

    序列号seq：占4个字节，用来标记数据段的顺序，TCP把连接中发送的所有数据字节都编上一个序号，第一个字节的编号由本地随机产生；给字节编上序号后，就给每一个报文段指派一个序号；序列号seq就是这个报文段中的第一个字节的数据编号。

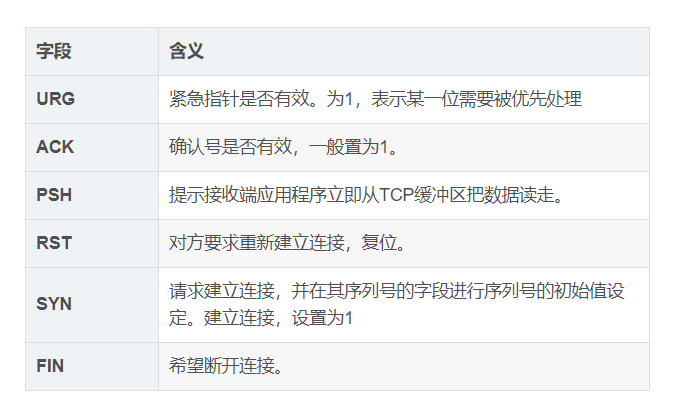
    确认号ack：占4个字节，期待收到对方下一个报文段的第一个数据字节的序号；序列号表示报文段携带数据的第一个字节的编号；而确认号指的是期望接收到下一个字节的编号；因此当前报文段最后一个字节的编号+1即为确认号。

    确认ACK：占1位，仅当ACK=1时，确认号字段才有效。ACK=0时，确认号无效

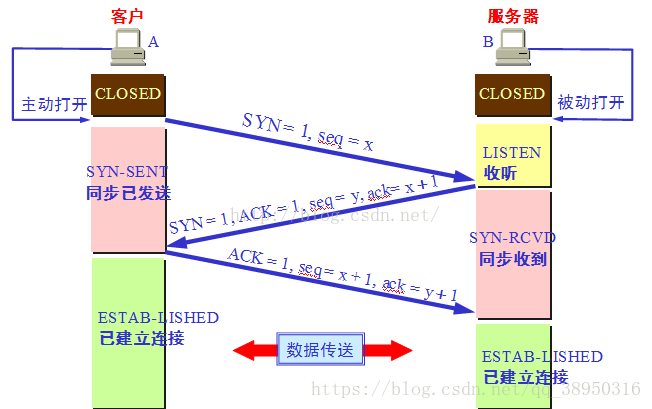
    同步SYN：连接建立时用于同步序号。当SYN=1，ACK=0时表示：这是一个连接请求报文段。若同意连接，则在响应报文段中使得SYN=1，ACK=1。因此，SYN=1表示这是一个连接请求，或连接接受报文。SYN这个标志位只有在TCP建产连接时才会被置1，握手完成后SYN标志位被置0。

    终止FIN：用来释放一个连接。FIN=1表示：此报文段的发送方的数据已经发送完毕，并要求释放运输连接

    PS：ACK、SYN和FIN这些大写的单词表示标志位，其值要么是1，要么是0；ack、seq小写的单词表示序号。



**三次握手过程理解**

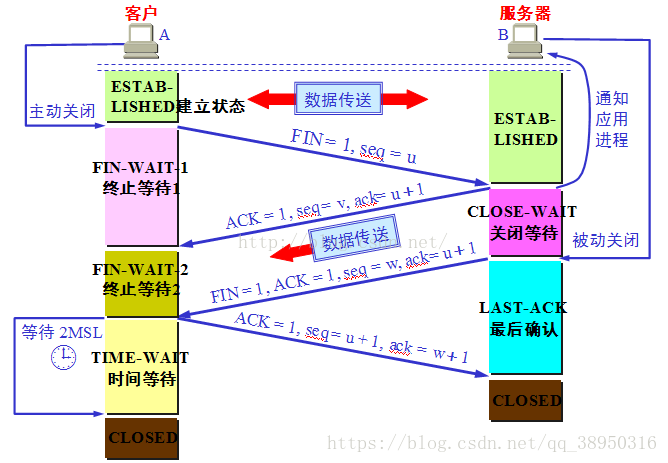


第一次握手：建立连接时，客户端发送syn包（syn=j）到服务器，并进入SYN\_SENT状态，等待服务器确认；SYN：同步序列编号（Synchronize Sequence Numbers）。

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=k+1），此包发送完毕，客户端和服务器进入ESTABLISHED（TCP连接成功）状态，完成三次握手。

