

第8章 Traceroute程序

8.1 引言

由Van Jacobson编写的Traceroute程序是一个能更深入探索TCP/IP协议的方便可用的工具。尽管不能保证从源端发往目的端的两份连续的IP数据报具有相同的路由，但是大多数情况下是这样的。Traceroute程序可以让我们看到IP数据报从一台主机传到另一台主机所经过的路由。Traceroute程序还可以让我们使用IP源路由选项。

使用手册上说：“程序由Steve Deering提议，由Van Jacobson实现，并由许多其他人根据C. Philip Wood, Tim Seaver 及Ken Adelman等人提出的令人信服的建议或补充意见进行调试。”

8.2 Traceroute程序的操作

在7.3节中，我们描述了IP记录路由选项（RR）。为什么不使用这个选项而另外开发一个新的应用程序？有三个方面的原因。首先，原先并不是所有的路由器都支持记录路由选项，因此该选项在某些路径上不能使用（Traceroute程序不需要中间路由器具备任何特殊的或可选的功能）。

其次，记录路由一般是单向的选项。发送端设置了该选项，那么接收端不得不从收到的IP首部中提取出所有的信息，然后全部返回给发送端。在7.3节中，我们看到大多数Ping服务器的实现（内核中的ICMP回显应答功能）把接收到的RR清单返回，但是这样使得记录下来的IP地址翻了一番（一来一回）。这样做会受到一些限制，这一点我们在下一段讨论（Traceroute程序只需要目的端运行一个UDP模块——其他不需要任何特殊的服务器应用程序）。

最后一个原因也是最主要的原因是，IP首部中留给选项的空间有限，不能存放当前大多数的路径。在IP首部选项字段中最多只能存放9个IP地址。在原先的ARPANET中这是足够的，但是对现在来说是远远不够的。

Traceroute程序使用ICMP报文和IP首部中的TTL字段（生存周期）。TTL字段是由发送端初始设置一个8 bit字段。推荐的初始值由分配数字RFC指定，当前值为64。较老版本的系统经常初始化为15或32。我们从第7章中的一些ping程序例子中可以看出，发送ICMP回显应答时经常把TTL设为最大值255。

每个处理数据报的路由器都需要把TTL的值减1或减去数据报在路由器中停留的秒数。由于大多数的路由器转发数据报的时延都小于1秒钟，因此TTL最终成为一个跳站的计数器，所经过的每个路由器都将其值减1。

RFC 1009 [Braden and Postel 1987]指出，如果路由器转发数据报的时延超过1秒，那么它会将TTL值减去所消耗的时间（秒数）。但很少有路由器这么实现。新的路由器需求文档RFC [Almquist 1993]为此指定它为可选择功能，允许把TTL看成一个跳站计数器。

TTL字段的目的是防止数据报在选路时无休止地在网络中流动。例如, 当路由器瘫痪或者两个路由器之间的连接丢失时, 选路协议有时会去检测丢失的路由并一直进行下去。在这段时间内, 数据报可能在循环回路被终止。TTL字段就是在这些循环传递的数据报上加上一个生存上限。

当路由器收到一份IP数据报, 如果其TTL字段是0或1, 则路由器不转发该数据报(接收到这种数据报的目的主机可以将其交给应用程序, 这是因为不需要转发该数据报。但是在通常情况下, 系统不应该接收TTL字段为0的数据报)。相反, 路由器将该数据报丢弃, 并给信源机发一份ICMP“超时”信息。Traceroute程序的关键在于包含这份ICMP信息的IP报文的信源地址是该路由器的IP地址。

我们现在可以猜想一下Traceroute程序的操作过程。它发送一份TTL字段为1的IP数据报给目的主机。处理这份数据报的第一个路由器将TTL值减1, 丢弃该数据报, 并发回一份超时ICMP报文。这样就得到了该路径中的第一个路由器的地址。然后Traceroute程序发送一份TTL值为2的数据报, 这样我们就可以得到第二个路由器的地址。继续这个过程直至该数据报到达目的主机。但是目的主机哪怕接收到TTL值为1的IP数据报, 也不会丢弃该数据报并产生一份超时ICMP报文, 这是因为数据报已经到达其最终目的地。那么我们该如何判断是否已经到达目的主机了呢?

