

## 第8章 T/TCP的实现：TCP概要

### 8.1 概述

本章内容覆盖了T/TCP对TCP数据结构和函数所做的全局性修改。增加了两个全局变量：`tcp_ccgen`，即全局CC计数器，以及`tcp_do_rfc1644`，这是一个标志变量，说明是否选用CC选项。TCP的协议交换记录项也作了修改，以支持隐式的打开和关闭。另外还在TCP控制块中增加了4个变量。

对`tcp_slowtimo`函数也作了简单修改，以便能够测量每个连接的持续时间。给定一个连接的持续时间，如果持续时间短于MSL，则如4.4节所述，T/TCP将截断TIME\_WAIT状态的保持时间。

### 8.2 代码介绍

T/TCP没有增加新的源文件，但是需要一些新的变量。

#### 全局变量

图8-1中给出了T/TCP新增加的全局变量，在各个TCP函数中都会用到。

变 量	数据类型	说 明
<code>tcp_ccgen</code>	<code>tcp_cc</code>	要发送的下一个CC值
<code>tcp_do_rfc1644</code>	<code>int</code>	如果为真(默认)，发送CC或CCnew选项

图8-1 T/TCP新增的全局变量

在第3章我们给出了一些有关`tcp_ccgen`变量的例子。在6.5节中也提到了`tcp_cc`的数据类型是用`typedef`定义的，是无符号长整数。`tcp_cc`变量值为0，表示它尚未定义。`tcp_ccgen`变量总是这样存取的：

```
tp->cc_send = CC_INC(tcp_ccgen);
```

其中`cc_send`是TCP控制块的新字段(见后面的图8-3)。宏`CC_INC`是在`<netinet/tcp_seq.h>`中定义的：

```
#define CC_INC(c)    ((++(c) == 0 ? ++(c) : (c))
```

由于这个值是在使用之前增加的，因此，`tcp_ccgen`要初始化为0，且它的第一个有用值为1。

为了按照模运算比较CC的值，定义了四个宏：`CC_LT`、`CC_LEQ`、`CC_GT`和`CC_GEQ`。这四个宏与卷2第649页定义四个`SEQ_xx`宏完全一样。

变量`tcp_do_rfc1644`与卷2中介绍的变量`tcp_do_rfc1323`相似。如果`tcp_do_rfc1644`为0，TCP不会向对方发送CC或CCnew选项。

## 统计量

T/TCP新增了5个计数器，如图8-2所示。它们加在 `tcpstat` 结构中，卷2第638页对这个结构有介绍。

tcpstat字段	说 明
<code>tcps_taook</code>	TAO正确时接收到SYN
<code>tcps_taofail</code>	接收到带有CC选项的SYN，但TAO测试失败
<code>tcps_badcchecho</code>	CCecho选项错误的SYN/ACK报文段
<code>tcps_impliedack</code>	隐含着对上一次连接的ACK的新SYN
<code>tcps_ccdrop</code>	因为无效的CC选项而丢弃的报文段

图8-2 在 `tcpstat` 结构中新增的T/TCP统计量

程序 `netstat` 必须经修改才能打印这些新字段的值。

### 8.3 TCP的 `protosw` 结构

我们在第5章提到过，TCP的 `protosw` 记录项 `inet sw[2]` (卷2第641页)的 `pr_flags` 字段在T/TCP中作了修改。新的插口层标志 `PR_IMPLOPCL` 必须包括在内，已有的标志 `PR_CONNREQUIRED` 和 `PR_WANTRCVD` 也必须包含在内。在 `sosend` 中，如果调用进程给出了一个目标地址，这个新的标志允许对一个未建连接的插口调用 `sendto`，并且如果指定了 `MSG_EOF` 标志，它所起的作用是发出一个 `PRU_SEND_EOF` 请求而不是 `PRU_SEND` 请求。

对 `protosw` 记录项所作的修改中有一个不是T/TCP所需的，即定义了 `tcp_sysctl` 函数作为 `pr_sysctl` 字段。这就允许系统管理员用前缀为 `net.inet.tcp` 的 `sysctl` 程序来修改能够控制TCP操作的一些变量值 (卷2介绍的Net/3代码仅仅支持 `sysctl` 程序通过 `ip_sysctl`、`icmp_sysctl` 和 `udp_sysctl` 函数对IP、ICMP和UDP的一些变量进行控制)。在图12-6中给出了 `tcp_sysctl` 函数。

### 8.4 TCP控制块

TCP控制块中新增了四个变量，卷2第643~644页说明了TCP控制块的 `tcpcb` 结构。我们在图8-3中仅给出了新的字段，并非整个结构。

变 量	数据类型	说 明
<code>t_duration</code>	<code>u_long</code>	以500ms为单位的连接持续时间
<code>t_maxopd</code>	<code>u_short</code>	MSS加上通常选项的长度
<code>cc_send</code>	<code>tcp_cc</code>	发送给对等端的CC值
<code>cc_recv</code>	<code>tcp_cc</code>	从对等端中接收到的CC值

图8-3 T/TCP在 `tcpcb` 结构中新增的字段

`t_duration` 用于确定T/TCP是否可以截断 `TIME_WAIT` 状态的保持时间，见4.4节的讨论。当控制块创建时它的值为0，由 `tcp_slowtimo` (8.6节) 每过500 ms加1。

