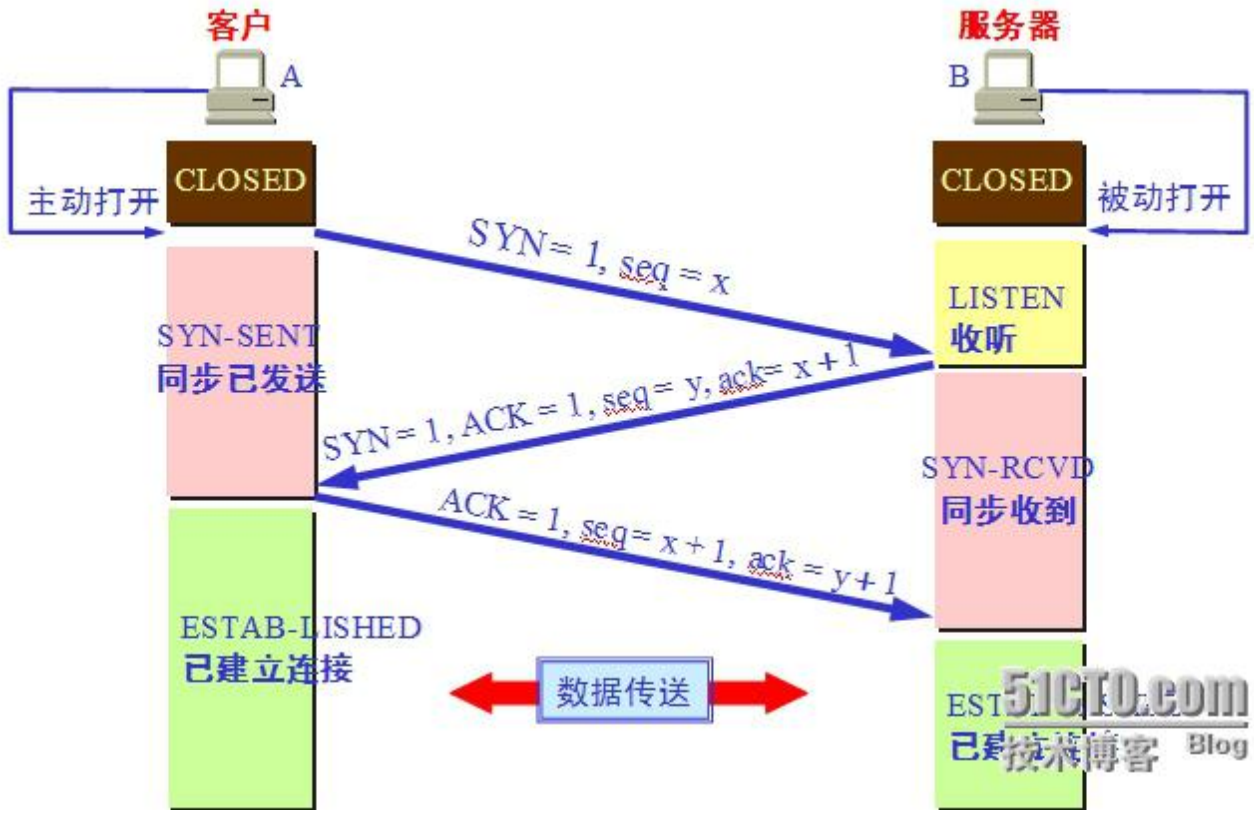
大家对 netstat -a 命令很熟悉吧，但是，你有没有注意到 STATE 一栏呢，基本上显示着 established,time_wait,close_wait 等，这些到底是什么意思呢，在这篇文章，我将会详细的阐述。