四次挥手过程理解

  
1）客户端进程发出连接释放报文，并且停止发送数据。释放数据报文首部，FIN=1，其序列号为seq=u（等于前面已经传送过来的数据的最后一个字节的序号加1），此时，客户端进入FIN-WAIT-1（终止等待1）状态。 TCP规定，FIN报文段即使不携带数据，也要消耗一个序号。  
2）服务器收到连接释放报文，发出确认报文，ACK=1，ack=u+1，并且带上自己的序列号seq=v，此时，服务端就进入了CLOSE-WAIT（关闭等待）状态。TCP服务器通知高层的应用进程，客户端向服务器的方向就释放了，这时候处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端依然要接受。这个状态还要持续一段时间，也就是整个CLOSE-WAIT状态持续的时间。  
3）客户端收到服务器的确认请求后，此时，客户端就进入FIN-WAIT-2（终止等待2）状态，等待服务器发送连接释放报文（在这之前还需要接受服务器发送的最后的数据）。  
4）服务器将最后的数据发送完毕后，就向客户端发送连接释放报文，FIN=1，ack=u+1，由于在半关闭状态，服务器很可能又发送了一些数据，假定此时的序列号为seq=w，此时，服务器就进入了LAST-ACK（最后确认）状态，等待客户端的确认。  
5）客户端收到服务器的连接释放报文后，必须发出确认，ACK=1，ack=w+1，而自己的序列号是seq=u+1，此时，客户端就进入了TIME-WAIT（时间等待）状态。注意此时TCP连接还没有释放，必须经过2∗∗MSL（最长报文段寿命）的时间后，当客户端撤销相应的TCB后，才进入CLOSED状态。  
6）服务器只要收到了客户端发出的确认，立即进入CLOSED状态。同样，撤销TCB后，就结束了这次的TCP连接。可以看到，服务器结束TCP连接的时间要比客户端早一些。

**常见面试题**  
【问题1】为什么连接的时候是三次握手，关闭的时候却是四次握手？

答：因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

【问题2】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？

答：虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假象网络是不可靠的，有可以最后一个ACK丢失。所以TIME\_WAIT状态就是用来重发可能丢失的ACK报文。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。

【问题3】为什么不能用两次握手进行连接？

答：3次握手完成两个重要的功能，既要双方做好发送数据的准备工作(双方都知道彼此已准备好)，也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。

       现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子，考虑计算机S和C之间的通信，假定C给S发送一个连接请求分组，S收到了这个分组，并发 送了确认应答分组。按照两次握手的协定，S认为连接已经成功地建立了，可以开始发送数据分组。可是，C在S的应答分组在传输中被丢失的情况下，将不知道S 是否已准备好，不知道S建立什么样的序列号，C甚至怀疑S是否收到自己的连接请求分组。在这种情况下，C认为连接还未建立成功，将忽略S发来的任何数据分 组，只等待连接确认应答分组。而S在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

【问题4】如果已经建立了连接，但是客户端突然出现故障了怎么办？

TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为2小时，若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔75秒钟发送一次。若一连发送10个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

https://www.cnblogs.com/bj-mr-li/p/11106390.html

作者：山尽  
链接：https://www.zhihu.com/question/24853633/answer/573627478  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

**三次握手的误解与错误类比(RFC解读)**

关于TCP三次握手几乎是应届毕业生面试常见的问题了，然而网上还很多比比皆是的错误，以知乎 [TCP 为什么是三次握手，而不是两次或四次？](https://www.zhihu.com/question/24853633) 上的热门答案为例子，第一个3.6K 次赞同的类比就是错误的：

三次握手：

“喂，你听得到吗？”

“我听得到呀，你听得到我吗？”

“我能听到你，今天 balabala……”

同样这个107次赞同的类比也是错误的：

握手和敬军礼一样，源自「敌我双方互相确认对方手里没有武器、无恶意」的仪式。（虽然双方互相请求确认需要四步，但由于中间的确认和请求是由同一个人执行的，所以合并成了一步）

正恩伸出手说：你看，我手里没有武器。（SYN）

朗普看了看说：嗯，确实没有。（ACK）

于是也伸出手说：你看，我手里也没有武器。（SYN）

正恩看了看说：嗯，看来你确实有诚意。（ACK）

这两个类比就是想当然的错误，为什么会错误，看完全文相信你便了然于心。

另外还有一个就是在谢希仁著《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”，这个只能算是表因，并不涉及本质。

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。”

如果你细读[RFC793](https://link.zhihu.com/?target=https%3A//www.ietf.org/rfc/rfc793.txt" \t "_blank)，也就是 TCP 的协议 RFC，你就会发现里面就讲到了为什么三次握手是必须的——TCP 需要 seq 序列号来做可靠重传或接收，而避免连接复用时无法分辨出 seq 是延迟或者是旧链接的 seq，因此需要三次握手来约定确定双方的 ISN（初始 seq 序列号）。

下面给出详细的 RFC 解读说明：（数据分组称为分段（Segment），国内通常用包来称呼）

我们首先要知道到一点就是， TCP 的可靠连接是靠 seq（ sequence numbers 序列号）来达成的。

A fundamental notion in the design is that every octet of data sent over a TCP connection has a sequence number. Since every octet is sequenced, each of them can be acknowledged. The acknowledgment mechanism employed is cumulative so that an acknowledgment of sequence number X indicates that all octets up to but not including X have been received.

