TCP协议的三次握手和四次挥手你真的懂吗?

背景

和女朋友异地恋一年多，为了保持感情我提议每天晚上视频聊天一次。

从好上开始，到现在，一年多也算坚持下来了。

问题

有时候聊天的过程中，我的网络或者她的网络可能会不好，视频就会卡住，听不到对方的声音，过一会儿之后才会恢复。

中间双方可能就要不断的确认网络是否恢复，但是有时候会：

她：“你可以听到了吗？”

我：“可以了，你呢？”、

她：“喂喂，你可以听到了吗？”

我：“可以了，我可以听到了，你呢？”

她：“你可以听到了吗？”

.....

这种情况很蛋疼，那么怎样才能找一个简单的办法，让两个人都确认自己可以听到对方的声音，对方也可以听到自己的声音呢?

注：以下情节纯属虚构

方案

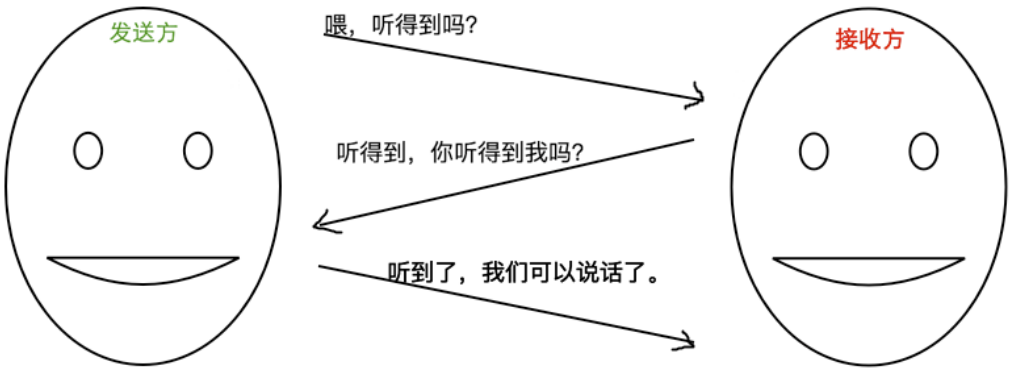
TCP建立连接为什么是三次握手，而不是两次或四次?

