两种消息模型: 点对点&&发布/订阅

点对点：

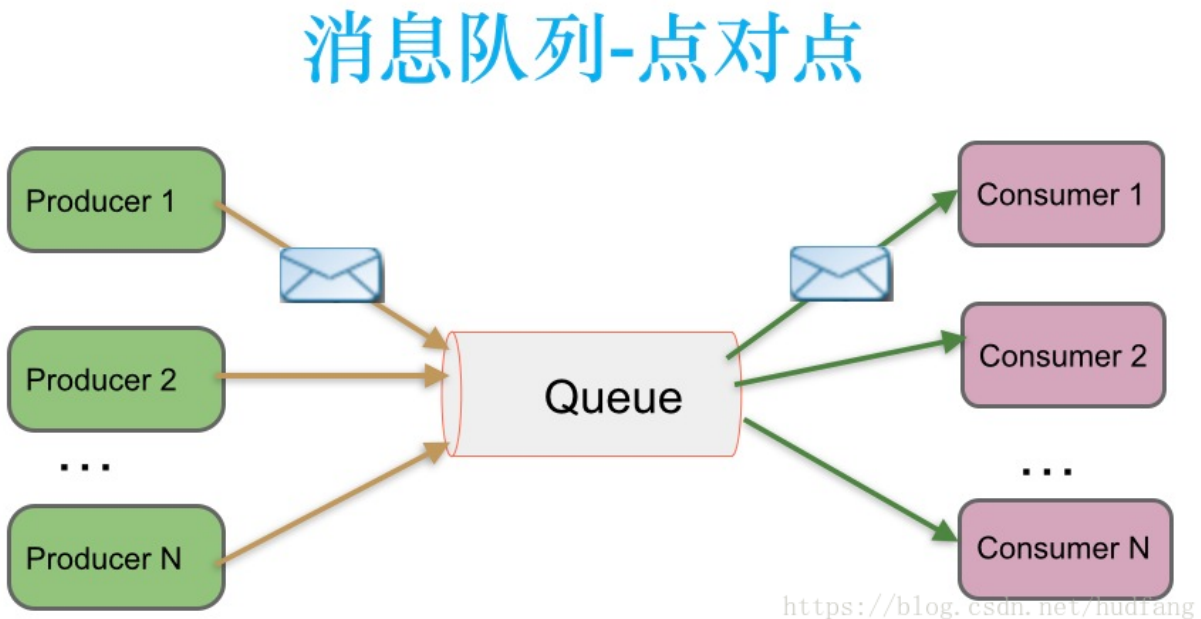
消息生产者生产消息发送到queue中，然后消息消费者从queue中取出并且消费消息。这里要注意：