TCP 设计中一个基本设定就是，通过TCP 连接发送的每一个包，都有一个sequence number。而因为每个包都是有序列号的，所以都能被确认收到这些包。

确认机制是累计的，所以一个对sequence number X 的确认，意味着 X 序列号之前(不包括 X) 包都是被确认接收到的。

The protocol places no restriction on a particular connection being used over and over again.   
The problem that arises from this is -- "how does the TCP identify duplicate segments from previous incarnations of the connection?" This problem becomes apparent if the connection is being opened and closed in quick succession, or if the connection breaks with loss of memory and is then reestablished.

TCP 协议是不限制一个特定的连接（两端 socket 一样）被重复使用的。

所以这样就有一个问题：这条连接突然断开重连后，TCP 怎么样识别之前旧链接重发的包？——这就需要独一无二的 ISN（初始序列号）机制。

When new connections are created, an initial sequence number (ISN) generator is employed which selects a new 32 bit ISN. The generator is bound to a (possibly fictitious) 32 bit clock whose low order bit is incremented roughly every 4 microseconds. Thus, the ISN cycles approximately every 4.55 hours. Since we assume that segments will stay in the network no more than the Maximum Segment Lifetime (MSL) and that the MSL is less than 4.55 hours we can reasonably assume that ISN's will be unique.

当一个新连接建立时，初始序列号（ initial sequence number ISN）生成器会生成一个新的32位的 ISN。

这个生成器会用一个32位长的时钟，差不多4µs 增长一次，因此 ISN 会在大约 4.55 小时循环一次

（2^32位的计数器，需要2^32\*4 µs才能自增完，除以1小时共有多少µs便可算出2^32\*4 /(1\*60\*60\*1000\*1000)=4.772185884 ）

而一个段在网络中并不会比最大分段寿命（Maximum Segment Lifetime (MSL) ，默认使用2分钟）长，MSL 比4.55小时要短，所以我们可以认为 ISN 会是唯一的。

发送方与接收方都会有自己的 ISN （下面的例子中就是 X 与 Y）来做双方互发通信，具体的描述如下：

1) A --> B SYN my sequence number is X 2) A <-- B ACK your sequence number is X 3) A <-- B SYN my sequence number is Y 4) A --> B ACK your sequence number is Y

2与3都是 B 发送给 A，因此可以合并在一起，因此成为three way (or three message) handshake（其实翻译为三步握手，或者是三次通信握手更为准确）

因此最终可以得出，三次握手是必须的：

A three way handshake is necessary because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISN's. The receiver of the first SYN has no way of knowing whether the segment was an old delayed one or not, unless it remembers the last sequence number used on the connection (which is not always possible), and so it must ask the sender to verify this SYN. The three way handshake and the advantages of a clock-driven scheme are discussed in [3].

三次握手（A three way handshake）是必须的， 因为 sequence numbers（序列号）没有绑定到整个网络的全局时钟（全部统一使用一个时钟，就可以确定这个包是不是延迟到的）以及 TCPs 可能有不同的机制来选择 ISN（初始序列号）。

接收方接收到第一个 SYN 时，没有办法知道这个 SYN 是是否延迟了很久了，除非他有办法记住在这条连接中，最后接收到的那个sequence numbers（然而这不总是可行的）。

这句话的意思是：一个 seq 过来了，跟现在记住的 seq 不一样，我怎么知道他是上条延迟的，还是上上条延迟的呢？

所以，接收方一定需要跟发送方确认 SYN。

假设不确认 SYN 中的 SEQ，那么就只有：

1) A --> B SYN my sequence number is X 2) A <-- B ACK your sequence number is X SYN my sequence number is Y

只有B确认了收到了 A 的 SEQ， A 无法确认收到 B 的。也就是说，只有 A 发送给 B 的包都是可靠的， 而 B 发送给 A 的则不是，所以这不是可靠的连接。这种情况如果只需要 A 发送给 B ，B 无需回应，则可以不做三次握手。

