第2章 T/TCP协议

2.1 概述

我们分两章(第2章和第4章)讨论T/TCP协议。这样，在深入研究T/TCP协议之前(第3章)，我们可以先看一些应用T/TCP的例子。本章主要对协议应用技巧和实现中用到的变量做一个介绍。下一章我们学习一些T/TCP应用的示例程序。第4章结束我们对T/TCP协议的学习。

在第1章中我们已经看到了，当把TCP协议应用于客户-服务器事务时会存在两个问题：

1) 如图1-8所示，三次握手使客户端测得的事务时间额外多出一个RTT。

2) 由于客户进程主动关闭连接(即由客户进程首先发出FIN)，因而在客户收到服务器的FIN后还要在TIME_WAIT状态滞留大约240秒。

   TIME_WAIT状态和16比特TCP端口号这两者结合起来限制了两台主机之间的最大事务速率。例如，如果同一台客户主机要不断地和同一台服务器主机进行事务通信，那么它要么每完成一次事务后等待240秒才开始下一个事务，要么为紧接着的事务选择另外一个端口号。但每240秒的时间内至多只能有64 512个端口(65 535减去1 023个知名端口)可用，从而每秒最多也就只能处理268个事务。在RTT值大约为1~3ms的局域网上，实际上可能会超过这个速率。

   而且，即使应用程序的事务速率低于每秒240次，比如每240秒只有50 000次。当客户端处于TIME_WAIT状态时，协议还是需要控制块来保存连接的状态。卷2中给出的BSD实现中，每个连接都需要一个IP控制块(84字节)，一个TCP控制块(140字节)和一个TCP/IP首部模板(40字节)。这样总共就需要13 200 000字节的内核存储空间。这个开销即便在内存不断扩大的今天依然显得大了些。

现在，T/TCP协议解决了这两个问题，采用的方法是绕过三次握手，并把TIME_WAIT状态的保持时间由240秒缩短到大约12秒。我们将在第4章中详细研究这两个特点。

T/TCP协议的核心称为TAO，即TCP加速打开，跳过了TCP的三次握手。T/TCP给主机建立的每个连接分配一个唯一的标识符，称为连接计数(CC)。每台T/TCP主机都要将不同主机对之间的最新连接计数CC保持一段时间。当服务器收到来自T/TCP客户的SYN时，如果其中携带的CC大于该主机对最新连接的CC，就保证这是一个新的SYN，于是就接受该连接请求，而不需要三次握手。这个过程称为TAO测试。如果测试失败，TCP还是用三次握手的老方法来确认当前这个SYN是否为新的。

2.2 T/TCP中的新TCP选项

T/TCP协议中有三个新的TCP选项。图2-1给出了目前TCP协议使用的所有选项。其中前3个出自最初的TCP协议规范，即RFC 793 [Postel 1981b]。而窗口宽度和时间戳则是在RFC 1323 [Jacobson, Braden, and Borman 1992]中定义的。最后三个选项(CC、CCnew和CCecho)则是T/TCP协议新引入的，在RFC 1644 [Braden 1994]中定义。最后这几个选项的使用规则如下：

1) CC选项在客户执行主动打开操作时发出的第一个 SYN报文段中使用。它也可以在其他一些报文段中使用，但前提是对方发过来的 SYN报文段中带有CC或CCnew选项。
2) CCnew选项只能在第一个 SYN报文段中使用。当需要执行正常的三次握手操作时，客户端的TCP协议就使用CCnew选项而不用CC选项。
3) CCecho选项仅在三次握手过程中的第二个报文段中使用：通常由服务器发出该报文段，并携带有SYN和ACK。该报文段将CC或CCnew的值返回给客户，告知客户本服务器支持T/TCP协议。