`t_maxopd` 是为了代码的方便而设的。它的取值是已有 `t_maxseg` 字段的值加上TCP选项通常所占用的字节数。`t_maxseg` 是每个报文段中的数据字节数。例如，在MTU为1500字节的一个以太网上，如果时间戳和T/TCP都用上了，`t_maxopd` 将为1460，`t_maxseg` 则为1440。

它们之间的差值 20 字节是由 12 字节的时间戳选项加上 8 字节的 CC 选项(图 2-4)造成的。  
t\_maxopd和t\_maxseg都是在tcp\_mssrcvd函数中计算并记录的。

最后两个变量来自 RFC 1644, 在第 2 章给出了有关这三个变量的例子。如果一个连接的两端主机都用了 CC 选项, cc\_recv的值将为非 0。

在 TCP 控制块的 t\_flags 字段中新定义了 6 个标志, 如图 8-4 所示, 是对卷 2 的图 24-14 中的 9 个标志的补充。

t_flags	说 明
TF_SENDSYN	发送 SYN(隐藏的半同步连接状态标志)
TF_SENDFIN	发送 FIN(隐藏的状态标志)
TF_SENDCCNEW	主动打开时发送 CCnew 选项而不是 CC 选项
TF_NOPUSH	不发送报文段, 只清空发送缓存
TF_RCVD_CC	当对端在 SYN 中发送了 CC 选项时设置该标志
TF_REQ_CC	已经/将在 SYN 中申请 CC 选项

图 8-4 T/TCP 新增的 t\_flags 及其取值

不要把 T/TCP 中的两个标志 TF\_SENDFIN 与 TF\_SENTFIN 混淆, 前者表示 TCP 需要发送 FIN, 而后者表示已经发出 FIN。

TF\_SENDSYN 和 TF\_SENDFIN 这两个名字源于 Bob Braden 的“T/TCP 的实现”。FreeBSD 实现中将这两个名字改为 TF\_NEEDSYN 和 TF\_NEEDFIN。我们选用了前面的名字, 因为已经用新的标志来表示是否需要发送控制标志, 如果选用后面的名字就会误解为需要接收 SYN 或 FIN。然而请注意, 因为选用了这样的名字, T/TCP 的 TF\_SENDFIN 标志和已有的 TF\_SENTFIN 标志(表明 TCP 已经发出了 FIN)仅有一个字符之差。

我们将在下一章的图 9-3 和图 9-7 中分别介绍 TF\_NOPUSH 和 TF\_SENDCCNEW 标志。

## 8.5 tcp\_init 函数

所有的 T/TCP 变量都不需要显式的初始化, 因此卷 2 中介绍的 tcp\_init 函数没有变化。全局变量 tcp\_ccgen 是没有初始化的外部变量, 按照 C 语言的规则, 它的默认值为 0。这样做不会出错, 因为在 8.2 节中定义的宏 CC\_INC 是先对该变量加 1, 然后再用, 因此在重启后, tcp\_ccgen 的第一个有用值是 1。

T/TCP 也要求在重启时将 TAO 缓存全部清空, 由于在重启时要初始化 IP 路由表, 所以 TAO 缓存不需要专门处理。在路由表中每增加一个新的 rtentry 结构, rtrequest 要将该结构初始化为 0(卷 2 第 489 页)。这就意味着 rmxp\_tao 结构中 3 个 TAO 变量的默认值都为 0(图 6-3)。为新主机创建新的 TAO 记录项时, T/TCP 需要将 tao\_cc 的值初始化为 0。

## 8.6 tcp\_slowtimo 函数

两个 TCP 定时函数中有一个增加了一行: 每次处理 500 ms 定时器时, 要对每个 TCP 控制块的 t\_duration 字段执行加 1 操作, 卷 2 第 666 页给出了 tcp\_slowtimo 函数。下面这一行

```
tp->t_duration++;
```

加在这个图的第94~95行之间。这个变量的用途是测量每个连接的长度，以500ms为单位。如果连接持续时间短于MSL，TIME\_WAIT状态的保持时间就可以截断，在4.4节中已经讨论过。

与这项优化有关的工作是在<netinet/timer.h>头文件中定义了下面这个常量：

```
#define TCPTV_TWTRUNC 8 /* RTO factor to truncate TIME_WAIT */
```

我们在图11-17和图11-19中可以看到，如果T/TCP连接是主动关闭，并且t\_duration的值小于TCPTV\_MSL(60个500ms，即30秒)，那么TIME\_WAIT状态的保持时间就是当前重传超时(RTO)乘以TCPTV\_TWTRUNC。在局域网环境中，RTO通常为3个500ms，即1.5秒，这将使TIME\_WAIT状态的保持时间缩短到12秒。

## 8.7 小结

T/TCP新增了两个全局变量(tcp\_ccgen和tcp\_do\_rfc1644)、4个TCP控制块字段和5个TCP统计结构计数器。

tcp\_slowtimo函数也作了修改，以500ms为时间单位计量每个TCP连接的持续时间。这个持续时间决定了T/TCP能否在主动关闭时截断TIME\_WAIT状态的保持时间。