大家很明白 **TCP 初始化连接三次握手**吧：发 SYN 包，然后返回 SYN/ACK 包，再发 ACK 包，连接正式建立。如下图：



但是大家明白关闭连接的工作原理吗？

关闭连接要四次握手：发 FIN 包，ACK 包，FIN 包，ACK 包，四次握手！！

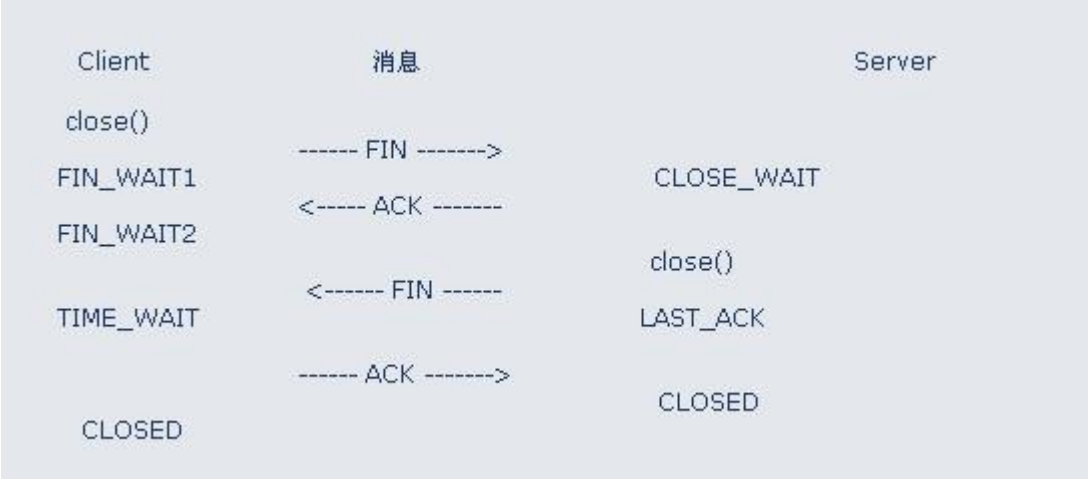
为什么呢，因为 TCP 连接是全双工，我关了你的连接，并不等于你关了我的连接。

客户端 TCP 状态迁移：
CLOSED->SYN_SENT->ESTABLISHED->FIN_WAIT_1->FIN_WAIT_2->TIME_WAIT->CLOSED
服务器 TCP 状态迁移：
CLOSED->LISTEN->SYN_RECEIVED->ESTABLISHED->CLOSE_WAIT->LAST_ACK->CLOSED
当客户端开始连接时，服务器还处于 LISTENING，客户端发一个 SYN 包后，他就处于 SYN_SENT 状态,服务器就处于 SYS 收到状态,然后互相确认进入连接状态 ESTABLISHED.

当客户端请求关闭连接时,客户端发送一个 FIN 包后,客户端就进入 FIN_WAIT_1 状态,等待

对方的确认包,服务器发送一个 ACK 包给客户,客户端收到 ACK 包后结束 FIN_WAIT_1 状态,进入 FIN_WAIT_2 状态,等待服务器发过来的关闭请求,服务器发一个 FIN 包后,进入 CLOSE_WAIT 状态,当客户端收到服务器的 FIN 包,FIN_WAIT_2 状态就结束,然后给服务器端的 FIN 包给以一个确认包,客户端这时进入 TIME_WAIT,当服务器收到确认包后,CLOSE_WAIT 状态结束了,这时候服务器端真正的关闭了连接.但是客户端还在 TIME_WAIT 状态下,什么时候结束呢.我在这里再讲到一个新名词:2MSL 等待状态,其实 **TIME_WAIT 就是 2MSL 等待状态**,为什么要设置这个状态,原因是有足够的时间让 ACK 包到达服务器端,如果服务器端没收到 ACK 包，超时了，然后重新发一个 FIN 包，直到服务器收到 ACK 包，TIME_WAIT 状态等待时间是在 TCP 重新启动后不连接任何请求的两倍.
大家有没有发现一个问题:如果对方在第三次握手的时候出问题,如发 FIN 包的时候,不知道什么原因丢了这个包,然而这边一直处在 FIN_WAIT_2 状态,而且 TCP/IP 并没有设置这个状态的过期时间,那他一直会保留这个状态下去,越来越多的 FIN_WAIT_2 状态会导致系统崩溃.

上面我碰到的这个问题主要因为 TCP 的结束流程未走完，造成连接未释放。现设客户端主动断开连接，流程如下



如上图所示，

Client 消息 Server

close()
------ FIN ------->
FIN_WAIT1 CLOSE_WAIT
<----- ACK -------
FIN_WAIT2
close()

<------ FIN ------

TIME_WAIT LAST_ACK

------ ACK ------->

CLOSED

CLOSED

由于 Server 的 Socket 在客户端已经关闭时而没有调用关闭，
造成服务器端的连接处在"挂起"状态，而客户端则处在等待应答的状态上。
此问题的典型特征是：
一端处于 FIN_WAIT2 ，而另一端处于 CLOSE_WAIT.

