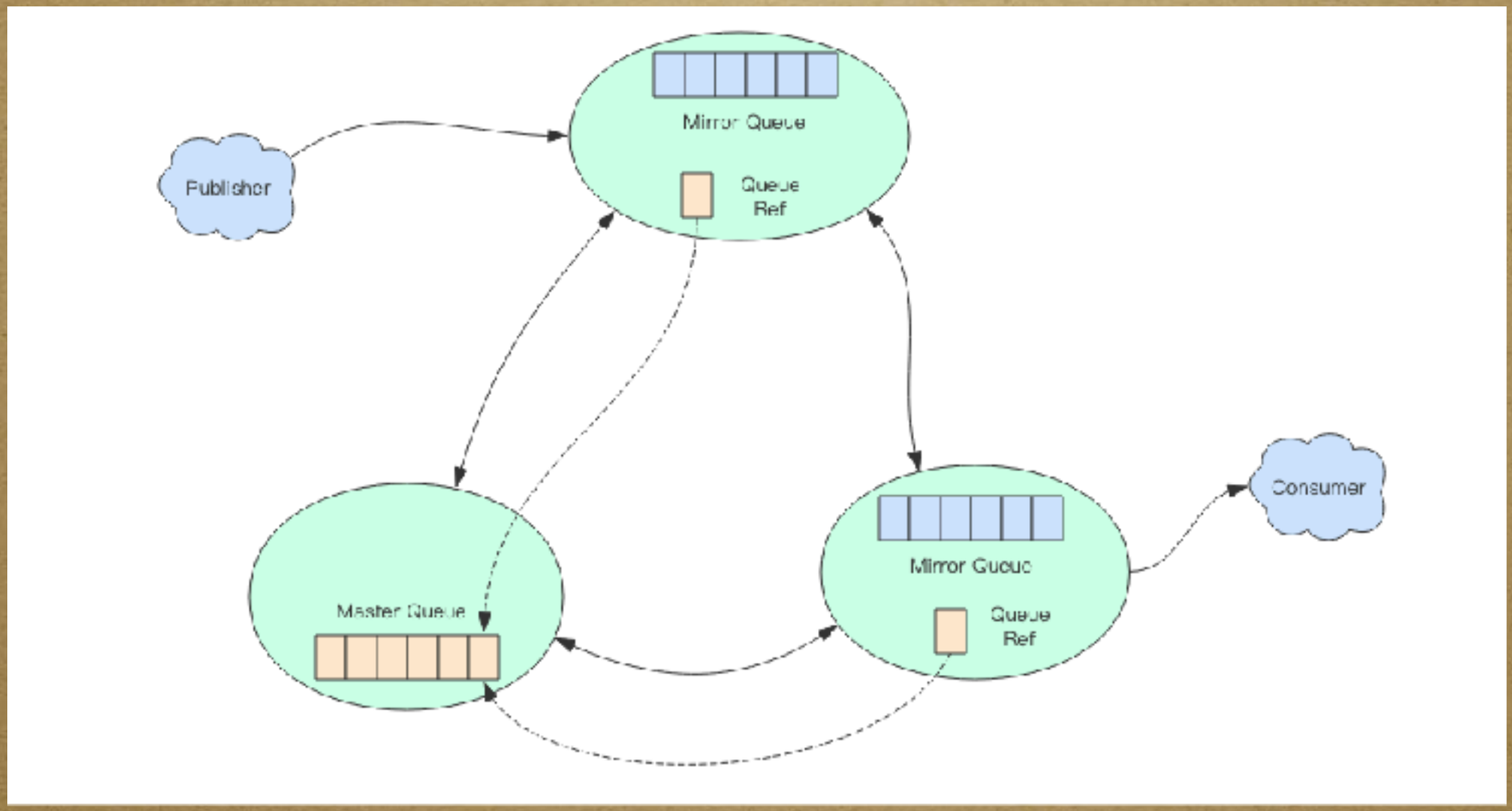
演示

集群



1. 元信息每个节点都有
2. 队列里的消息只有一份
3. 客户端只会链接一个节点(负载均衡)
4. 服务端转发

镜像队列


Publisher
Mirror Queue
Queue
Ref
Master Queue
Mirror Queue
Queue
Ref
Consumer

镜像队列

1. 镜像队列是默默无闻的
2. ha-mode=all|exactly|nodes
3. ha-params=n/2+1
4. x-queue-master-locator=min-master|client-local|random
5. ha-sync-mode=manual|automatic
6. ha-sync-batch-size=1000
7. ha-promote-on-shutdown=when-synced|always
8. ha-promote-on-failure=when-synced|always(v3.7.5+)

镜像队列

Disk and RAM Nodes

A node can be a *disk node* or a *RAM node*. (**Note:** *disk* and *disc* are used interchangeably). RAM nodes store internal database tables in RAM only. This does not include messages, message store indices, queue indices and other node state.

In the vast majority of cases you want all your nodes to be disk nodes; RAM nodes are a special case that can be used to improve the performance clusters with high queue, exchange, or binding churn. RAM nodes do not provide higher message rates. When in doubt, use disk nodes only.

Since RAM nodes store internal database tables in RAM only, they must sync them from a peer node on startup. This means that a cluster must contain at least one disk node. It is therefore not possible to manually remove the last remaining disk node in a cluster.

配置 Ram Node 对性能提升无益

镜像队列

演示

Game Over