TCP，名为传输控制协议，是一种可靠的传输层协议，IP协议号为6。

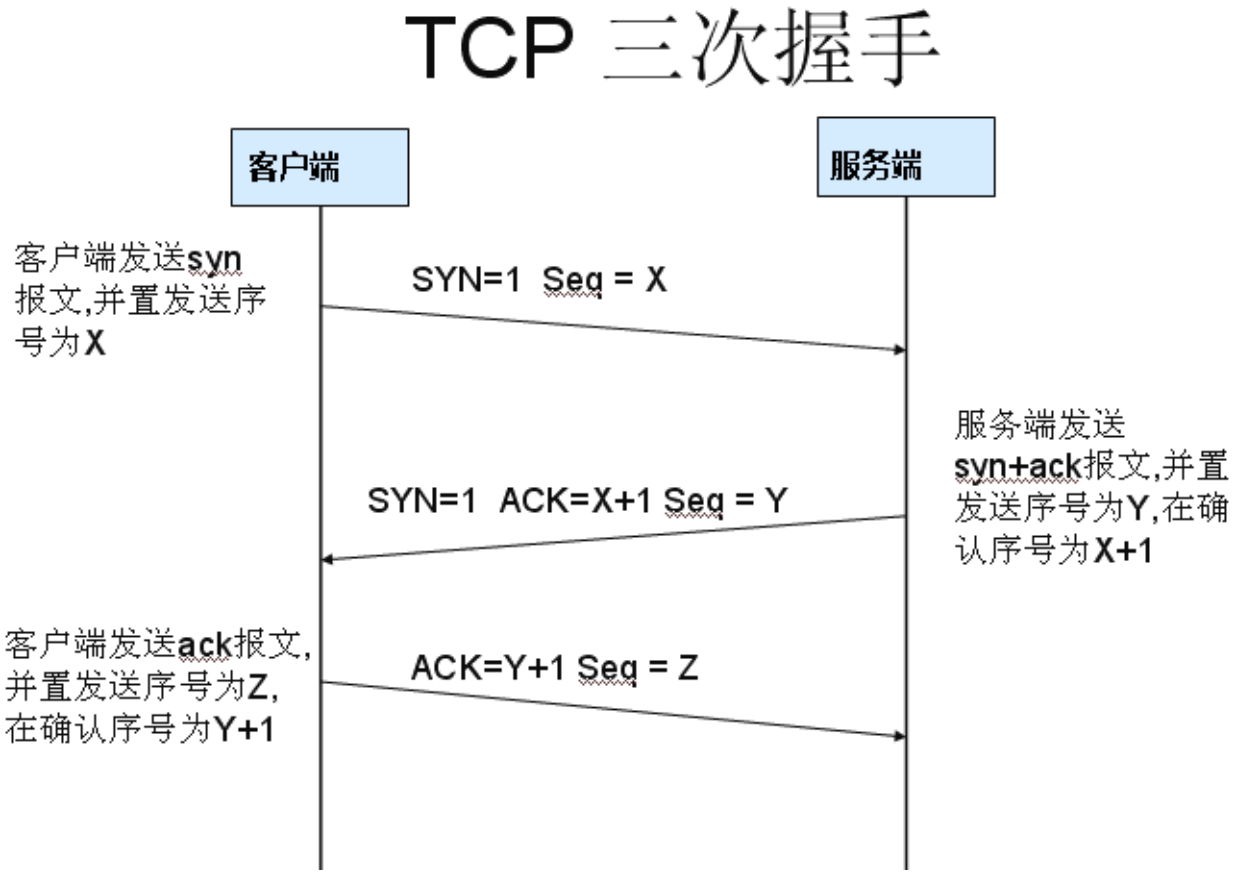
顺便说一句，原则上任何数据传输都无法确保绝对可靠，三次握手只是确保可靠的基本需要。