不过，根本问题还是程序写的不好，有待提高

CLOSE_WAIT，TCP 的癌症，TCP 的朋友。

CLOSE_WAIT 状态的生成原因
首先我们知道，如果我们的服务器程序 APACHE 处于 CLOSE_WAIT 状态的话，说明套接字是被动关闭的！

因为如果是 CLIENT 端主动断掉当前连接的话，那么双方关闭这个 TCP 连接共需要四个 packet：

Client ---> FIN ---> Server

Client <--- ACK <--- Server

这时候 Client 端处于 FIN_WAIT_2 状态；而 Server 程序处于 CLOSE_WAIT 状态。

Client <--- FIN <--- Server

这时 Server 发送 FIN 给 Client，Server 就置为 LAST_ACK 状态。

Client ---> ACK ---> Server

Client 回应了 ACK，那么 Server 的套接字才会真正置为 CLOSED 状态。

Server 程序处于 CLOSE_WAIT 状态，而不是 LAST_ACK 状态，说明还没有发 FIN 给 Client，那么可能是在关闭连接之前还有许多数据要发送或者其他事要做，导致没有发这个 FIN packet。

通常来说，一个 CLOSE_WAIT 会维持至少 2 个小时的时间。如果有个流氓特地写了个程序，给你造成一堆的 CLOSE_WAIT，消耗

你的资源，那么通常是等不到释放那一刻，系统就已经解决崩溃了。

只能通过修改一下 TCP/IP 的参数，来缩短这个时间：修改 tcp_keepalive_*系列参数有助于解决这个问题。

连接进程是通过一系列状态表示的，这些状态有：

LISTEN，SYN-SENT，SYN-RECEIVED，ESTABLISHED，FIN-WAIT-1，FIN-WAIT-2，CLOSE-WAIT，CLOSING，LAST-ACK，TIME-WAIT 和 CLOSED。

各个状态的意义如下：

CLOSED- 初始状态，表示 TCP 连接是"关闭着的"或"未打开的"。

LISTEN - 表示服务器端的某个 SOCKET 处于监听状态，可以接受客户端的连接。

SYN-SENT - 这个状态与 SYN_RCVD 状态相呼应，当客户端 SOCKET 执行 connect()进行连接时，它首先发送 SYN 报文，然后随即进入到 SYN_SENT 状态，并等待服务端的发送三次握手中的第 2 个报文。SYN_SENT 状态表示客户端已发送 SYN 报文。

SYN-RCVD - 表示接收到了 SYN 报文。在正常情况下，这个状态是服务器端的 SOCKET 在建立 TCP 连接时的三次握手会话过程中的一个中间状态，很短暂，基本上用 netstat 很难看到这种状态，除非故意写一个监测程序，将三次 TCP 握手过程中最后一个 ACK 报文不予发送。当 TCP 连接处于此状态时，再收到客户端的 ACK 报文，它就会进入到 ESTABLISHED 状态。

ESTABLISHED- 表示 TCP 连接已经成功建立，数据可以传送给用户；

FIN-WAIT-1 - 这个状态得好好解释一下，其实 FIN_WAIT_1 和 FIN_WAIT_2 两种状态的真正含义都是表示等待对方的 FIN 报文。而这两种状态的区别是：FIN_WAIT_1 状态实际上是当 SOCKET 在 ESTABLISHED 状态时，它想主动关闭连接，向对方发送了 FIN 报文，此时该 SOCKET 进入到 FIN_WAIT_1 状态。而当对方回应 ACK 报文后，则进入到 FIN_WAIT_2 状态。当然在实际的正常情况下，无论对方处于任何种情况下，都应该马上回应 ACK 报文，所以 FIN_WAIT_1 状态一般是比较难见到的，而 FIN_WAIT_2 状态有时仍可以用 netstat 看到。

FIN-WAIT-2 - 上面已经解释了这种状态的由来，实际上 FIN_WAIT_2 状态下的 SOCKET 表示半连接，即有一方调用 close()主动要求关闭连接。注意：FIN_WAIT_2 是没有超时的（不像 TIME_WAIT 状态），这种状态下如果对方不关闭（不配合完成 4 次挥手过程），那这个 FIN_WAIT_2 状态将一直保持到系统重启，越来越多的 FIN_WAIT_2 状态会导致内核崩溃。

CLOSE-WAIT - 表示正在等待关闭。怎么理解呢？当对方 close()一个 SOCKET 后发送 FIN 报文给自己，你的系统毫无疑问地将会回应一个 ACK 报文给对方，此时 TCP 连接则进入到

