[**通俗易懂地讲解TCP建立连接的三次握手和释放连接的四次挥手**](https://www.cnblogs.com/xiaoming0601/p/6001021.html)

**TCP建立连接时,为什么要进行三次挥手?**

　　每一次TCP连接都需要三个阶段：连接建立、数据传送和连接释放。三次握手就发生在连接建立阶段。 在谢希仁著《计算机网络》第四版中讲三次握手的目的是为了防止已失效的连接请求报文段突然又传送到了服务端，因而产生错误。在另一部经典的《计算机网络》一书中讲三次握手的目的是为了解决网络中存在延迟的重复分组的问题。 　　这两种不用的表述其实阐明的是同一个问题。

　　谢希仁版《计算机网络》中的例子是这样的，已失效的连接请求报文段的产生在这样一种情况下：client发出的第一个连接请求报文段并没有丢失，而是在某个网络结点长时间的滞留了，以致延误到连接释放以后的某个时间才到达server。本来这是一个早已失效的报文段。但server收到此失效的连接请求报文段后，就误认为是client再次发出的一个新的连接请求。于是就向client发出确认报文段，同意建立连接。假设不采用三次握手，那么只要server发出确认，新的连接就建立了。由于现在client并没有发出建立连接的请求，因此不会理睬server的确认，也不会向server发送数据。但server却以为新的运输连接已经建立，并一直等待client发来数据。这样，server的很多资源就白白浪费掉了。采用三次握手的办法可以防止上述现象发生。例如刚才那种情况，client不会向server的确认发出确认。server由于收不到确认，就知道client并没有要求建立连接。

　　这个例子很清晰的阐释了三次握手对于建立可靠连接的意义。在Google Groups的TopLanguage中看到一帖讨论TCP三次握手觉得很有意思。贴主提出的问题，在众多回复中，有一条回复写道：这个问题的本质是, 信道不可靠, 但是通信双发需要就某个问题达成一致. 而要解决这个问题, 无论你在消息中包含什么信息, 三次通信是理论上的最小值. 所以三次握手不是TCP本身的要求, 而是为了满足"在不可靠信道上可靠地传输信息"这一需求所导致的. 请注意这里的本质需求,信道不可靠, 数据传输要可靠. 三次达到了, 那后面你想接着握手也好, 发数据也好, 跟进行可靠信息传输的需求就没关系了. 因此,如果信道是可靠的, 即无论什么时候发出消息, 对方一定能收到, 或者你不关心是否要保证对方收到你的消息, 那就能像UDP那样直接发送消息就可以了. 。这可视为对三次握手目的的另一种解答思路。

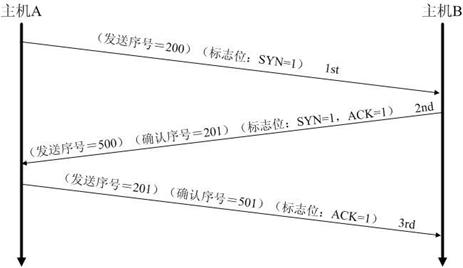
**举个打电话的例子**：

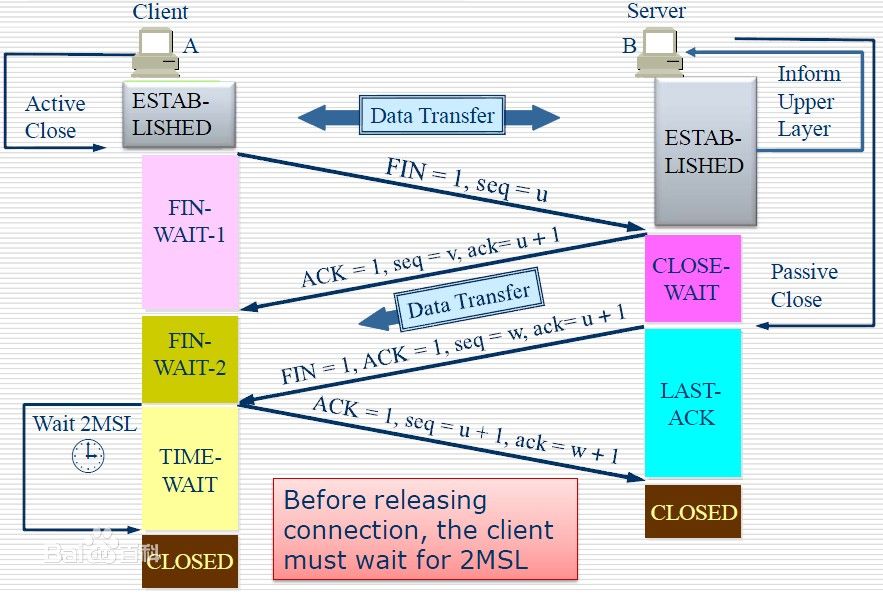
　　A : 你好我是A，你听得到我在说话吗

　　B : 听到了，我是B，你听到我在说话吗

　　A : 嗯，听到了

　　建立连接，开始聊天！





**为什么TCP协议终止链接要四次？**

1、当主机A确认发送完数据且知道B已经接受完了，想要关闭发送数据口（当然确认信号还是可以发），就会发FIN给主机B。

2、主机B收到A发送的FIN，表示收到了，就会发送ACK回复。

3、但这是B可能还在发送数据，没有想要关闭数据口的意思，所以FIN与ACK不是同时发送的，而是等到B数据发送完了，才会发送FIN给主机A。

4、A收到B发来的FIN，知道B的数据也发送完了，回复ACK， A等待2MSL以后，没有收到B传来的任何消息，知道B已经收到自己的ACK了，A就关闭链接，B也关闭链接了。

A为什么等待2MSL，从TIME\_WAIT到CLOSE？

 在Client发送出最后的ACK回复，但该ACK可能丢失。Server如果没有收到ACK，将不断重复发送FIN片段。所以Client不能立即关闭，它必须确认Server接收到了该ACK。Client会在发送出ACK之后进入到TIME\_WAIT状态。Client会设置一个计时器，等待2MSL的时间。如果在该时间内再次收到FIN，那么Client会重发ACK并再次等待2MSL。所谓的2MSL是两倍的MSL(Maximum Segment Lifetime)。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果直到2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。

**这个网上转载的例子不错：**

**三次握手：**  
A:“喂，你听得到吗？”A->SYN\_SEND

B:“我听得到呀，你听得到我吗？”应答与请求同时发出 B->SYN\_RCVD | A->ESTABLISHED

A:“我能听到你，今天balabala……”B->ESTABLISHED

**四次挥手：**  
A:“喂，我不说了。”A->FIN\_WAIT1

B:“我知道了。等下，上一句还没说完。Balabala…..”B->CLOSE\_WAIT | A->FIN\_WAIT2

B:”好了，说完了，我也不说了。”B->LAST\_ACK

A:”我知道了。”A->TIME\_WAIT | B->CLOSED

A等待2MSL,保证B收到了消息,否则重说一次”我知道了”,A->CLOSED