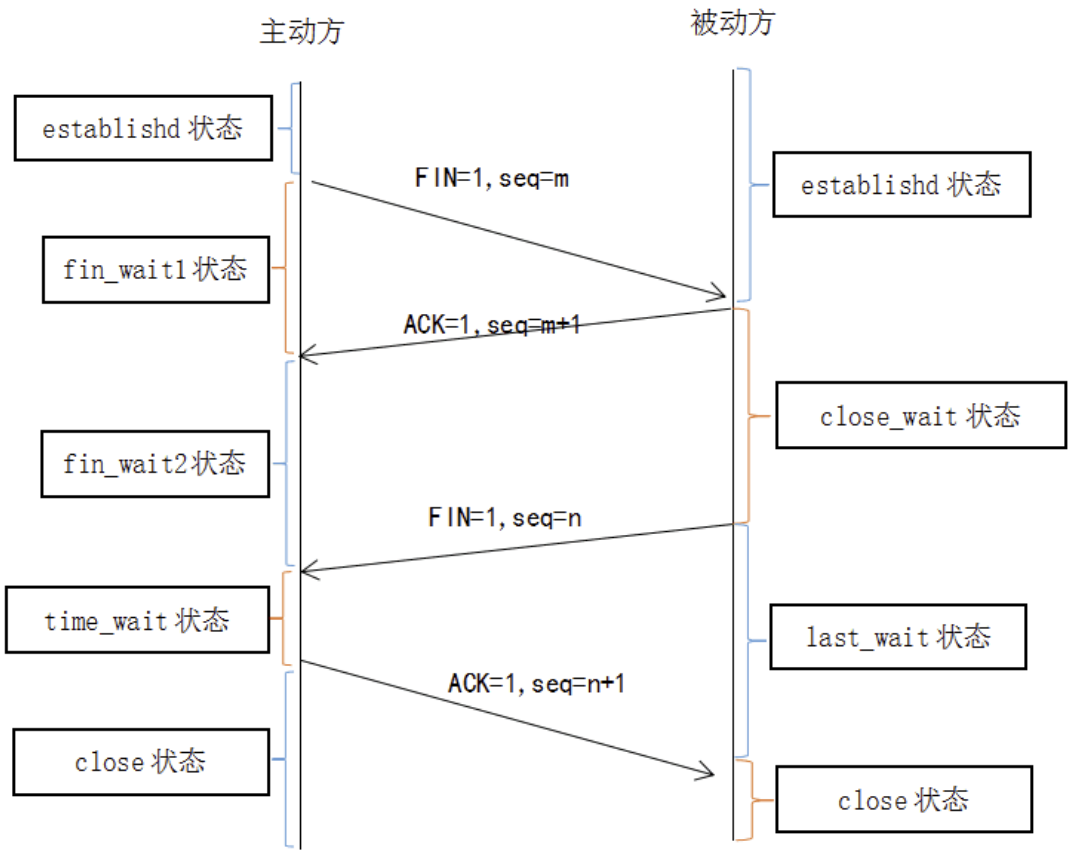
举个日常例子，打电话时我们对话如下：

TCP三次握手



对应为客户端与服务器之间的通信：





TCP 关闭连接（四次握手）

于是有了如下对话：

我：1+1等于几?

她：2,2+2等于几?

我：4

首先两个人约定协议

1.感觉网络情况不对的时候，任何一方都可以发起询问

2.任何情况下，若发起询问后5秒还没收到回复，则认为网络不通

3.网络不通的情况下等1min路由器之后再发起询问

对于我而言，发起 “1+1等于几”的询问后

1. 若5s内没有收到回复，则认为网络不通
2. 若收到回复，则我确认①我能听到她的消息 ②她能听到我的消息，然后回复她的问题的答案

对于她而言，当感觉网络情况不对的时候

1. 若没有收到我的询问，则她发起询问
2. 若收到“1+1等于几”，则她确认 ①她可以听到我的消息，然后回复我的问题的答案和她的问题“2，

2+2等于几”