CLOSE_WAIT 状态。接下来呢，你需要检查自己是否还有数据要发送给对方，如果没有的话，那你也就可以 close()这个 SOCKET 并发送 FIN 报文给对方，即关闭自己到对方这个方向的连接。有数据的话则看程序的策略，继续发送或丢弃。简单地说，当你处于 CLOSE_WAIT 状态下，需要完成的事情是等待你去关闭连接。

CLOSING - 这种状态在实际情况中应该很少见，属于一种比较罕见的例外状态。正常情况下，当一方发送 FIN 报文后，按理来说是应该先收到（或同时收到）对方的 ACK 报文，再收到对方的 FIN 报文。但是 CLOSING 状态表示一方发送 FIN 报文后，并没有收到对方的 ACK 报文，反而却也收到了对方的 FIN 报文。什么情况下会出现此种情况呢？那就是当双方几乎在同时 close()一个 SOCKET 的话，就出现了双方同时发送 FIN 报文的情况，这时就会出现 CLOSING 状态，表示双方都正在关闭 SOCKET 连接。

LAST-ACK - 当被动关闭的一方在发送 FIN 报文后，等待对方的 ACK 报文的时候，就处于 LAST_ACK 状态。当收到对方的 ACK 报文后，也就可以进入到 CLOSED 可用状态了。

TIME-WAIT - 等待足够的时间以确保远程 TCP 接收到连接中断请求的确认； 表示收到了对方的 FIN 报文，并发送出了 ACK 报文。TIME_WAIT 状态下的 TCP 连接会等待 2*MSL（Max Segment Lifetime，最大分段生存期，指一个 TCP 报文在 Internet 上的最长生存时间。每个具体的 TCP 协议实现都必须选择一个确定的 MSL 值，RFC 1122 建议是 2 分钟，但 BSD 传统实现采用了 30 秒，Linux 可以 cat /proc/sys/net/ipv4/tcp_fin_timeout 看到本机的这个值），然后即可回到 CLOSED 可用状态了。如果 FIN_WAIT_1 状态下，收到了对方同时带 FIN 标志和 ACK 标志的报文时，可以直接进入到 TIME_WAIT 状态，而无须经过 FIN_WAIT_2 状态。

TCP 连接过程是状态的转换，促使发生状态转换的是用户调用：OPEN，SEND，RECEIVE，CLOSE，ABORT 和 STATUS；传送过来的数据段，特别那些包括以下标记的数据段 SYN，ACK，RST 和 FIN；还有超时，上面所说的都会使 TCP 状态发生变化。