本章以及下一章的例子中我们还会进一步讨论这些选项。

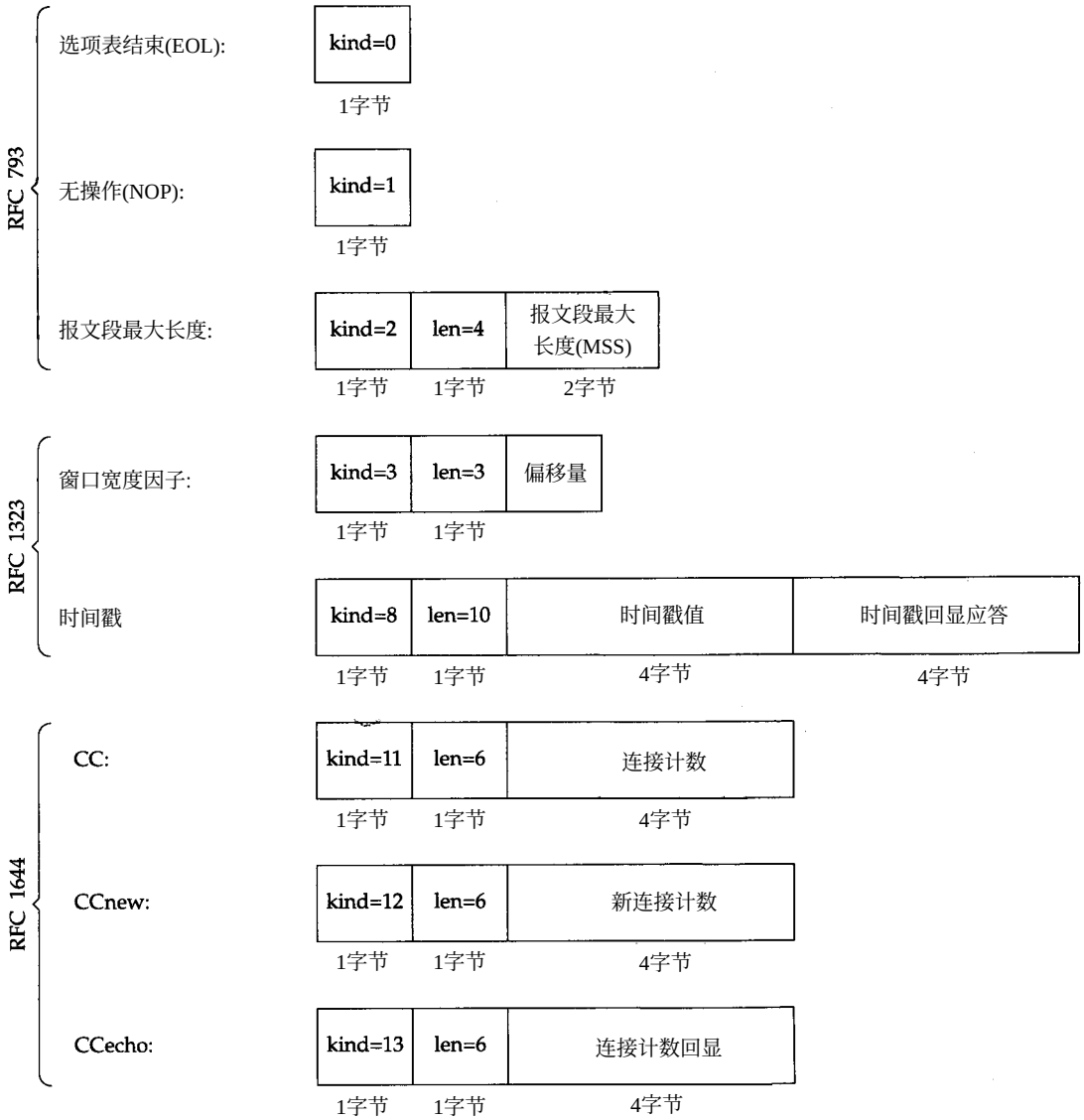

图2-1 TCP选项

不难发现，T/TCP的3个新选项均为6字节长。为了使这些选项继续按4字节定界(这在某些系统体系结构中有助于提高性能)，我们通常在这些选项的前面加上两个单字节的无操作(NOP)。

如果客户既支持RFC 1323，也支持T/TCP协议，这时客户发给服务器的第一个SYN报文段中的TCP选项，如图2-2所示。我们特意给出了每个选项的类型值和长度值；NOP用阴影表示，其类型值为1。第二个选项是窗口宽度，这里用“ WS ”标记。方格上方的数字是每个选项相对于选项字段起始的字节偏移量。TCP协议选项的最大长度为40字节，本例中的TCP选项共需28字节。从图中可以看出，采用NOP填充以后，所有4个4字节的值都符合4字节定界规则。