3. 若5s内没有收到我的回复“4”，则她确认 ②我听不见她的消息
4. 若5s内收到了我的回复“4”，则她确认 ②我可以听见她的消息

这样，如果上面的对话得以完成，就证明双方都可以确认自己可以听到对方的声音，对方也可以听到自己的声音！

这个故事可以解释TCP为什么要三次握手吗 ... 囧

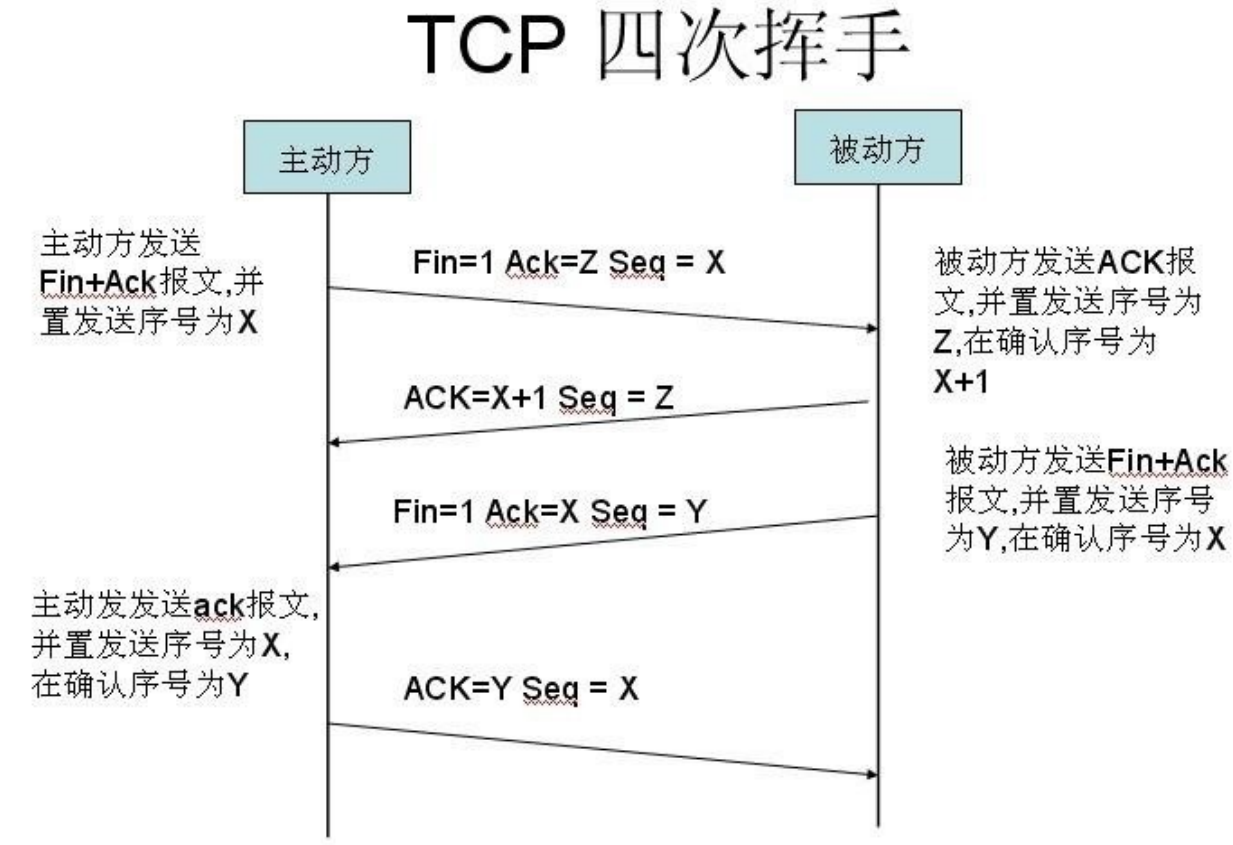
关于四次挥手

先由客户端向服务器端发送一个FIN，请求关闭数据传输。

当服务器接收到客户端的FIN时，向客户端发送一个ACK，其中ack的值等于FIN+SEQ

然后服务器向客户端发送一个FIN，告诉客户端应用程序关闭。

当客户端收到服务器端的FIN是，回复一个ACK给服务器端。其中ack的值等于FIN+SEQ



为什么要4次挥手?

确保数据能够完整传输。

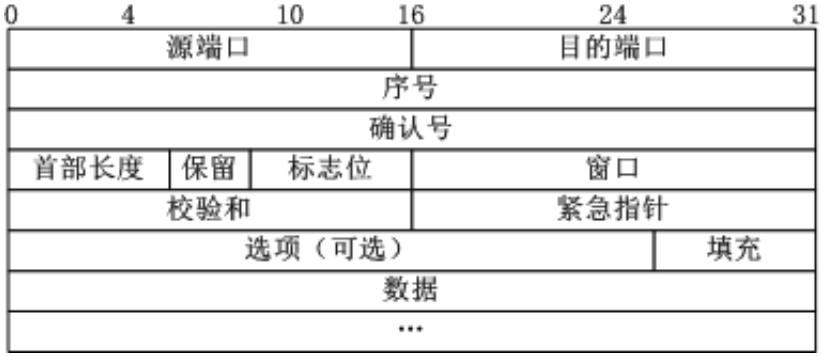
当被动方收到主动方的FIN报文通知时，它仅仅表示主动方没有数据再发送给被动方了。