所以，正确的类比应该是这样的：TCP 传递信息可以理解为美国与中国用货船来传货物，但因为一首轮船穿放不下，货物要分开一只只轮船来发货。

所以需要一个序列号来识别该货物是第几个，以便到达后将其拼接回原来的货物。

因为同一条航道（也就是 tcp连接）上，可能会有多批货物发送（复用 tcp 连接）。发货时，双方需要通知对方这个序列号是从哪里开始（init seq）的，这样才能辨识过来的是不是一个对的货物，以及能拼接成完整的货物。

货物运输拼接（tcp）最重要的是可靠性，如果没有用三次握手来确认双方都可以获得对方的 序列号（seq）的话，就无法知道当前航班（连接）中，对的货物序号是怎么样的了。

**三次握手详细过程**

TCP A TCP B

​

1. CLOSED LISTEN

​

2. SYN-SENT --> <SEQ=100><CTL=SYN> --> SYN-RECEIVED

​

3. ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK> <-- SYN-RECEIVED

​

4. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK> --> ESTABLISHED

​

5. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK><DATA> --> ESTABLISHED

​

Basic 3-Way Handshake for Connection Synchronization

​

Figure 7.

在上图

* 第二行中， A 发送了 SEQ 100，标志位是 SYN；
* 第三行，B 发回了 ACK 101 与 SEQ 300，标志位是 SYN 与 ACK（两个过程合并了）。注意，ACK 是101意味着，B 希望接收到 101序列号开始的数据段。
* 第四行，A 返回了空的数据，SEQ 101， ACK 301，标志位为 ACK。至此，双方的开始 SEQ （也就是 ISN）号100与300都被确认接收到了。
* 第五行，开始正式发送数据包，注意的是 ACK 依旧是第四行的301，因为没有需要 ACK 的 SYN 了（第四行已经 ACK 完）。

以上，4 最后这个确认的过程，是可以带上数据的。

The principle reason for the three-way handshake is to prevent old

duplicate connection initiations from causing confusion. To deal with

this, a special control message, reset, has been devised. If the

receiving TCP is in a non-synchronized state (i.e., SYN-SENT,

SYN-RECEIVED), it returns to LISTEN on receiving an acceptable reset.

If the TCP is in one of the synchronized states (ESTABLISHED,

FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), it

aborts the connection and informs its user. We discuss this latter

case under "half-open" connections below.

三次握手的原则设计是防止旧复用链接的初始化导致问题，为了解决此问题，我们设计了reset这个特别的控制信号来处理。

如果接收中的 TCP 在一个未同步状态如 SYN-SENT, SYN-RECEIVED，它会返回 reset 给对方。

如果 TCP 是同步状态中如(ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT)，他会终止此连接并通知用户。

看起来有点绕，我们举个图例看看：

TCP A TCP B

​

1. CLOSED LISTEN

​

2. SYN-SENT --> <SEQ=100><CTL=SYN> ...

​

3. (duplicate) ... <SEQ=90><CTL=SYN> --> SYN-RECEIVED

​

4. SYN-SENT <-- <SEQ=300><ACK=91><CTL=SYN,ACK> <-- SYN-RECEIVED

​

5. SYN-SENT --> <SEQ=91><CTL=RST> --> LISTEN

​

6. ... <SEQ=100><CTL=SYN> --> SYN-RECEIVED

​

7. SYN-SENT <-- <SEQ=400><ACK=101><CTL=SYN,ACK> <-- SYN-RECEIVED

​

8. ESTABLISHED --> <SEQ=101><ACK=401><CTL=ACK> --> ESTABLISHED

​

Recovery from Old Duplicate SYN

这是 复用连接时，旧在途包发往新连接中的例子。

* 3中，一个旧的重复的 SYN到达 B。
* 4中， B分别不出是否旧的，照样子正常回包。
* 5中，A检测到 B 返回的ACK不正确，所以返回 RST(reset)
* 6中，B接收到 RST(reset)信号，于是变成 LISTEN 状态。
* 7中，新连接正常的 SYN终于到达了，三次握手正常进行。

这种是简化的情况，但是可以看出 TCP 是如何处理复用旧链接的包到达的。

ps：关于全局时钟的方法，我找到了一篇论文：[http://mirrors.ustc.edu.cn/rfc/ien/ien193.pdf](https://link.zhihu.com/?target=http%3A//mirrors.ustc.edu.cn/rfc/ien/ien193.pdf), 有兴趣的同学可以看看。

在这里要感谢 余晟以为 这个公众号的文章，里面一句提到rfc里有 三次握手是必须的原因，因此我顺势把rfc看了一下。

还有非常感谢我的 mentor，在我的编程生涯里，以身作则的给我做了一个非常棒的榜样，也修正了我很多编程的坏习惯。 - = - 怎么改正的请看：[《计算机网络》这门课为何如此之难？](https://www.zhihu.com/question/19718686/answer/575028538)