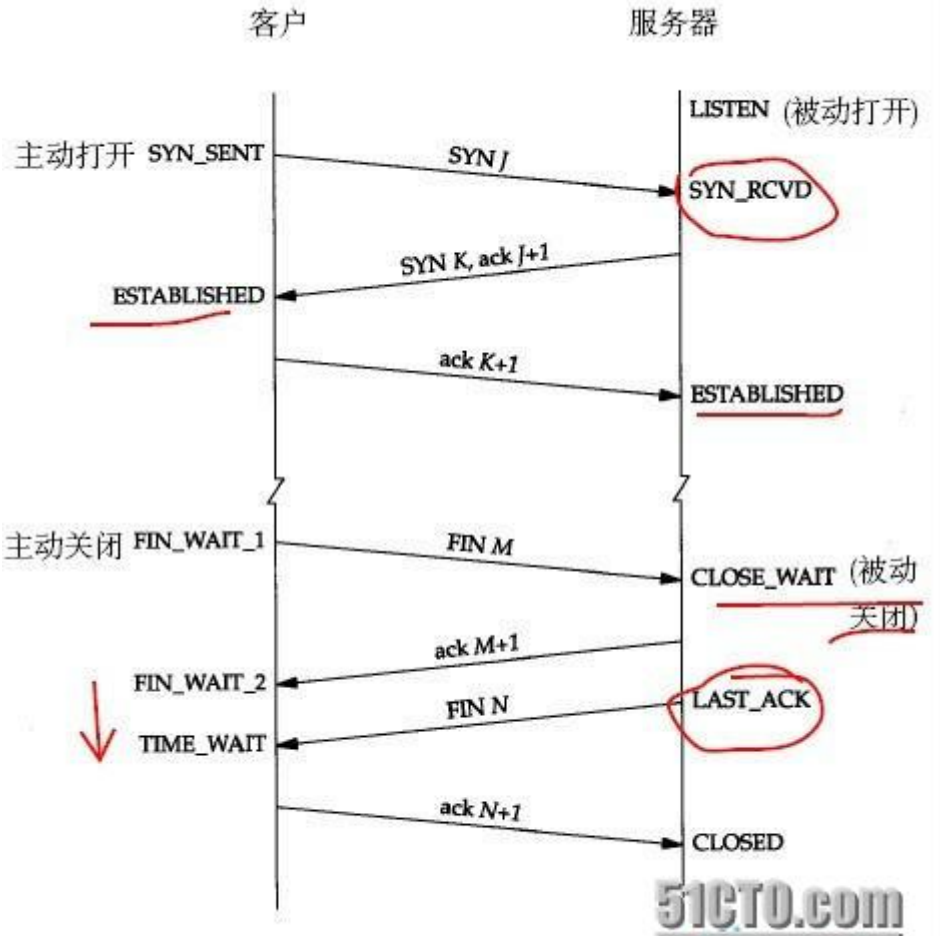
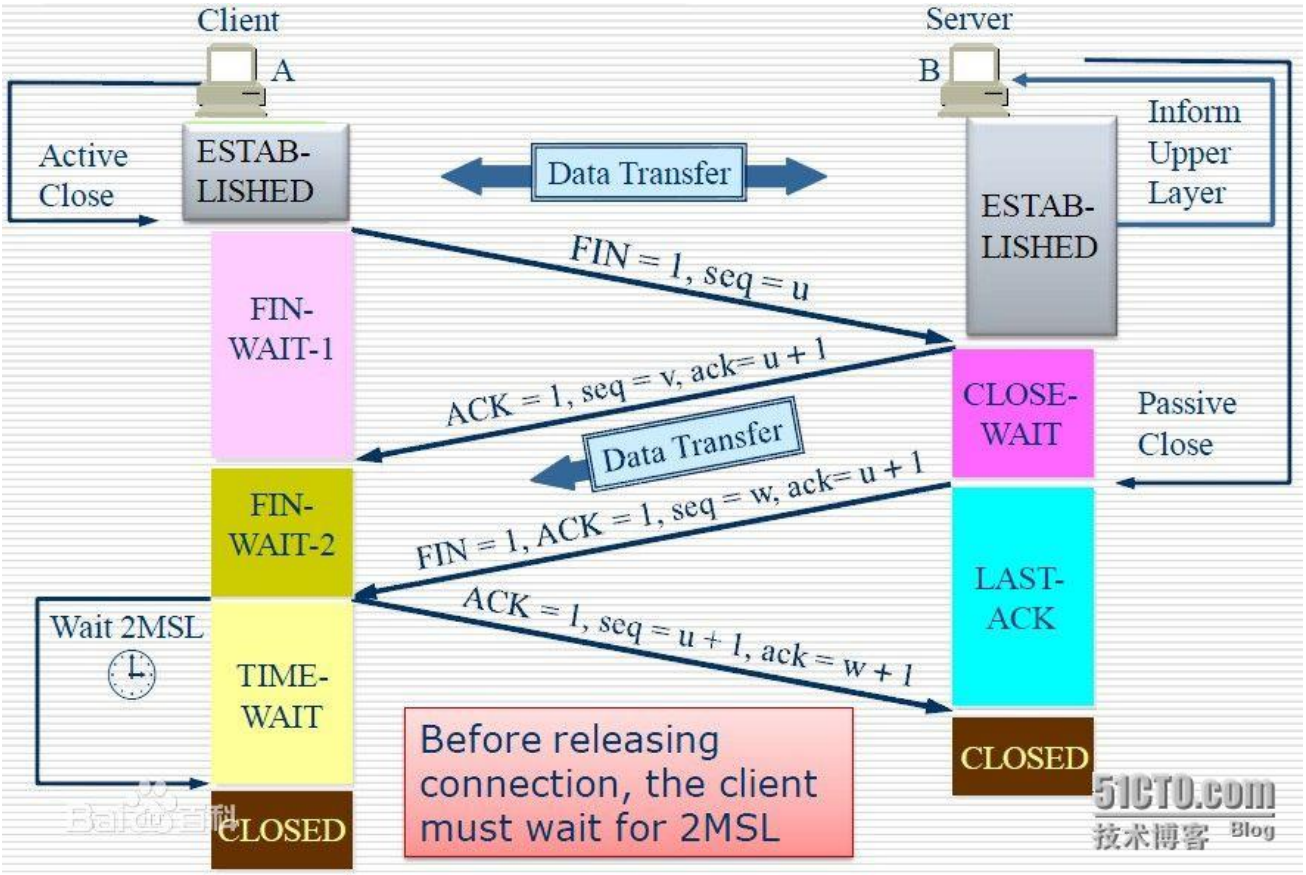


