三次握手、四次挥手

两个序号和三个标志位：

（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。

一、描述：

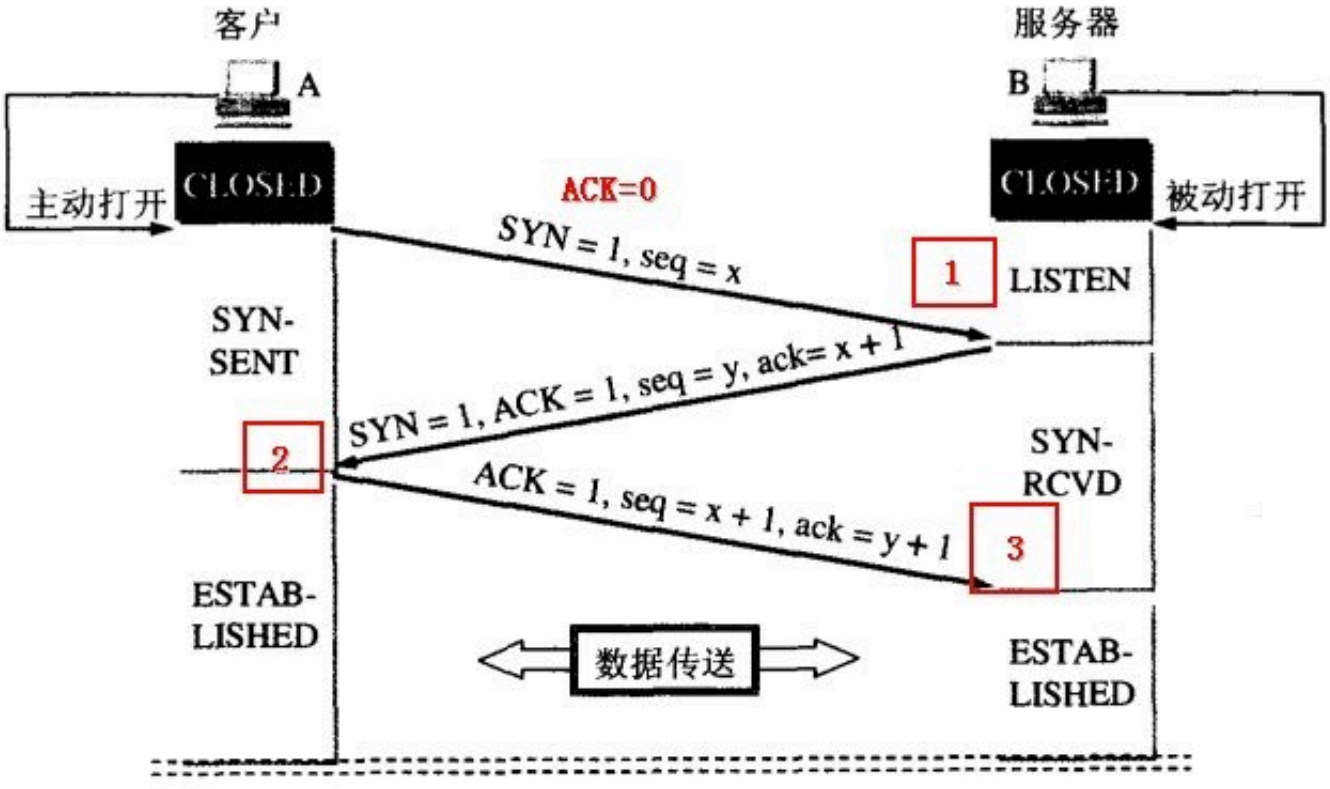


图 5-31　用三次握手建立 TCP 连接

1.在第一次消息发送中，A随机选取一个序列号作为自己的初始序号发送给B；
2.第二次消息发送时，B使用ack确认序号对A的数据包进行确认；
3.第三条消息，A告诉B收到了B的确认消息并准备建立连接，A自己此条消息的序列号是x+1，所

以seq=x+1，而ack=y+1是表示A正准备接收B序列号为y+1的数据包。

二、为什么不是两次?

