【建立 TCP 连接】（三次握手）

由于 TCP 协议提供可靠的连接服务，于是采用有保障的三次握手方式来创建一个 TCP 连接。三次握手的具体过程如下：

1. 客户端发送一个带 SYN 标志的 TCP 报文（报文 1）到服务器端，表示希望建立一个 TCP 连接。
2. 服务器发送一个带 ACK 标志和 SYN 标志的 TCP 报文（报文 2）给客户端，ACK 用于对报文 1 的回应，SYN 用于询问客户端是否准备好进行数据传输。
3. 客户端发送一个带 ACK 标志的 TCP 报文（报文 3），作为报文 2 的回应。

至此，一个 TCP 连接就建立起来了。（详见下图）

【终止 TCP 连接】（四次挥手）

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。原则是主动关闭的一方（如已传输完所有数据等原因）发送一个 FIN 报文来表示终止这个方向的连接，收到一个 FIN 意味着这个方向不再有数据流动，但另一个方向仍能继续发送数据，直到另一个方向也发送 FIN 报文。四次挥手的具体过程如下：

1. 客户端发送一个 FIN 报文（报文 4）给服务器，表示我将关闭客户端到服务器端这个方向的连接。
2. 服务器收到报文 4 后，发送一个 ACK 报文（报文 5）给客户端，序号为报文 4 的序号加 1。
3. 服务器发送一个 FIN 报文（报文 6）给客户端，表示自己也将关闭服务器端到客户端这个方向的连接。
4. 客户端收到报文 6 后，发回一个 ACK 报文（报文 7）给服务器，序号为报文 6 的序号加 1。

至此，一个 TCP 连接就关闭了。（4 次挥手不是关闭 TCP 连接的唯一办法，见下文 Q3 疑问）

【TCP 连接状态】

下面是每一个 TCP 连接在任意时刻可能处于的状态，在 Linux 下可以在 netstat 命令的最后一列（State 列）里看到。
各个状态的含义如下：

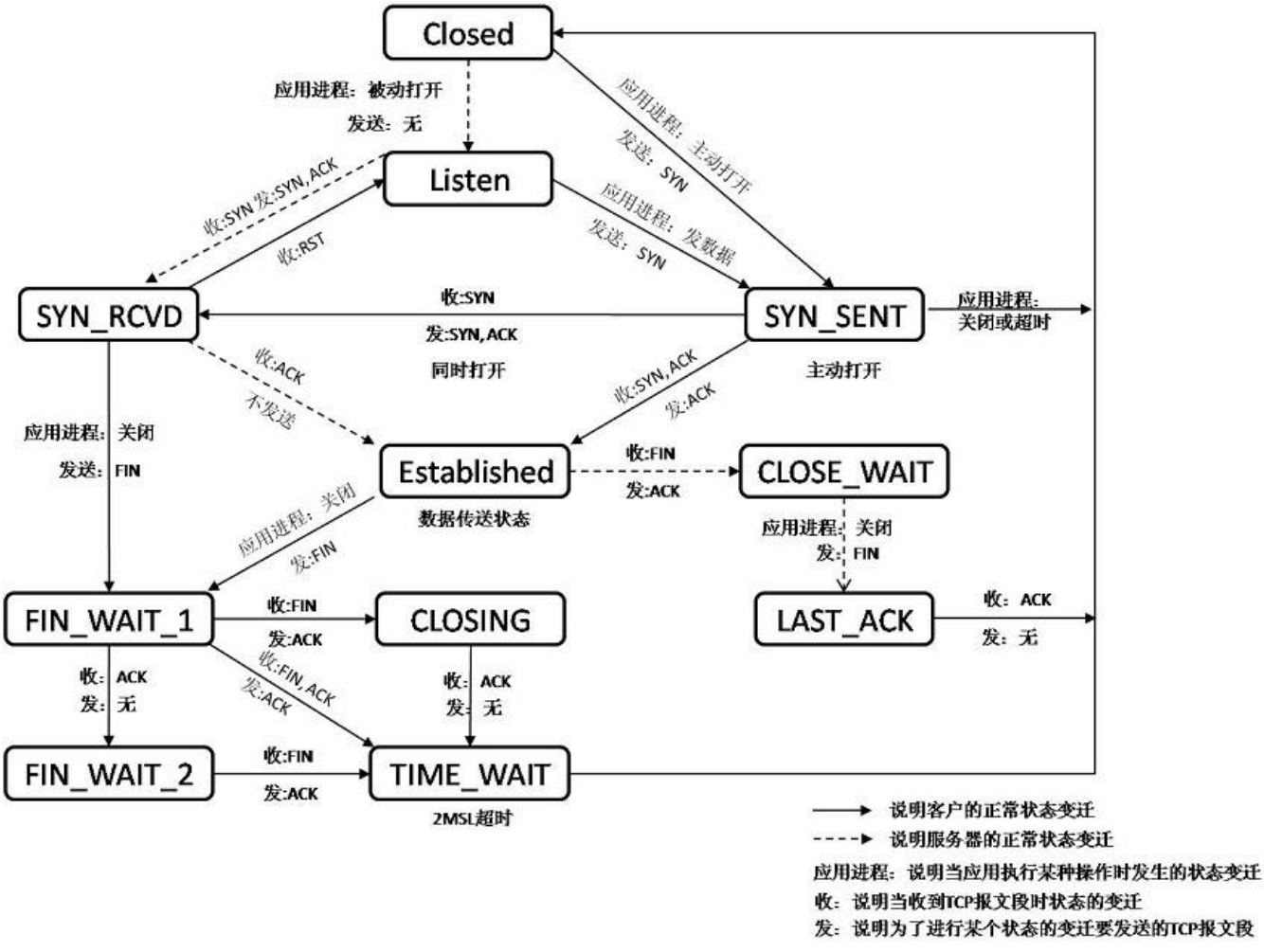
- CLOSED：初始状态，表示 TCP 连接是“关闭着的”或“未打开的”。
- LISTEN ：表示服务器端的某个 SOCKET 处于监听状态，可以接受客户端的连接。
- SYN_RCVD ：表示接收到了 SYN 报文。在正常情况下，这个状态是服务器端的 SOCKET 在建立 TCP 连接时的三次握手会话过程中的一个中间状态，很短暂，基本上用 netstat 很难看到这种状态，除非故意写一个监测程序，将三次 TCP 握手过程中最后一个 ACK 报文不予发送。当 TCP 连接处于此状态时，再收到客户端的 ACK 报文，它就会进入到 ESTABLISHED 状态。
- SYN_SENT ：这个状态与 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。
- ESTABLISHED ：表示 TCP 连接已经成功建立。
- 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 crash。
- TIME_WAIT ：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。
- 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，这是就会出现 CLOSING 状态，表示双方都正在关闭 SOCKET 连接。

- 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。
- 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。

【TCP 状态变迁图】

下面是收集自网上的几张图片，展示 TCP 连接的各种状态的变迁可能：



TCP 三次握手，四次挥手的时序图：