但未必被动方所有的数据都完整的发送给了主动方，所以被动方不会马上关闭SOCKET,它可能还需要发送一些数据给主动方后，

再发送FIN报文给主动方，告诉主动方同意关闭连接，所以这里的ACK报文和FIN报文多数情况下都是分开发送的。

一、TCP报文格式

TCP报文格式图：



上图中有几个字段需要重点介绍下：

（1）序号：Seq序号，占32位，用来标识从TCP源端向目的端发送的字节流，发起方发送数据时对此进行标记。

（2）确认序号：Ack序号，占32位，只有ACK标志位为1时，确认序号字段才有效，Ack=Seq+1。

（3）标志位：共6个，即URG、ACK、PSH、RST、SYN、FIN等，具体含义如下：

（A）URG：紧急指针（urgent pointer）有效。

（B）ACK：确认序号有效。

（C）PSH：接收方应该尽快将这个报文交给应用层。

（D）RST：重置连接。

（E）SYN：发起一个新连接。

（F）FIN：释放一个连接。

需要注意的是：

（A）不要将确认序号Ack与标志位中的ACK搞混了。

（B）确认方Ack=发起方Req+1，两端配对。

二、三次握手

TCP(Transmission Control Protocol)　传输控制协议

TCP是主机对主机层的传输控制协议，提供可靠的连接服务，采用三次握手确认建立一个连接

位码即tcp标志位,有6种标示:

SYN(synchronous建立联机)

ACK(acknowledgement 确认)

PSH(push传送)

FIN(finish结束)

RST(reset重置)

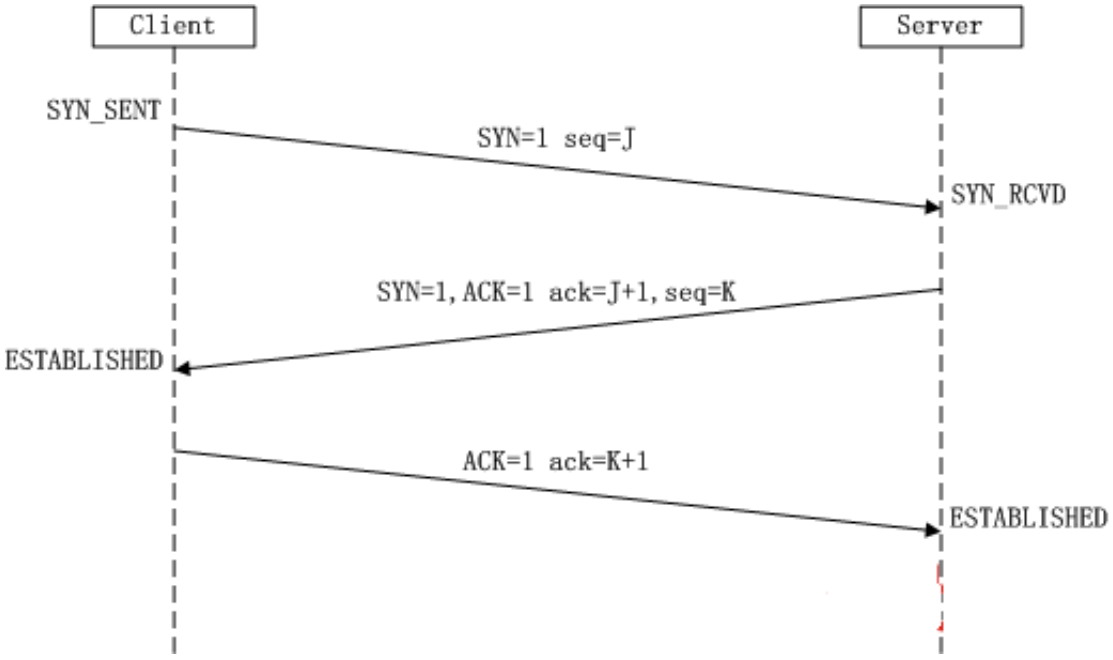
URG(urgent紧急)

