### 建立连接：

1. 客户端（连接发起者）发送一个SYN报文段，并指明自己想要连接的端口号和它的客户端初始序列号（ISN(c)）。通常还会借此发送一个或多个选项。进入SYN\_SEND状态。
2. 服务器发送自己的SYN报文作为回应，并包含了它的初始序列号（ISN(s)）。并将ISN(c)的数值+1作为确认号（ACK）。进入SYN\_RECV状态。
3. 客户端发送ACK消息，为ISN(s)+1。进入ESTABLISHED状态。
4. 服务器接收到ACK后也进入ESTABLISHED状态。

三次握手的目的：建立连接，防止服务器资源浪费，交换双方初始序列号（Initial Sequence Number，ISN）。

### 关闭连接：

1. 主动关闭者主机1发送一个FIN段，并指明序列号K，和一个ACK用于确认最近一次发来的数据。进入FIN\_WAIT\_1状态。
2. 被动关闭者主机2收到FIN后将K+1作为ACK返回。进入CLOSED\_WAIT状态。
3. 主机1接收到FIN的ACK后进入FIN\_WAIT\_2状态。
4. 主机2发送FIN段，并指明自己的序列号L，和ACK用于确认上一个数据。进入LAST\_ACK状态。
5. 主机1接收到FIN后，返回ACK，将L+1做为序列号。进入TIME\_WAIT状态。
6. 主机2接收到ACK后进入CLOSED状态。主机1在等待2MSL后若仍未接收到奥主机2 的消息，则认为主机2已关闭，进入CLOSED状态。

7个报文段是每一个TCP连接在正常建立与关闭时的基本开销。

### 其它：

1. TCP半关闭：被动关闭者在发送完第一个FIN的ACK后，可以继续发送任意数量的数据，此时的状态称为半关闭状态。虽然TCP支持半关闭状态，但很少有应用程序使用这一特性。
2. 同时打开与关闭：在特定安排情况下，两个应用程序同时主动打开/关闭连接。此时打开和关闭过程都只需要4个报文段，双方的打开/关闭请求及双方的ACK。

### 初始化序列号：

TCP报文在经过路由传播后，可能出现延迟抵达与排序混乱的情况，为了解决这个问题，需要仔细选择初始序列号，防止出现与其它连接序列号重叠的情况。

一个TCP连接是被2个IP及两个端口号唯一确定的，所以一个TCP连接可能有多个实例，为了避免实例间序列号重叠的问题，也需要选择合适的初始序列号。

为了降低风险，也可以在应用层使用合适的方法校验数据。

如果选择合适的序列号、ip及端口，那么任何人都能伪造出一个TCP报文段，从而打断正常连接。为了防止伪造报文，首先可以使初始序列号变得难以猜出，另一种方式是加密。

现代系统通常采用半随机的方法选择初始化序列号。

### TCP选项：



1. EOL：指明选项列表的结尾，后续的选项无需再处理。
2. NOP：允许发送者在必要的时候多4个字节填充某个选项。
3. MSS：最大段大小，只记录TCP数据的字节数，不包括头部。在建立连接的时候，通信的双方都要在SYN报文的MSS选项中说明自己允许的最大段大小，默认为536字节。

IPV4中的典型值为1460字节，IPV6则为1440（头部比IPV4多20字节），加上头部整个报文段的大小为1500字节。

最大段的大小不是双方协商的结果，而是双方指定的一个限定的数值。

1. WSOPT：窗口缩放选项。将TCP窗口大小字段的范围由16增加到30位。TCP头部无需改变窗口大小字段的大小，仍然维持16位，使用WSOPT字段作为该字段的比例因子。该比例因子使用0-14来指定原窗口字段值左移位数，最大能提供65535\*2^14的窗口，接近2^30-1（1GB）。

窗口的移动数值是由TCP通信方根据接收缓存的大小自动选取的，缓存的大小是由系统设定的，但是应用程序通常都有改变其大小的能力。

1. SACK：选择确认选项。包含了接收方已经成功接收的数据块的序列号范围，每个范围被称作一个SACK块，由一对32位的序列号表示。

由于TCP头部选项空间有限，因此一个报文段中最大SACK块数目为3。

1. TSOPT：时间戳选项。要求发送方在每个报文段中添加两个4字节的时间戳数值，发送方将一个32位的数值填充到第一部分，接收方将收到的时间戳原封不动地填充到第二部分。

因此发送方可以根据接收到的每个ACK中的时间戳，估算出TCP的往返时间RTT，并由此作为超时重传时间确定的依据之一。

同时可以作为新报文和旧报文的判断依据。序列号只有32位，所以会出现序列号相同的情况，可以通过时间戳防止序号绕回。

1. UTO：用户超时选项。指明了TCP发送者愿意等待ACK的时间。这是一个建议性的数值，另一端不必须遵从。（未广泛使用）
2. TCP-AO：认证选项。为了防御各种TCP欺骗攻击，双方借助通信密钥，确认TCP报文是否被篡改过。（未广泛使用）

### 最大传输路径发现：

TCP的报文段大小，很大程度上控制了生成的IP数据报。在底层传输的单个数据报大小有限制，如在以太网中，携带的帧的大小被限制为1500字节以内。

连接建立时，TCP使用对外接口的最大传输单元的最小值，或者根据对方声明的最大段大小（MSS）作为最大段大小。

若接收到ICMP的PTB消息（ICMPv4地址不可达<需要分片>或ICMPv6中数据包太大<packet to big>），TCP就会减少数据报大小，然后用修改后的报文段进行重传。如果在PTB消息中已经包含了下一跳推荐的最大传输单元，则参照该大小进行重传。否则需要尝试多个数值，使用二分法选择一个合适的值。

由于路由是动态变化的，在减少段大小的一段时间后，需要尝试一个更大的值（接近初始的发送方最大段大小）。

在TCP依赖与ICMP来调整段大小的情况下，若ICMP被禁止转发或其它原因导致无法接收到ICMP消息，则一旦TCP采用更大的段大小时，会导致大的数据包无法被传输，而只能传输小的数据包，导致服务出现问题，而这种问题很难进行排查。

### TCP连接状态：

## TIME\_WAIT状态：

也被称为2MSL等待状态，主机会等待两倍最大段生存期的时间才进入CLOSED状态。

MSL代表任何报文段在被丢弃前在网络中被允许存在的最长时间。TCP报文以IP数据报传输，而IP数据报有TTL字段和跳数限制字段，限制了其生存时间。

然而现实中该数值是可以修改的，linux系统中net.ipv4.tcp\_fin\_timeout记录了TIME\_WAIT状态需要等待的超时时间。Windows系统也是一样。

若最后的ACK丢失，则重传ACK，此行为不是主动的，而是因为接收到了重传的FIN报文。

在TIME\_WAIT状态下，通信双方将连接定为不可用，其它主机若要建立连接需要等待，直到TIME\_WAIT状态结束。但许多API及实现中提供了绕开该机制的方法，使得即使处于TIME\_WAIT状态也可立即进行绑定。

## FIN\_WAIT\_2状态：

进入FIN\_WAIT\_2状态后，若为非半关闭操作，且未接收到FIN报文时，主机会无限等待。

为了防止这种情况出现，有各种实现方式，如linux，若为非半关闭操作，则设置一个计时器，若计时器超时时连接是空闲的，那么直接进入CLOSED状态。可以通过调整net.ipv4.tcp\_fin\_timeout来设置计数器，默认为60s。