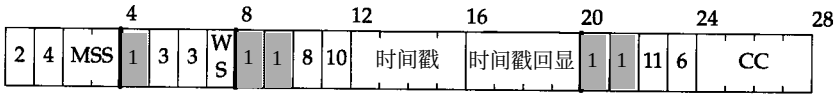

图2-2 同时支持RFC 1323和T/TCP的客户发给服务器的第一个SYN报文段的TCP选项

如果服务器既不支持RFC 1323，也不支持T/TCP协议，它发给客户带有SYN和ACK的应答中就只有报文段最大长度(MSS)选项。但如果服务器既支持RFC 1323，也支持T/TCP协议，那么它给客户的应答中将包含图2-3所示的TCP选项，总长为36字节。

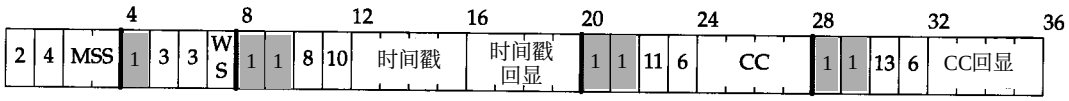

图2-3 服务器对图2-2所示请求的应答中的TCP选项

由于CCecho选项总是和CC选项一起发送，因此T/TCP协议的设计本可以把这两个选项合二为一，从而为宝贵的TCP协议选项空间节省4个字节。或者也可以这样，这种最坏的选项排列只在服务器给出SYN/ACK时出现，而它们的出现无论如何总要使TCP处理速度变慢的，因此索性连NOP字节也省去，实际上可以节省7个字节。

因为报文段的最大长度和窗口宽度选项只在SYN报文段中出现，而CCecho选项只在SYN/ACK报文段中出现，因此，如果连接两端都支持RFC 1323和T/TCP协议，则自此以后的报文段中也都只包含时间戳和CC选项，如图2-4所示。

图2-4 两端都支持RFC 1323和T/TCP时非SYN报文段所包含的TCP选项

可以看出，一旦连接建立，时间戳和连接计数CC选项给所有的TCP报文段都增加了20字节。当讲到T/TCP协议时，我们常常用一般术语CC选项作为本节所引入的3个TCP选项的统称。

时间戳和CC选项带来了多大的额外开销呢？假设两台主机位于两个不同的网络上，报文段最大长度MSS设为典型值512字节。要传递一兆字节的文件，如果没有这些选项，则需要1 954个报文段；如果使用时间戳和CC选项，则需要2 033个报文段，较前者增加了4% 。如果报文段最大长度MSS为1 460字节，那么报文段数只增加了1.5%。

2.3 T/TCP 实现所需变量

T/TCP协议要求内核保存一些新增的信息，本节将对这些信息加以描述，后面几节将讨论

如何使用这些新信息。

(1) tcp_ccgen：这是一个32位的全局整型变量，记录待用的CC值。每当主机建立了一个连接，该变量的值就加1，无论是主动还是被动，也不论是否使用T/TCP协议。该变量永不为0。当变量渐渐增长时，如果又回到了0，那么就将其值置为1。

(2) 每主机高速缓存(per-host cache)，其中包含了三个新变量：tao_cc、tao_ccsent和tao_mssopt。该高速缓存也称为TAO高速缓存。我们将看到，T/TCP协议为每一个与之通信的主机创建一个路由表项，并把这些信息存储在路由表项中(把每主机高速缓存安排在路由表中是很方便的。当然也可以另开一张完全分离的表作为这个每主机高速缓存。T/TCP协议不需要对IP路由功能做任何改动)。在每主机高速缓存中创建一个新表项时，tao_cc和tao_ccsent必须初始化为0，表示它们尚未定义。