消息被消费以后，queue中不再有存储，所以消息消费者不可能消费到已经被消费的消息。

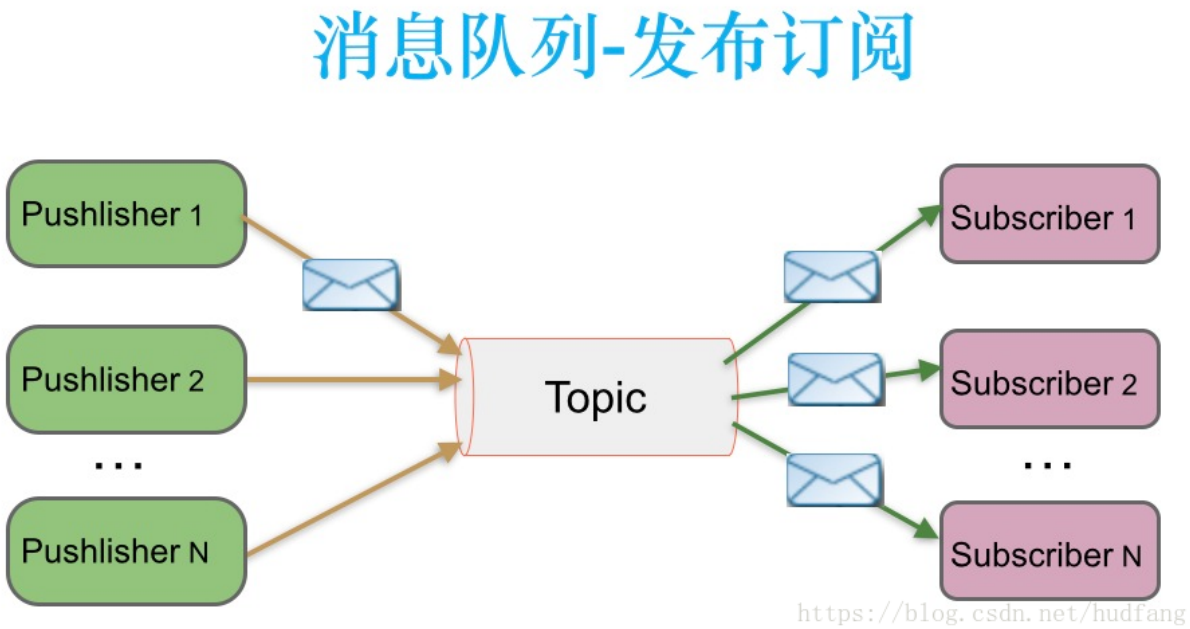
Queue支持存在多个消费者，但是对一个消息而言，只会有一个消费者可以消费。

发布/订阅

消息生产者（发布）将消息发布到topic中，同时有多个消息消费者（订阅）消费该消息。和点对点方式不同，发布到topic的消息会被所有订阅者消费。



生产者发送一条消息到queue，只有一个消费者能收到



发布者发送到topic的消息，只有订阅了topic的订阅者才会收到消息

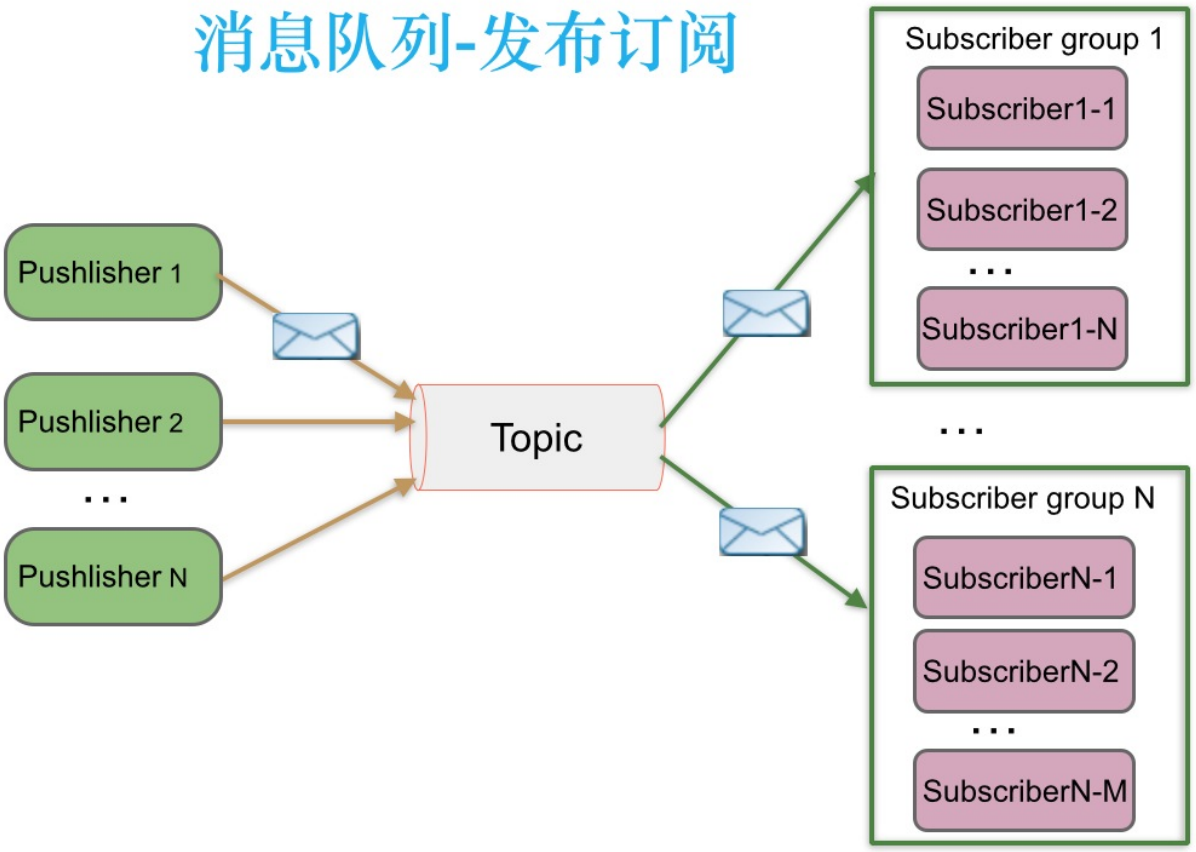
小结

queue实现了负载均衡，一个消息只能被一个消费者接受，当没有消费者可用时，这个消息会被保存直到有 一个可用的消费者，一个queue可以有很多消费者，他们之间实现了负载均衡，所以Queue实现了一个可靠的负载均衡。

topic实现了发布和订阅，当你发布一个消息，所有订阅这个topic的服务都能得到这个消息，所以从1到N个订阅者都能得到一个消息的拷贝， 只有在消息代理收到消息时有一个有效订阅时的订阅者才能得到这个消息的拷贝

