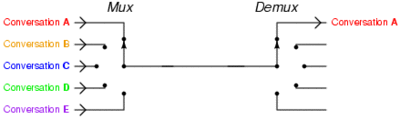
## IO多路复用

其实I/O多路复用这个坑爹翻译可能是这个概念在中文里面如此难理解的原因。所谓的I/O多路复用在英文中其实叫 I/O multiplexing.重要的事情再说一遍： I/O multiplexing 这里面的 multiplexing 指的其实是在单个线程通过记录跟踪每一个Sock(I/O流)的状态来同时管理多个I/O流. 发明它的原因，是尽量多的提高服务器的吞吐能力。

===============================================================================

https://juejin.im/post/5a069b6d51882509e5432656#heading-11

\*ip地址位于网络层

\*mac地址位于数据链路层

\*tcp/ip协议连接需要三次握手（a发送b接受--b确定接受功能正常，b发送a接收--a确定发送和接受都正常，a再次发送b接受--b确定发送正常，双方均通过握手确定了自己的收发功能正常，可以建立连接）

断开需要四次（a发送断开连接请求，b发送确认a进入等待，b再次发送断开连接请求，a发送确认并断开连接）

中断连接端可以是客户端，也可以是服务器端。

第一次挥手：客户端发送一个FIN=M，用来关闭客户端到服务器端的数据传送，客户端进入FIN\_WAIT\_1状态。意思是说"我客户端没有数据要发给你了"，但是如果你服务器端还有数据没有发送完成，则不必急着关闭连接，可以继续发送数据。

第二次挥手：服务器端收到FIN后，先发送ack=M+1，告诉客户端，你的请求我收到了，但是我还没准备好，请继续你等我的消息。这个时候客户端就进入FIN\_WAIT\_2 状态，继续等待服务器端的FIN报文。

第三次挥手：当服务器端确定数据已发送完成，则向客户端发送FIN=N报文，告诉客户端，好了，我这边数据发完了，准备好关闭连接了。服务器端进入LAST\_ACK状态。

第四次挥手：客户端收到FIN=N报文后，就知道可以关闭连接了，但是他还是不相信网络，怕服务器端不知道要关闭，所以发送ack=N+1后进入TIME\_WAIT状态，如果Server端没有收到ACK则可以重传。服务器端收到ACK后，就知道可以断开连接了。客户端等待了2MSL后依然没有收到回复，则证明服务器端已正常关闭，那好，我客户端也可以关闭连接了。最终完成了四次握手。

\*TCP面向有连接请求，是可靠的连接

\*UDP面向无连接请求，是不可靠的连接

\*Netty是基于java NIO(NoneBlocking IO)技术封装的一套框架

Netty是什么？

1）本质：JBoss做的一个Jar包

2）目的：快速开发高性能、高可靠性的网络服务器和客户端程序

3）优点：提供异步的、事件驱动的网络应用程序框架和工具

通俗的说：一个好使的处理Socket的东东

为什么netty基于NIO而不是AIO?

AIO是异步非阻塞的，NIO是同步非阻塞的。理论上讲，AIO的吞吐量肯定比NIO的要大。两种IO的概念如下：

NIO模型: 同步非阻塞。服务器实现模式为一个请求一个线程，但客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有I/O请求时才启动一个线程进行处理。

AIO模型: 异步非阻塞。服务器实现模式为一个有效请求一个线程，客户端的I/O请求都是由OS先完成了再通知服务器应用去启动线程进行处理。注：AIO又称为NIO2.0，在JDK7才开始支持。

主要原因如下：

Netty不看重Windows上的使用，在Linux系统上，AIO的底层实现仍使用EPOLL，没有很好实现AIO，因此在性能上没有明显的优势，而且被JDK封装了一层不容易深度优化。

Netty整体架构是reactor模型, 而AIO是proactor模型, 混合在一起会非常混乱,把AIO也改造成reactor模型看起来是把epoll绕个弯又绕回来。

AIO还有个缺点是接收数据需要预先分配缓存, 而不是NIO那种需要接收时才需要分配缓存, 所以对连接数量非常大但流量小的情况, 内存浪费很多。

Linux上AIO不够成熟，处理回调结果速度跟不上处理需求，比如外卖员太少，顾客太多，供不应求，造成处理速度有瓶颈（待验证）。

作者原话引用：

作者原话：

Not faster than NIO (epoll) on unix systems (which is true)

There is no daragram suppport

Unnecessary threading model (too much abstraction without usage)

在unix系统上不比NIO（epoll）快（这是真的）

没有daragram支持

不必要的线程模型（没有使用过多的抽象）