为什么 A 还要发送一次确认呢？这主要是为了防止已失效的连接请求报文段突然又传送到了 B，因而产生错误。

所谓“已失效的连接请求报文段”是这样产生的。考虑一种正常情况。A 发出连接请求，但因连接请求报文丢失而未收到确认。于是 A 再重传一次连接请求。后来收到了确认，建立了连接。数据传输完毕后，就释放了连接。A 共发送了两个连接请求报文段，其中第一个丢失，第二个到达了 B。没有“已失效的连接请求报文段”。

现假定出现一种异常情况，即 A 发出的第一个连接请求报文段并没有丢失，而是在某些网络结点长时间滞留了，以致延误到连接释放以后的某个时间才到达 B。本来这是一个早已失效的报文段。但 B 收到此失效的连接请求报文段后，就误认为是 A 又发出一次新的连接请求。于是就向 A 发出确认报文段，同意建立连接。假定不采用三次握手，那么只要 B 发出确认，新的连接就建立了。

由于现在 A 并没有发出建立连接的请求，因此不会理睬 B 的确认，也不会向 B 发送数据。但 B 却以为新的运输连接已经建立了，并一直等待 A 发来数据。B 的许多资源就这样白白浪费了。

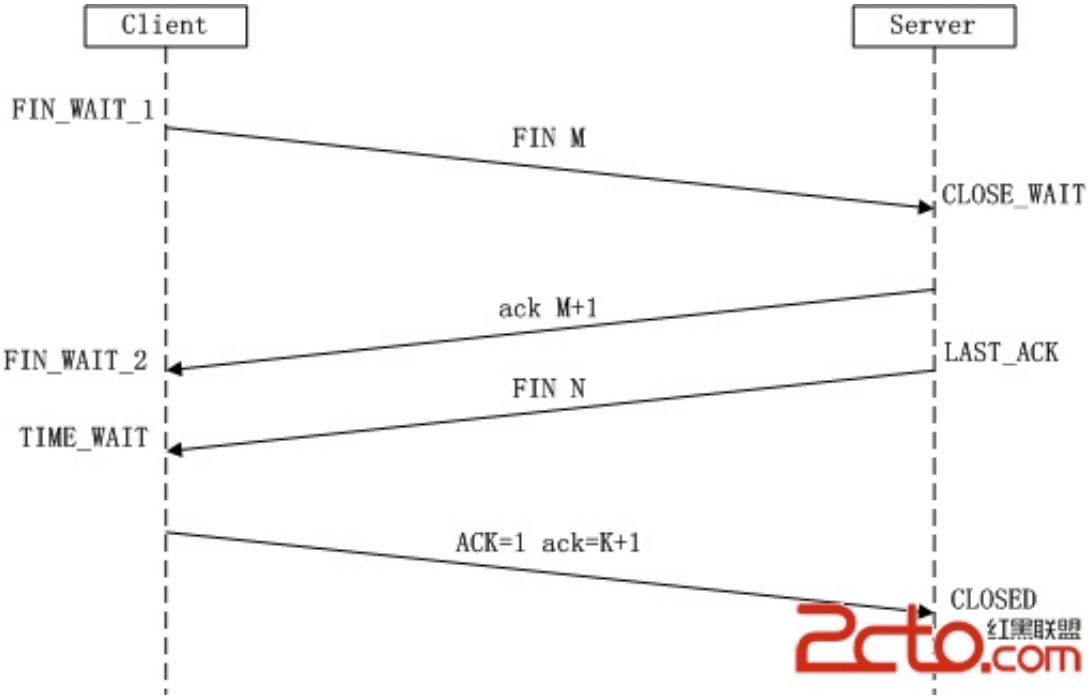
采用三次握手的办法可以防止上述现象的发生。例如在刚才的情况下，A 不会向 B 的确认发出确认。B 由于收不到确认，就知道 A 并没有要求建立连接。

四次挥手

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

> （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(标志位)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。



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server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，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。

为什么TIME_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回到CLOSE状态?

不应该是为了防止B发送的FIN=1的包的丢失，因为如果A没有收到来自B的释放连接请求，是不会进入TIME-WAIT状态的。

所以正确的解释是：A发送的确认释放连接信息B没有收到，这时候B会再次发送一个FIN=1的释放连接请求，而这个时候A还处于TIME-WAIT，所以可以再次发送确认信息