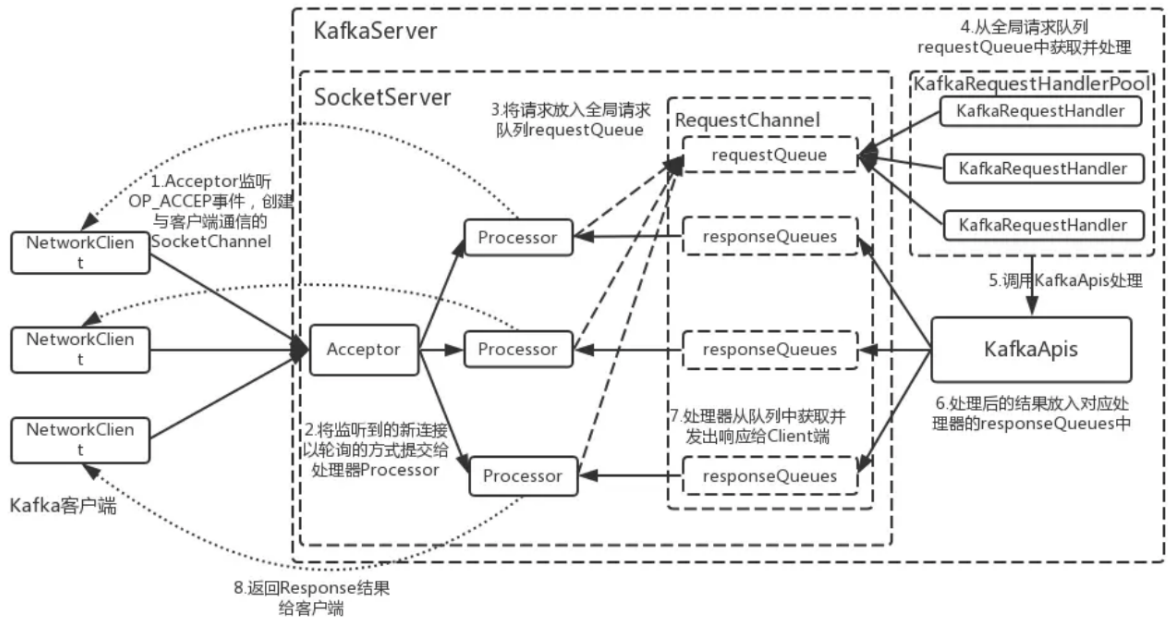
Kafka的网络通信模型是基于NIO的***Reactor***多线程模型来设计的。这里先引用Kafka源码中注释的一段话：

> An NIO socket server. The threading model is 1 Acceptor thread that handles new connections.Acceptor has N Processor threads that each have their own selector and read requests from sockets.M Handler threads that handle requests and produce responses back to the processor threads for writing.

Kafka的网络通信层模型，主要采用了**1（1个Acceptor线程）+N（N个Processor线程）+M（M个业务处理线程）。**

线程数	线程名	线程具体
1	kafka-socket-acceptor_%x	Acceptor线程，负责监听
N	kafka-network-thread_%d	Processor线程，负责对
M	kafka-request-handler-_%d	Worker线程，处理具体的业务



Kafka消息队列的通信层模型—1+N+M模型.png

初始化的Acceptor数量取决于用户配置的 **listeners** 有几个，每个Acceptor 对应的 **Processor** 的数量取决于 ***num.network.threads*** 配置，该配置默认值是

3，表示每个Acceptor分配3个Processor。每个Processor 对应的 Handler 数量由 ***num.io.threads*** 配置决定，该配置默认值是8。

请求过程

1. kakfa server在启动时调用SocketServer#startup()方法，这个方法内会初始化N个Acceptor开始监听OP_ACCEPT事件，等待客户端连接。初始化的Acceptor数量取决于用户配置的listeners有几个。在初始化每个Acceptor的同时，还会初始化M个Processor，并分配给Acceptor用于监听连接事件。
2. Acceptor接收到一个新的连接时，会将这个请求以轮询的方式分配给它管理的其中一个Processor处理
3. Processor收到一个连接时，便开始监听它的OP_READ事件
4. 如果Processor发现有请求发过来，就将这个请求放入Request队列中，等待处理。该Request队列的容量由配置***queued.max.requests***决定，默认值是500.
5. kakfa server在启动时会初始化KafkaRequestHandlerPool类，该类在初始化时会构造一些的KafkaRequestHandler线程并启动。
6. KafkaRequestHandler线程启动后，会不断自旋，从request queue中获取请求，然后交给KafkaApis进行处理。KafkaApis根据请求的类型进行不同的业务处理
7. KafkaApis组件处理完后，会将结果放入对应的Processor的response queue中，等待Processor处理
8. Processor也是一个不断自旋的线程，在自旋的过程中，Processor会检查自己的response queue中是否有新的结果，如果有新的结果就将其从队列中取出，准备发回给客户端
9. Processor通过NioChannel将结果写回客户端，自此一个通信流程结束

参考文章：

https://www.jianshu.com/p/a6b9e5342878
https://blog.csdn.net/u013332124/article/details/81367597