tao_cc记录的是最后一次从对应主机接收到且不含ACK的合法的SYN报文段(即主动打开连接)中的CC值。当T/TCP主机收到一个带有CC选项的SYN报文段时，如果CC选项的值大于tao_cc，那么主机就知道这是一个新的SYN报文段，而不是一个重复的老SYN，这样就可以跳过三次握手(TAO测试)。

tao_ccsent记录的是发给相应主机的最后一个不含ACK的SYN报文段(即主动打开连接)中的CC值。如果该值未定义(为0)，那么只有当对方发回一个CCecho选项，表示其可以使用T/TCP协议时，才将tao_ccsent设置为非0。

tao_mssopt是最后一次从相应主机接收到的报文段最大长度选项值。

(3) 现有的TCP控制块中需要增加三个新的变量：cc_send、cc_recv和t_duration。第1个变量记录的是该连接上发送的每一个报文段中的CC值，第2个变量记录的是希望对方发来的报文段中所携带的CC值，最后一个变量则用来记录连接已经建立了多长时间(以系统的时钟滴答计算)。当连接主动关闭时，如果该时间计数器显示的连接持

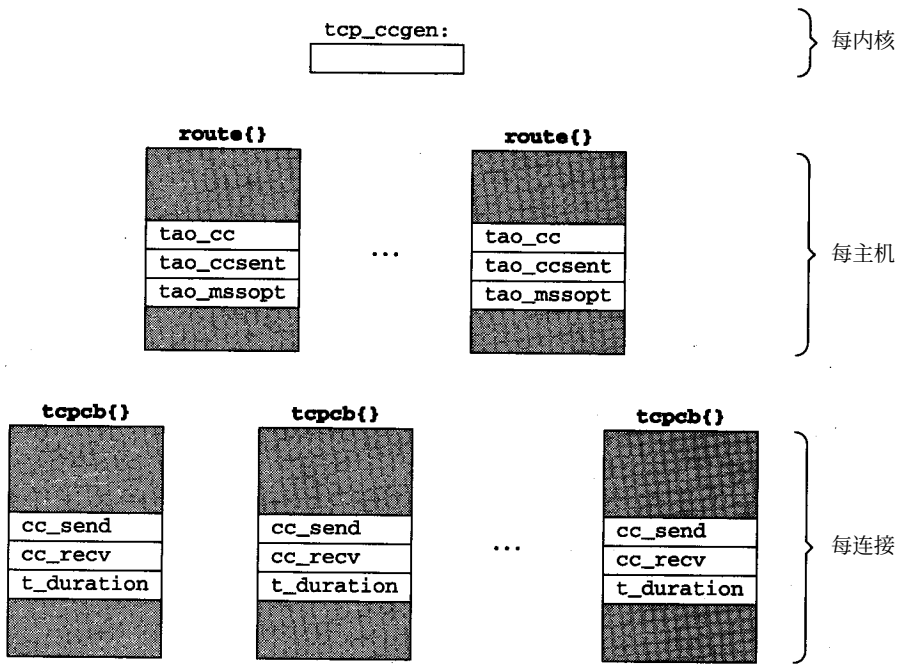

图2-5 T/TCP实现中的变量

续时间小于报文段最大生存时间 MSL，则TIME_WAIT状态将被截断。我们在 4.4节中将更详细地讨论这个问题。

我们在图2-5中给出这些新变量。在后续章节讲 T/TCP协议实现时就用这些变量。

在这个图中，我们用{}表示结构。图中的TCP控制块是一个tcpcb结构。所有TCP协议的实现都必须为其中的连接保存并维护一个控制块，控制块的形式可以这样那样，但必须包含特定连接的所有变量。

2.4 状态变迁图

TCP协议的工作过程可以用图 2-6所示的状态变迁图来描述。大多数状态变迁图都把状态变迁时发送的报文段标在变迁线的边上。例如，从 CLOSED状态到SYN_SENT状态的变迁就标明发送了一个SYN报文段。在图2-6中则没有采用这种标记方法，而是在每个状态框中标出处于该状态时要发送的报文段类型。例如，当处于 SYN_RECV状态时，要发出一个带有 SYN的报文段，其中还包括对所收到 SYN的确认(ACK)。而当处于CLOSE_WAIT状态时，要发出对所收到FIN的确认(ACK)。