图18-13 TCP正常连接建立和终止所对应的状态



TCP 连接的状态转换图

这个图 N 多人都知道，它对排除和定位网络或系统故障时大有帮助，但是怎样牢牢地将这张图刻在脑中呢？那么你就一定要对这张图的每一个状态，及转换的过程有深刻地认识，不能只停留在一知半解之中。下面对这张图的 11 种状态详细解释一下，以便加强记忆！不过在这之前，先回顾一下 TCP 建立连接的三次握手过程，以及关闭连接的四次握手过程。

1、建立连接协议（三次握手）

（1）客户端发送一个带 SYN 标志的 TCP 报文到服务器。这是三次握手过程中的报文 1。

（2） 服务器端回应客户端的，这是三次握手中的第 2 个报文，这个报文同时带 ACK 标志和 SYN 标志。因此它表示对刚才客户端 SYN 报文的回应；同时又标志 SYN 给客户端，询问客户端是否准备好进行数据通讯。

（3） 客户必须再次回应服务段一个 ACK 报文，这是报文段 3。

2、连接终止协议（四次握手）

由于 TCP 连接是全双工的，因此每个方向都必须单独进行关闭。这原则是当一方完成它的数据发送任务后就能发送一个 FIN 来终止这个方向的连接。收到一个 FIN 只意味着这

一方向上没有数据流动，一个 TCP 连接在收到一个 FIN 后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。

（1）TCP 客户端发送一个 FIN，用来关闭客户到服务器的数据传送（报文段 4）。

（2） 服务器收到这个 FIN，它发回一个 ACK，确认序号为收到的序号加 1（报文段 5）。和 SYN 一样，一个 FIN 将占用一个序号。

（3） 服务器关闭客户端的连接，发送一个 FIN 给客户端（报文段 6）。

（4） 客户段发回 ACK 报文确认，并将确认序号设置为收到序号加 1（报文段 7）。

CLOSED: 这个没什么好说的了，表示初始状态。

LISTEN: 这个也是非常容易理解的一个状态，表示服务器端的某个 SOCKET 处于监听状态，可以接受连接了。

SYN_RCVD: 这个状态表示接受到了 SYN 报文，在正常情况下，这个状态是服务器端的 SOCKET 在建立 TCP 连接时的三次握手会话过程中的一个中间状态，很短暂，基本上用 netstat 你是很难看到这种状态的，除非你特意写了一个客户端测试程序，故意将三次 TCP 握手过程中最后一个 ACK 报文不予发送。因此这种状态时，当收到客户端的 ACK 报文后，它会进入到 ESTABLISHED 状态。

SYN_SENT: 这个状态与 SYN_RCVD 遥想呼应，当客户端 SOCKET 执行 CONNECT 连接时，它首先发送 SYN 报文，因此也随即它会进入到了 SYN_SENT 状态，并等待服务端的发送三次握手中的第 2 个报文。SYN_SENT 状态表示客户端已发送 SYN 报文。

ESTABLISHED：这个容易理解了，表示连接已经建立了。

FIN_WAIT_1: 这个状态要好好解释一下，其实 FIN_WAIT_1 和 FIN_WAIT_2 状态的真正含义都是表示等待对方的 FIN 报文。而这两种状态的区别是：FIN_WAIT_1 状态实际上是当 SOCKET 在 ESTABLISHED 状态时，它想主动关闭连接，向对方发送了 FIN 报文，此时该 SOCKET 即进入到 FIN_WAIT_1 状态。而当对方回应 ACK 报文后，则进入到 FIN_WAIT_2 状态，当然在实际的正常情况下，无论对方何种情况下，都应该马上回应 ACK 报文，所以 FIN_WAIT_1 状态一般是比较难见到的，而 FIN_WAIT_2 状态还有时常常可以用 netstat 看到。