Sequence number(顺序号码)

Acknowledge number(确认号码)

establish 建立，创建

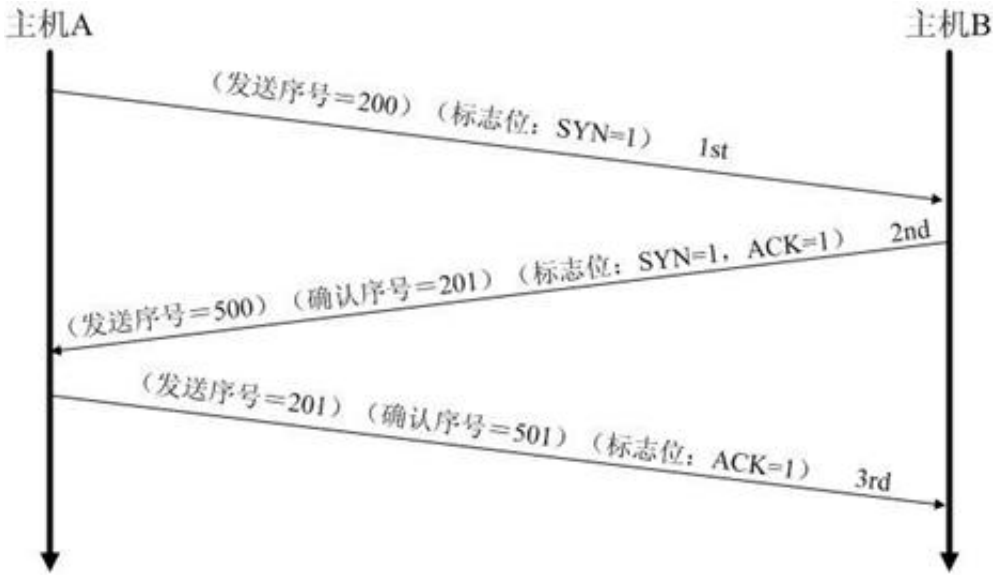
所谓**三次握手**（Three-Way Handshake）即建立TCP连接，**是指建立一个TCP连接时，需要客户端和服务端总共发送3个包以确认连接的建立**。在socket编程中，这一过程由客户端执行connect来触发，整个流程如下图所示：



（1）第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，**Client进入SYN_SENT状态**，等待Server确认。

（2）第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack (number)=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，**Server进入SYN_RCVD状态**。

（3）第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，**如果正确则连接建立成功，Client和Server进入ESTABLISHED状态**，完成三次握手，随后Client与Server之间可以开始传输数据了。