我们之所以要这样做是因为，在 T/TCP协议中我们经常需要处理可能造成多次状态变迁的报文段。于是在处理一个报文段时，重要的是处理完报文段后连接所处的最终状态，因为它决定了应答的内容。而如果不使用 T/TCP协议，每收到一个报文段通常至多只引起一次状态变迁，只有在收到SYN/ACK报文段时才是例外，很快我们就要讨论这个问题。

与RFC 793 [Postel 1981b]中的TCP协议状态变迁图相比，图2-6还有另外一些不同之处。

- RFC 793的状态变迁图中，当应用程序发送数据时，会有从 LISTEN状态到SYN_SENT状态的变迁。但实际上典型的 API很少提供这种功能。
- RFC 1122 [Braden 1989]中描绘了一个直接从FIN_WAIT_1状态到TIME_WAIT状态的变迁，这发生在收到了一个带有 FIN和对所发FIN的确认(ACK)的报文段时。但是当收到这样一个报文段时，通常都是先处理 ACK使状态变迁到 FIN_WAIT_2，接着再处理 FIN，并变迁到TIME_WAIT状态。因此，图2-6也能正确处理这样的报文段。这就是收到一个报文段导致两次状态变迁的例子。
- 除了SYN_SENT之外的所有状态都发送 ACK(处于LISTEN这个末梢状态时，则什么也不发送)。这是因为发送 ACK是不受条件限制的：标准 TCP报文段的首部总是留有 ACK的位置。因此， TCP总是确认已接收到的报文段最高序列号 (加1)，只有在处理主动打开(SYN_SENT)的SYN报文段和一些重建(RST)报文段时才是例外。

TCP输入的处理顺序

TCP协议收到报文段时，对其中所携带的各种控制信息 (SYN、FIN、ACK、URG和RST标志，还可能有数据和选项)的处理顺序不是随意的，也不是各种实现可以自行决定的。 RFC 793中对处理顺序有明确的规定。图 11-1给这些步骤做了个小结，该小结同时也用黑体标明了T/TCP中所做的改动。

例如，当T/TCP客户收到一个携带有SYN、数据、 FIN和ACK的报文段时，协议首先处理的是SYN(因为此时的插口还处于 SYN_SENT状态)，接着是处理 ACK标志，再接着是数据，最后才是FIN。三个标志中的任何一个都有可能引起相应插口的连接状态改变。