Traceroute程序发送一份UDP数据报给目的主机, 但它选择一个不可能的值作为UDP端口号(大于30 000), 使目的主机的任何一个应用程序都不可能使用该端口。因为, 当该数据报到达时, 将使目的主机的UDP模块产生一份“端口不可达”错误(见6.5节)的ICMP报文。这样, Traceroute程序所要做的就是区分接收到的ICMP报文是超时还是端口不可达, 以判断什么时候结束。

Traceroute程序必须可以为发送的数据报设置TTL字段。并非所有与TCP/IP接口的程序都支持这项功能, 同时并非所有的实现都支持这项能力, 但目前大部分系统都支持这项功能, 并可以运行Traceroute程序。这个程序界面通常要求用户具有超级用户权限, 这意味着它可能需要特殊的权限以在你的主机上运行该程序。

8.3 局域网输出

现在已经做好运行Traceroute程序并观察其输出的准备了。我们将使用从svr4到slip, 经路由器bsdi的简单互联网(见内封面)。bsdi和slip之间是9600 b/s的SLIP链路。

```
svr4 % traceroute slip
traceroute to slip (140.252.13.65), 30 hops max, 40 byte packets
 1 bsdi (140.252.13.35)  20 ms  10 ms  10 ms
 2 slip (140.252.13.65)  120 ms  120 ms  120 ms
```

输出的第1个无标号行给出了目的主机名和其IP地址, 指出traceroute程序最大的TTL字段值为30。40字节的数据报包含20字节IP首部、8字节的UDP首部和12字节的用户数据(12字节的用户数据包含每发一个数据报就加1的序列号, 送出TTL的副本以及发送数据报的时间)。

输出的后面两行以TTL开始, 接下来是主机或路由器名以及其IP地址。对于每个TTL值, 发送3份数据报。每接收到一份ICMP报文, 就计算并打印出往返时间。如果在5秒钟内仍未收到3份数据报的任意一份的响应, 则打印一个星号, 并发送下一份数据报。在上述输出结果中, TTL字段为1的前3份数据报的ICMP报文分别在20 ms、10 ms和10 ms收到。TTL字段为2的3份数

据报的ICMP报文则在120 ms后收到。由于TTL字段为2到达最终目的主机，因此程序就此停止。

往返时间是由发送主机的 traceroute 程序计算的。它是指从 traceroute 程序到该路由器的总往返时间。如果我们对每段路径的时间感兴趣，可以用 TTL 字段为 N+1 所打印出来的时间减去 TTL 字段为 N 的时间。

图8-1给出了 tcpdump 的运行输出结果。正如我们所预想的那样，第 1 个发往 bsdi 的探测数据报的往返时间是 20 ms、而后面两个数据报往返时间是 10 ms 的原因是发生了一次 ARP 交换。tcpdump 结果证实了确实是这种情况。

```

1  0.0                arp who-has bsdi tell svr4
2  0.000586 (0.0006)  arp reply bsdi is-at 0:0:c0:6f:2d:40
3  0.003067 (0.0025)  svr4.42804 > slip.33435: udp 12 [ttl 1]
4  0.004325 (0.0013)  bsdi > svr4: icmp: time exceeded in-transit
5  0.069810 (0.0655)  svr4.42804 > slip.33436: udp 12 [ttl 1]
6  0.071149 (0.0013)  bsdi > svr4: icmp: time exceeded in-transit
7  0.085162 (0.0140)  svr4.42804 > slip.33437: udp 12 [ttl 1]
8  0.086375 (0.0012)  bsdi > svr4: icmp: time exceeded in-transit
9  0.118608 (0.0322)  svr4.42804 > slip.33438: udp 12
10 0.226464 (0.1079)  slip > svr4: icmp: slip udp port 33438 unreachable
11 0.287296 (0.0608)  svr4.42804 > slip.33439: udp 12
12 0.395230 (0.1079)  slip > svr4: icmp: slip udp port 33439 unreachable
13 0.409504 (0.0143)  svr4.42804 > slip.33440: udp 12
14 0.517430 (0.1079)  slip > svr4: icmp: slip udp port 33440 unreachable

```

图8-1 从svr4到slip的traceroute程序示例的tcpdump输出结果

目的主机UDP端口号最开始设置为 33435，且每发送一个数据报加 1。可以通过命令行选项来改变开始的端口号。UDP数据报包含 12 个字节的用户数据，我们在前面 traceroute 程序输出的 40 字节数据报中已经对其进行了描述。

后面 tcpdump 打印出了 TTL 字段为 1 的 IP 数据报的注释 [ttl 1]。当 TTL 值为 0 或 1 时，tcpdump 打印出这条信息，以提示我们数据报中有些不太寻常之处。在这里可以预见到 TTL 值为 1；而在其他一些应用程序中，它可以警告我们数据报可能无法到达其最终目的主机。我们不可能看到路由器传送一个 TTL 值为 0 的数据报，除非发出该数据报的该路由器已经崩溃。

因为 bsdi 路由器将 TTL 值减到 0，因此我们预计它将发回“传送超时”的 ICMP 报文。即使这份被丢弃的 IP 报文发送往 slip，路由器也会发回 ICMP 报文。