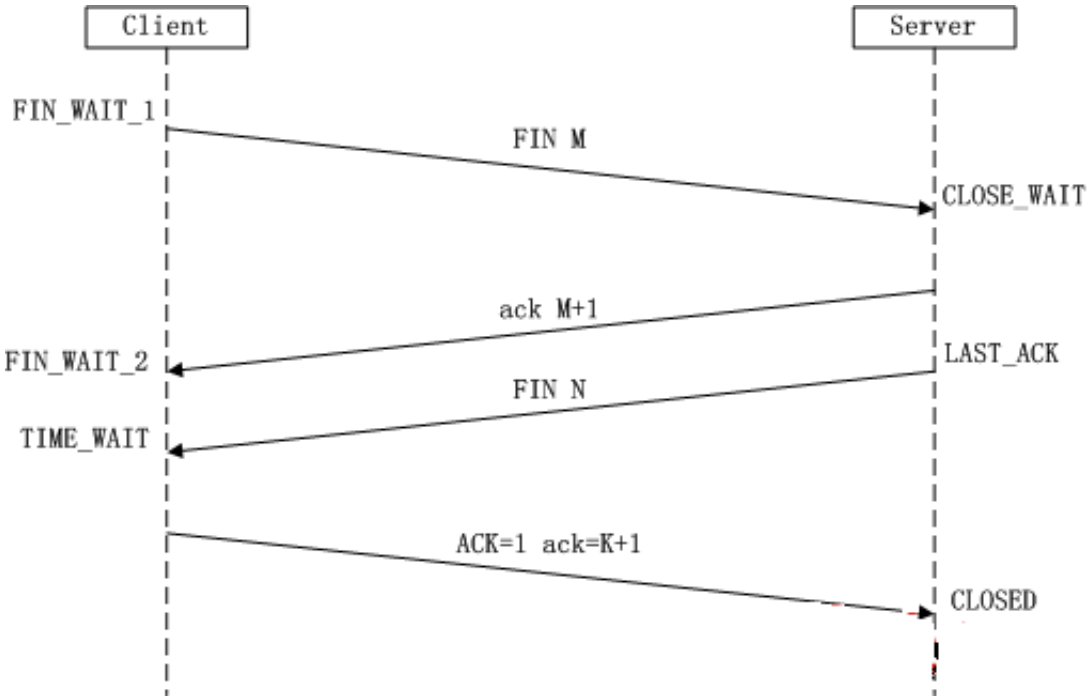
SYN攻击：

在三次握手过程中，**Server发送SYN-ACK之后，收到Client的ACK之前的TCP连接称为半连接（half-open connect），此时Server处于SYN_RCVD状态**，当收到ACK后，Server转入ESTABLISHED状态。**SYN攻击就是Client在短时间内伪造大量不存在的IP地址，并向Server不断地发送SYN包，Server回复确认包，并等待Client的确认，由于源地址是不存在的，因此，Server需要不断重发直至超时**，这些**伪造的SYN包将长时间占用未连接队列，导致正常的SYN请求因为队列满而被丢弃**，从而引起网络堵塞甚至系统瘫痪。SYN攻击时一种典型的DDOS攻击，检测SYN攻击的方式非常简单，即当Server上有大量半连接状态且源IP地址是随机的，则可以断定遭到SYN攻击了，使用如下命令可以让之现行：

#netstat -nap | grep SYN_RECV

三、四次挥手

三次握手耳熟能详，四次挥手估计就..所谓四次挥手（Four-Way Wavehand）即终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认连接的断开。在socket编程中，这一过程由客户端或服务端任一方执行close来触发，整个流程如下图所示：