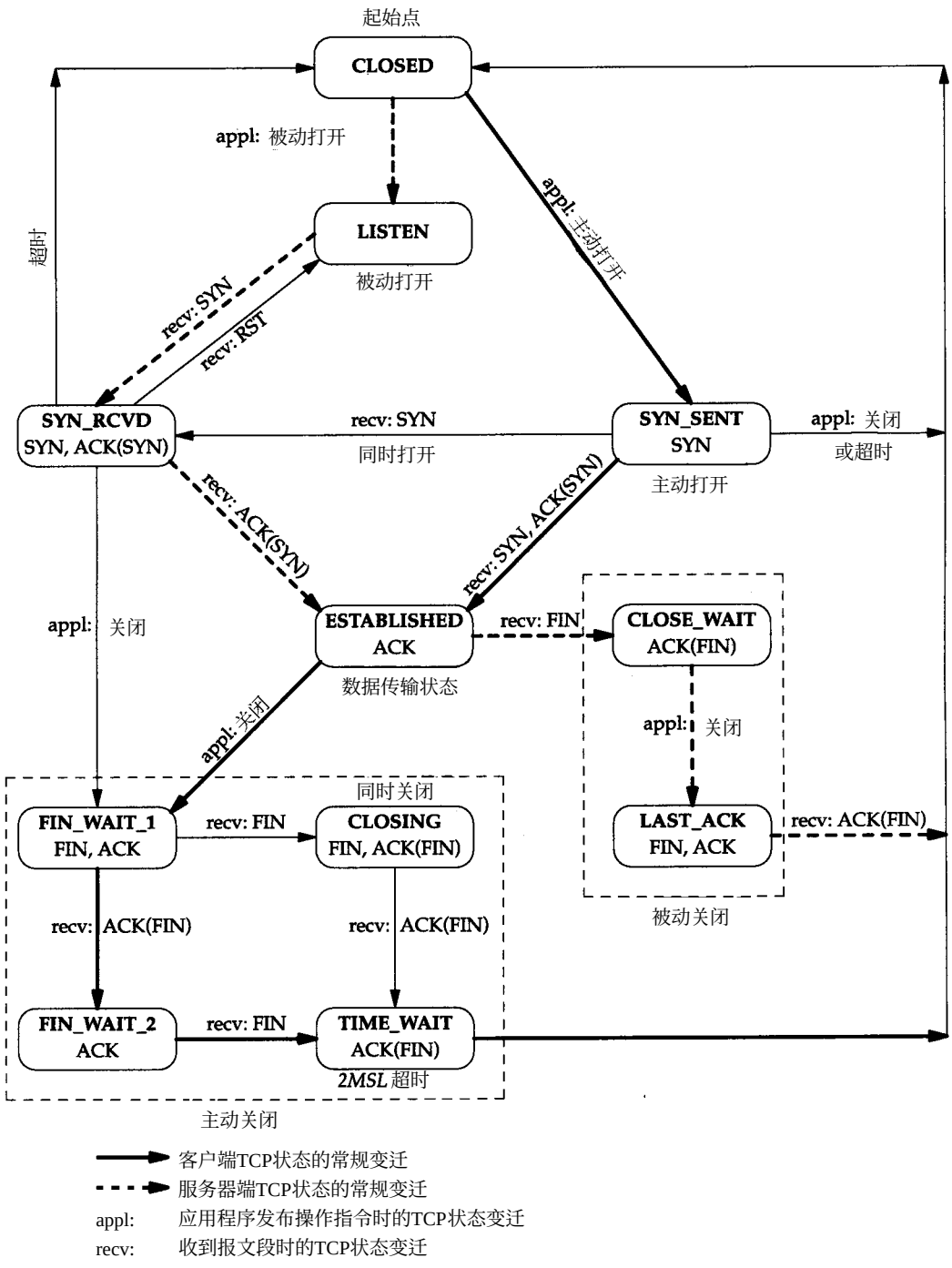

图2-6 TCP的状态变迁图

2.5 T/TCP的扩展状态

T/TCP中定义了7个扩展状态，这些扩展状态都称为加星状态。它们分别是：SYN_SENT*、SYN_RCVD*、ESTABLISHED*、CLOSE_WAIT*、LAST_ACK*、FIN_WAIT_1*和

CLOSING*。例如，在图1-12中，客户发出的第一个报文段中包含有 SYN标志、数据和FIN。当该报文段是在主动打开中发送出去时，客户随即进入 SYN_SENT*状态，而不是进入通常的SYN_SENT状态，这是因为随报文段还必须发出一个 FIN。当收到服务器的应答时，该应答中包含有服务器的SYN、数据和FIN，以及对客户的SYN、数据和FIN的确认(ACK)。这时客户端插口的连接状态要经历一系列的状态变迁：

- 对客户SYN的ACK将连接的状态变迁到FIN_WAIT_1。传统的ESTABLISHED状态就这样完全跳过去了，因为这时客户已经发出了 FIN。
- 对客户FIN的ACK将连接状态变迁到了FIN_WAIT_2。
- 收到服务器的FIN，连接状态变迁到TIME_WAIT。

RFC 1379详细描述了包括所有这些加星状态后的状态变迁图演变过程。当然，得到的结果远比图2-6复杂，其中有很多重叠的线。幸运的是，无星状态和对应的加星状态之间只是一些简单的关系。