疑问

发布订阅模式下，能否实现订阅者负载均衡消费呢？当发布者消息量很大时，显然单个订阅者的处理能力是不足的。实际上现实场景中是多个订阅者节点组成一个订阅组负载均衡消费topic消息即分组订阅， 这样订阅者很容易实现消费能力线性扩展。

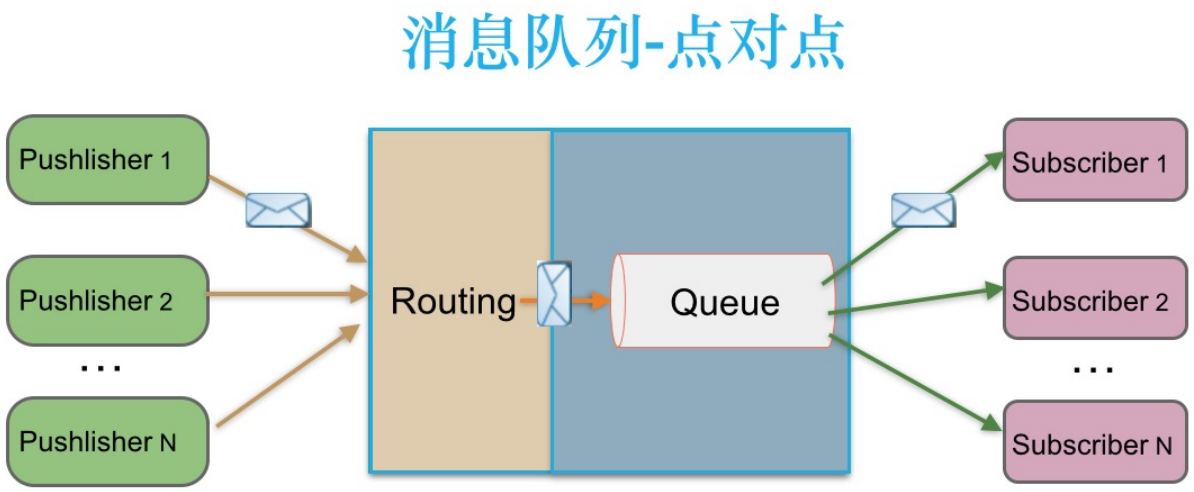


3 流行消息队列的消息模型比较

传统企业型消息队列ActiveMQ遵循了JMS规范，实现了点对点和发布订阅模型，但其他流行的消息队列RabbitMQ、Kafka并没有遵循老态龙钟的JMS规范，是通过什么方式实现消费负载均衡、多订阅呢？

3.1 RabbitMQ

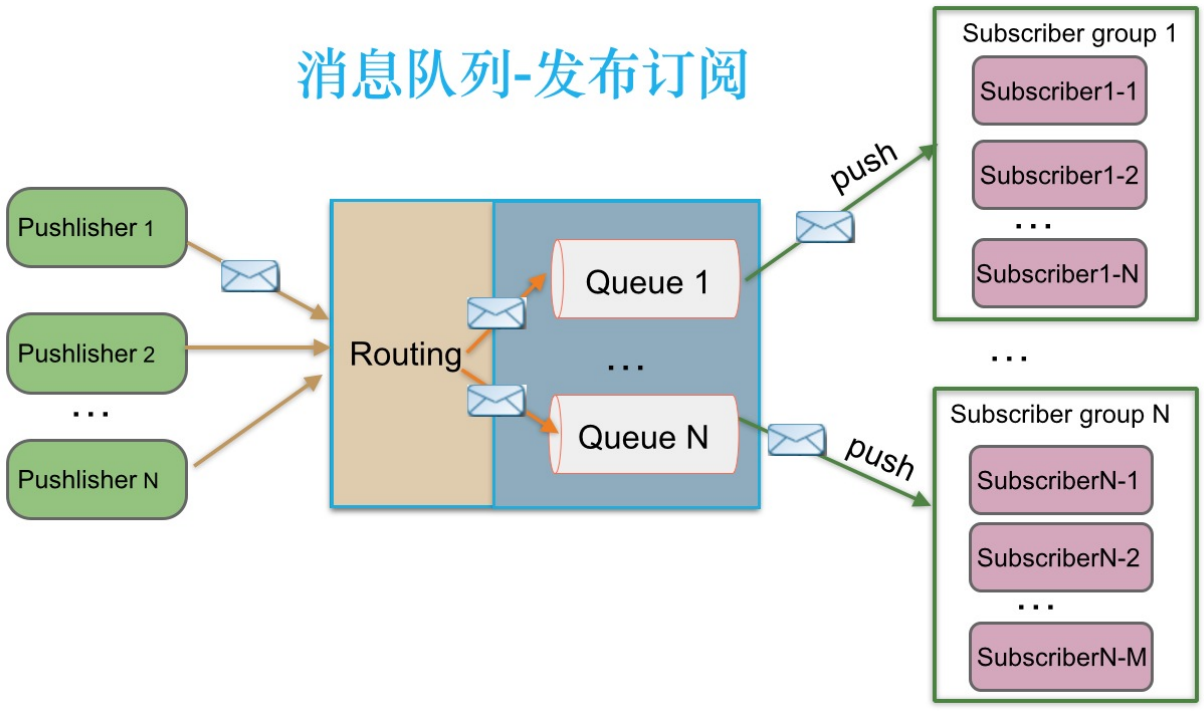
RabbitMQ实现了AQMP协议，AQMP协议定义了消息路由规则和方式。生产端通过路由规则发送消息到不同queue，消费端根据queue名称消费消息。此外RabbitMQ是向消费端推送消息，订阅关系和消费状态保存在服务端。