FIN_WAIT_2：上面已经详细解释了这种状态，实际上 FIN_WAIT_2 状态下的 SOCKET，表示半连接，也即有一方要求 close 连接，但另外还告诉对方，我暂时还有点数据需要传送给你，稍后再关闭连接。

TIME_WAIT: 表示收到了对方的 FIN 报文，并发送出了 ACK 报文，就等 2MSL 后即可回到 CLOSED 可用状态了。如果 FIN_WAIT_1 状态下，收到了对方同时带 FIN 标志和 ACK 标志的报文时，可以直接进入到 TIME_WAIT 状态，而无须经过 FIN_WAIT_2 状态。

CLOSING: 这种状态比较特殊，实际情况中应该是很少见，属于一种比较罕见的例外状态。正常情况下，当你发送 FIN 报文后，按理来说是应该先收到（或同时收到）对方的 ACK 报文，再收到对方的 FIN 报文。但是 CLOSING 状态表示你发送 FIN 报文后，并没有收到对方的 ACK 报文，反而却也收到了对方的 FIN 报文。什么情况下会出现此种情况呢？其实细想一下，也不难得出结论：那就是如果双方几乎在同时 close 一个 SOCKET 的话，那么就出现了双方同时发送 FIN 报文的情况，也即会出现 CLOSING 状态，表示双方都正在关闭 SOCKET 连接。

CLOSE_WAIT: 这种状态的含义其实是表示在等待关闭。怎么理解呢？当对方 close 一个 SOCKET 后发送 FIN 报文给自己，你系统毫无疑问地会回应一个 ACK 报文给对方，此时则进入到 CLOSE_WAIT 状态。接下来呢，实际上你真正需要考虑的事情是察看你是否还有数据发送给对方，如果没有的话，那么你也就可以 close 这个 SOCKET，发送 FIN 报文给对方，也即关闭连接。所以你在 CLOSE_WAIT 状态下，需要完成的事情是等待你去关闭连接。

LAST_ACK: 这个状态还是比较容易好理解的，它是被动关闭一方在发送 FIN 报文后，最后等待对方的 ACK 报文。当收到 ACK 报文后，也即可以进入到 CLOSED 可用状态了。

最后有 3 个问题的回答，我自己分析后的结论（不一定保证 100%正确）

1、为什么建立连接协议是三次握手，而关闭连接却是四次握手呢？

这是因为，服务端的 LISTEN 状态下的 SOCKET 当收到 SYN 报文的建连请求后，它可以把 ACK 和 SYN（ACK 起应答作用，而 SYN 起同步作用）放在一个报文里来发送。但关闭连接时，当收到对方的 FIN 报文通知时，它仅仅表示对方没有数据发送给你了；但未必你所有的数据都全部发送给对方了，所以你可以未必会马上会关闭 SOCKET,也即你可能还需要发送一些数据给对方之后，再发送 FIN 报文给对方来表示你同意现在可以关闭连接了，所以它这里的 ACK 报文和 FIN 报文多数情况下都是分开发送的。

2、为什么 TIME_WAIT 状态还需要等 2MSL 后才能返回到 CLOSED 状态？

这是因为，虽然双方都同意关闭连接了，而且握手的 4 个报文也都协调和发送完毕，按理可以直接回到 CLOSED 状态（就好比从 SYN_SEND 状态到 ESTABLISH 状态那样）；但是因为我们必须要假想网络是不可靠的，你无法保证你最后发送的 ACK 报文会一定被对方收到，因此对方处于 LAST_ACK 状态下的 SOCKET 可能会因为超时未收到 ACK 报文，而重发 FIN 报文，所以这个 TIME_WAIT 状态的作用就是用来重发可能丢失的 ACK 报文，并保证于此。

3、关闭 TCP 连接一定需要 4 次挥手吗？

不一定，4 次挥手关闭 TCP 连接是最安全的做法。但在有些时候，我们不喜欢 TIME_WAIT 状态（如当 MSL 数值设置过大导致服务器端有太多 TIME_WAIT 状态的 TCP 连接，减少这些条目数可以更快地关闭连接，为新连接释放更多资源），这时我们可以通过设置 SOCKET

变量的 SO_LINGER 标志来避免 SOCKET 在 close()之后进入 TIME_WAIT 状态，这时将通过发送 RST 强制终止 TCP 连接（取代正常的 TCP 四次握手的终止方式）。但这并不是一个很好的主意，TIME_WAIT 对于我们来说往往是有利的。