- SYN_SENT*状态和SYN_RCVD*状态与对应的无星状态几乎完全相同，唯一的不同之处是在加星状态下要发出一个 FIN。这就是说，当一端主动打开连接、并且应用程序在连接建立之前就指定了 MSG_EOF(发送FIN)时就进入相应的加星状态。在这种情况下，客户端一般是进入SYN_SENT*状态，SYN_RCVD*状态只有当双方碰巧同时执行打开连接操作的偶然情况下才会出现，关于这一点我们在卷 1的18.8节中已有详细讨论。
- ESTABLISHED*、CLOSE_WAIT*、LAST_ACK*、FIN_WAIT_1*和CLOSING*这五个状态与对应的不加星状态除了要发送SYN外也完全相同。当连接处于这五个状态之一时，叫做已经半同步了。当接收端处于被动状态且收到一个带有 TAO测试、可选数据和可选FIN的SYN报文段时，连接即进入这些加星状态 (4.5节详细描述了TAO测试)。之所以用半同步这个词是因为，一旦收到 SYN接收端就认为连接已经建立了 (因为已经通过了TAO测试)，尽管此时刚刚完成了常规三次握手过程的一半。

图2-7给出了加星状态和对应的常规状态。对于每个可能的状态，表中还列出了所发出的报文段类型。

我们将会看到，从实现的角度来看，这些加星的状态是很容易处理的。除了要保持当前已有的无星状态外，在每个连接的TCP控制块中还有两个额外的标志：

- TF_SENDFIN表示需要发送FIN(对应于SYN_SENT*状态和SYN_RCVD*状态)；
- TF_SENDSYN表示需要发送SYN(对应于图2-7中的5个半同步加星状态)。

常规状态	说　明	发　送	加星状态	发　送
CLOSED	关闭	RST, ACK		
LISTEN	监听连接请求(被动打开)			
SYN_SENT	已发出SYN(主动打开)	SYN	SYN_SENT*	SYN, FIN
SYN_RCVD	已经发出和收到SYN；等待ACK	SYN, ACK	SYN_RCVD*	SYN, FIN, ACK
ESTABLISHED	连接已经建立(数据传输)	ACK	ESTABLISHED*	SYN, ACK
CLOSE_WAIT	收到FIN，等待应用程序关闭	ACK	CLOSE_WAIT*	SYN, ACK
FIN_WAIT_1	已经关闭，发出FIN；等待ACK和FIN	FIN, ACK	FIN_WAIT_1*	SYN, FIN, ACK
CLOSING	两端同时关闭；等待ACK	FIN, ACK	CLOSING*	SYN, FIN, ACK
LAST_ACK	收到FIN已经关闭；等待ACK	FIN, ACK	LAST_ACK*	SYN, FIN, ACK
FIN_WAIT_2	已经关闭；等待FIN	ACK		
TIME_WAIT	主动关闭后长达2MSL的等待状态	ACK		

图2-7　TCP根据不同的当前状态(常规或加星)所发送的内容

在图2-7中，加星状态下把SYN和FIN这两个新标志置于开状态时用黑体标出。

2.6 小结

T/TCP的核心是TAO，即TCP加速打开。这项技术使得 T/TCP服务器收到T/TCP客户的SYN报文段后能够知道这个 SYN是新的，从而可以跳过三次握手。确保服务器所收 SYN是新SYN的技术(TAO测试)是为主机已经建立的每个连接分配一个唯一的标识符：连接计数 CC。每个T/TCP主机都要把与每一个对等主机之间最新连接的 CC值保留一段时间。如果所收 SYN报文段的CC值大于从对等主机接收的最新CC值，那么TAO测试成功。

T/TCP定义了3个新的选项： CC、CCnew和CCecho。所有选项都包含一个长度域 (这和RFC 1323中规定的其他选项一样)，使不认识这些选项的TCP实现能跳过它们。如果某个连接使用了T/TCP协议，那么每个报文段都将包含连接计数选项 (不过有时在客户的SYN报文段中用CCnew代替CC)。

T/TCP加入了一个全局内核变量，还在每主机高速缓存中加入了 3个变量，并为正在使用的每个连接控制块增加了3个变量。本书中讨论的T/TCP实现利用业已存在的路由表作为每主机高速缓存。

TCP的状态变迁图有10个状态， T/TCP协议在此基础上还增加了7个额外的状态。但实际上协议实现是简单的：由于新的状态只是已有状态的扩充，因而只需要为每个连接引入两个新的标志，分别指示是否需要发送一个 SYN报文段以及是否需要发送一个FIN报文段，即可定义7种新的状态。