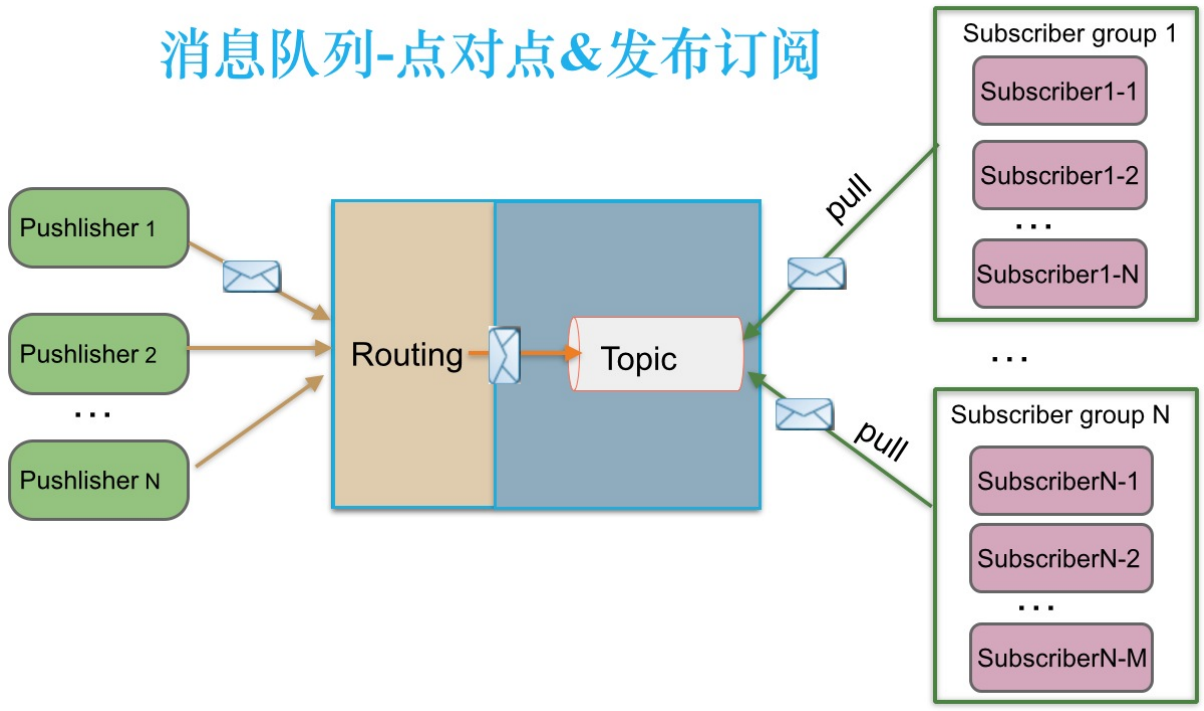
生产端发送一条消息通过路由投递到Queue，只有一个消费者能消费到。

当RabbitMQ需要支持多订阅时，发布者发送的消息通过路由同时写到多个Queue，不同订阅组消费此消息。

RabbitMQ既支持内存队列也支持持久化队列，消费端为推模型，消费状态和订阅关系由服务端负责维护，消息消费完后立即删除，不保留历史消息。所以支持多订阅时，消息会多个拷贝



3.2 Kafka



Kafka只支持消息持久化，消费端为拉模型，消费状态和订阅关系由客户端端负责维护，消息消费完后不会立即删除，会保留历史消息。因此支持多订阅时，消息只会存储一份就可以了。