**TCP正常建立和关闭的状态变化**

TCP连接的建立可以简单的称为三次握手，而连接的中止则可以叫做 四次握手。

**建立连接**

在TCP/IP协议中，TCP协议提供可靠的连接服务，采用三次握手建立一个连接。

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\_SEND状态，等待服务器确认；

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状态，完成三次握手。

完成三次握手，客户端与服务器开始传送数据，也就是ESTABLISHED状态。

**结束连接**

    TCP有一个特别的概念叫做half-close，这个概念是说，TCP的连接是全双工（可以同时发送和接收）连接，因此在关闭连接的时候，必须关闭传和送两个方向上的连接。客户机给服务器一个FIN为1 的TCP报文，然后服务器返回给客户端一个确认ACK报文，并且发送一个FIN报文，当客户机回复ACK报文后（四次握手），连接就结束了。

**TCP状态变迁图**

这是一个看起来比较复杂的状态迁移图，因为它包含了两个部分---服务器的状态迁移和客户端的状态迁移，如果从某一个 角度出发来看这个图，就会清晰许多，这里面的服务器和客户端都不是绝对的，发送数据的就是客户端，接受数据的就是服务器。

客户端的状态可以用如下的流程来表示：

CLOSED->SYN\_SENT->ESTABLISHED->FIN\_WAIT\_1->FIN\_WAIT\_2->TIME\_WAIT->CLOSED

以上流程是在程序正常的情况下应该有的流程，从书中的图中可以看到，在建立连接时，当客户端收到SYN报文的ACK以后，客户端就打开了数据交互地 连接。而结束连接则通常是客户端主动结束的，客户端结束应用程序以后，需要经历FIN\_WAIT\_1，FIN\_WAIT\_2等状态，这些状态的迁移就是前 面提到的结束连接的四次握手。

服务器的状态可以用如下的流程来表示：

CLOSED->LISTEN->SYN收到 ->ESTABLISHED->CLOSE\_WAIT->LAST\_ACK->CLOSED

在建立连接的时候，服务器端是在第三次握手之后才进入数据交互状态，而关闭连接则是在关闭连接的第二次握手以后（注意不是第四次）。而关闭以后还要 等待客户端给出最后的ACK包才能进入初始的状态。

其他状态迁移

书中的图还有一些其他的状态迁移，这些状态迁移针对服务器和客户端两方面的总结如下

1. LISTEN->SYN\_SENT，对于这个解释就很简单了，服务器有时候也要打开连接的嘛。  
2. SYN\_SENT->SYN收到，服务器和客户端在SYN\_SENT状态下如果收到SYN数据报，则都需要发送SYN的ACK数据报并把自己的状态 调整到SYN收到状态，准备进入ESTABLISHED  
3. SYN\_SENT->CLOSED，在发送超时的情况下，会返回到CLOSED状态。  
4. SYN\_收到->LISTEN，如果受到RST包，会返回到LISTEN状态。  
5. SYN\_收到->FIN\_WAIT\_1，这个迁移是说，可以不用到ESTABLISHED状态，而可以直接跳转到FIN\_WAIT\_1状态并等待关 闭。

2MSL等待状态

书中给的图里面，有一个TIME\_WAIT等待状态，这个状态又叫做2MSL状态，说的是在TIME\_WAIT2发送了最后一个ACK数据报以后， 要进入 TIME\_WAIT状态，这个状态是防止最后一次握手的数据报没有传送到对方那里而准备的（注意这不是四次握手，这是第四次握手的保险状态）。这个状态在 很大程度上保证了双方都可以正常结束，但是，问题也来了。

由于插口的2MSL状态（插口是IP和端口对的意思，socket），使得应用程序在2MSL时间内是无法再次使用同一个插口的，对于客户程序还好 一些，但是对于服务程序，例如httpd，它总是要使用同一个端口来进行服务，而在 2MSL时间内，启动httpd就会出现错误（插口被使用）。为了避免这个错误，服务器给出了一个平静时间的概念，这是说在2MSL时间内，虽然可以重新 启动服务器，但是这个服务器还是要平静的等待2MSL时间的过去才能进行下一次连接。

FIN\_WAIT\_2状态

这就是著名的半关闭的状态了，这是在关闭连接时，客户端和服务器两次握手之后的状态。在这个状态下，应用程序还有接受数据的能力，但是已经无法发送 数据，但是也有一种可能是，客户端一直处于FIN\_WAIT\_2状态，而服务器则一直处于WAIT\_CLOSE状态，而直到应用层来决定关闭这个状态。

RST，同时打开和同时关闭

RST是另一种关闭连接的方式，应用程序应该可以判断RST包的真实性，即是否为异常中止。而同时打开和同时关闭则是两种特殊的TCP状态，发生的 概率很小。