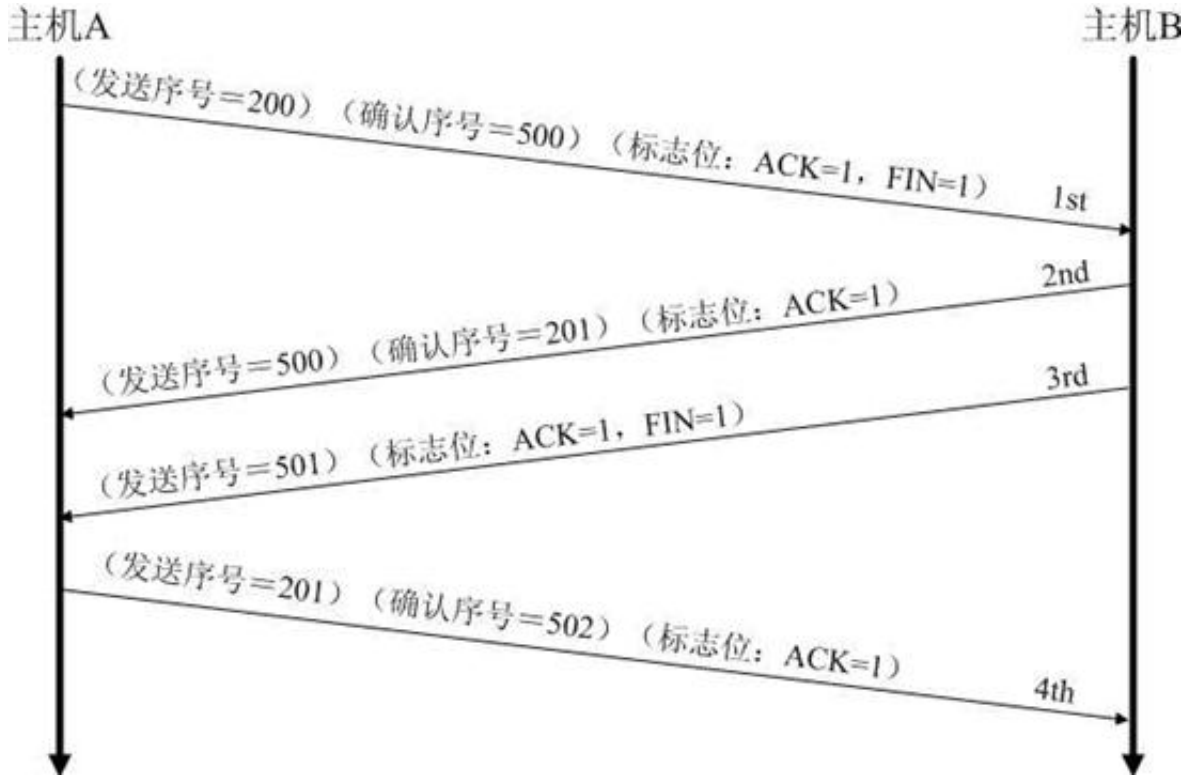
由于TCP连接时全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭，上图描述的即是如此。

（1）第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN_WAIT_1状态。

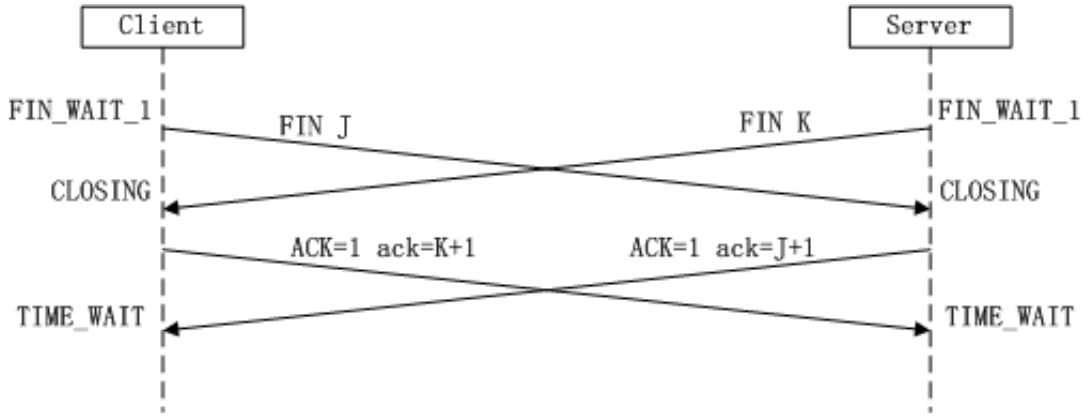
（2）第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE_WAIT状态。

（3）第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST_ACK状态。

（4）第四次挥手：Client收到FIN后，Client进入TIME_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。



上面是一方主动关闭，另一方被动关闭的情况，实际中还会出现同时发起主动关闭的情况，具体流程如下图：



流程和状态在上图中已经很明了了，在此不再赘述，可以参考前面的四次挥手解析步骤。

四、附注

关于三次握手与四次挥手通常都会有典型的面试题，在此提出供有需求的XDJM们参考：

（1）三次握手是什么或者流程？四次握手呢？答案前面分析就是。

（2）为什么建立连接是三次握手，而关闭连接却是四次挥手呢？

这是因为**服务端在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。而关闭连接时，当收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据**，己方也未必全部数据都发送给对方了，所以己方可以立即close，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会分开发送。