假设客户应用程序决定要关闭该连接。（注意到服务器也能选择关闭该连接。）这引起客户 TCP 发送一个带有 FIN 比特被置为 1 的 TCP 报文段，并进入 FIN_WAIT_1 状态。当处在 FIN_WAIT_1 状态时，客户 TCP 等待一个来自服务器的带有确认的 TCP 报文段。当它收到该报文段时，客户 TCP 进入 FIN_WAIT_2 状态。当处在 FIN_WAIT_2 状态时，客户等待来自服务器的 FIN 比特被置为 1 的另一个报文段；当收到该报文段后，客户 TCP 对服务器的报文段进行确认，并进入 TIME_WAIT 状态。假定 ACK 丢失，TIME_WAIT 状态使 TCP 客户重传最后的确认报文。在 TIME_WAIT 状态中所消耗的时间是与具体实现有关的，而典型的值是 30 秒、1 分钟或 2 分钟。经过等待后，连接就正式关闭，客户端所有资源（包括端口号）将被释放。

关注安全性

SYN 洪泛攻击

我们在 TCP 三次握手的讨论中已经看到，服务器为了响应一个收到的 SYN，分配并初始化连接变量和缓存。然后服务器发送一个 SYNACK 进行响应，并等待来自客户的 ACK 报文段。如果某客户不发送 ACK 来完成该三次握手的第三步，最终（通常在一分多钟之后）服务器将终止该半开连接并回收资源。

这种 TCP 连接管理协议为经典的 DoS 攻击即 **SYN 洪泛攻击**（SYN flood attack）提供了环境。在这种攻击中，攻击者发送大量的 TCP SYN 报文段，而不完成第三次握手的步骤。随着这种 SYN 报文段纷至沓来，服务器不断为这些半开连接分配资源（但从未使用），导致服务器的连接资源被消耗殆尽。这种 SYN 洪泛攻击是被记载的众多 DoS 攻击中的第一种［CERT SYN 1996］。幸运的是，现在有一种有效的防御系统，称为 **SYN cookie**［RFC 4987］，它们被部署在大多数主流操作系统中。SYN cookie 以下列方式工作：

- 当服务器接收到一个 SYN 报文段时，它并不知道该报文段是来自一个合法的用户，还是一个 SYN 洪泛攻击的一部分。因此服务器不会为该报文段生成一个半开连接。相反，服务器生成一个初始 TCP 序列号，该序列号是 SYN 报文段的源和目的 IP 地址与端口号以及仅有该服务器知道的秘密数的一个复杂函数（散列函数）。这种精心制作的初始序列号被称为“cookie”。服务器则发送具有这种特殊初始序列号的 SYNACK 分组。**重要的是，服务器并不记忆该 cookie 或任何对应于 SYN 的其他状态信息。**
- 如果客户是合法的，则它将返回一个 ACK 报文段。当服务器收到该 ACK，需要验证该 ACK 是与前面发送的某些 SYN 相对应的。如果服务器没有维护有关 SYN 报文段的记忆，这是怎样完成的呢？正如你可能猜测的那样，它是借助于 cookie 来做到的。前面讲过对于一个合法的 ACK，在确认字段中的值等于在 SYNACK 字段（此时为 cookie 值）中的值加 1（参见图 3-39）。服务器则将使用在 SYNACK 报文段中的源和目的地 IP 地址与端口号（它们与初始的 SYN 中的相同）以及秘密数运行相同的散列函数。如果该函数的结果加 1 与在客户的 SYNACK 中的确认（cookie）值相同的话，服务器认为该 ACK 对应于较早的 SYN 报文段，因此它是合法的。服务器则生成一个具有套接字的全开的连接。