有两种不同的 ICMP “超时”报文（见 6.2 节的图 6-3），它们的 ICMP 报文中 code 字段不同。图 8-2 给出了这种 ICMP 差错报文的格式。

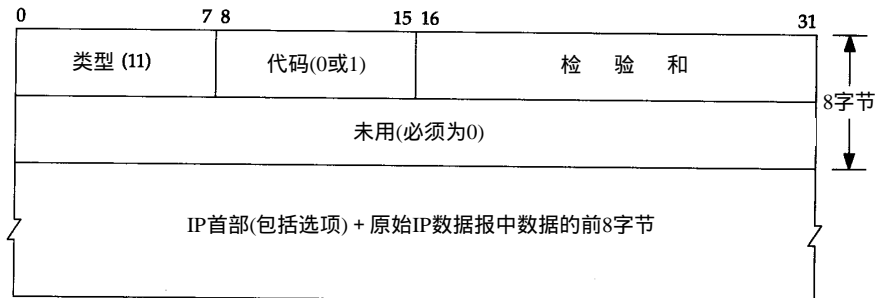


图8-2 ICMP超时报文

我们所讨论的ICMP报文是在TTL值等于0时产生的, 其code字段为0。

主机在组装分片时可能发生超时, 这时, 它将发送一份“组装报文超时”的ICMP报文(我们将在11.5节讨论分片和组装)。这种差错报文将code字段置1。

图8-1的第9~14行对应于TTL为2的3份数据报。这3份报文到达最终目的主机, 并产生一份ICMP端口不可达报文。

计算出SLIP链路的往返时间是很有意义的, 就象我们在7.2节中所举的Ping例子, 将链路值设置为1200b/s一样。发送出的UDP数据报共42个字节, 包括12字节的数据、8字节UDP首部、20字节的IP首部以及(至少)2字节的SLIP帧(2.4节)。但是与Ping不一样的是, 返回的数据报大小是变化的。从图6-9可以看出, 返回的ICMP报文包含发生差错的数据报的IP首部以及紧随该IP首部的8字节数据(在traceroute程序中, 即UDP首部)。这样, 总共就是 $20 + 8 + 20 + 8 + 2$, 即58字节。在数据速率为960 b/s的情况下, 预计的RTT就是 $(42 + 58/960)$, 即104 ms。这个值与svr4上所估算出来的110 ms是吻合的。

图8-1中的源端口号(42804)看起来有些大。traceroute程序将其发送的UDP数据报的源端口号设置为Unix进程号与32768之间的逻辑或值。对于在同一台主机上多次运行traceroute程序的情况, 每个进程都查看ICMP返回的UDP首部的源端口号, 并且只处理那些对自己发送应答的报文。

关于traceroute程序, 还有一些必须指出的事项。首先, 并不能保证现在的路由也是将来所要采用的路由, 甚至两份连续的IP数据报都可能采用不同的路由。如果在运行程序时, 路由发生改变, 就会观察到这种变化, 这是因为对于一个给定的TTL, 如果其路由发生变化, traceroute程序将打印出新的IP地址。

第二, 不能保证ICMP报文的路由与traceroute程序发送的UDP数据报采用同一路由。这表明所打印出来的往返时间可能并不能真正体现数据报发出和返回的时间差(如果UDP数据报从信源到路由器的时间是1秒, 而ICMP报文用另一条路由返回信源用了3秒时间, 则打印出来的往返时间是4秒)。

第三, 返回的ICMP报文中的信源IP地址是UDP数据报到达的路由器接口的IP地址。这与IP记录路由选项(7.3节)不同, 记录的IP地址指的是发送接口地址。由于每个定义的路由器都有2个或更多的接口, 因此, 从A主机到B主机上运行traceroute程序和从B主机到A主机上运行traceroute程序所得到的结果可能是不同的。事实上, 如果我们从slip主机到svr4上运行traceroute程序, 其输出结果变成了:

```
slip % traceroute svr4
traceroute to svr4 (140.252.13.34), 30 hops max, 40 byte packets
 1  bsdi (140.252.13.66)  110 ms  110 ms  110 ms
 2  svr4 (140.252.13.34)  110 ms  120 ms  110 ms
```

这次打印出来的bsdi主机的IP地址是140.252.13.66, 对应于SLIP接口; 而上次的地址是140.252.13.35, 是以太网接口地址。由于traceroute程序同时也打印出与IP地址相关的主机名, 因而主机名也可能变化(在我们的例子中, bsdi上的两个接口都采用相同的名字)。

考虑图8-3的情况。它给出了两个局域网通过一个路由器相连的情况。两个路由器通过一个点对点的链路相连。如果我们在左边LAN的一个主机上运行traceroute程序, 那么它将发现路由器的IP地址为if1和if3。但在另一种情况下, 就会发现打印出来的IP地址为if4和if2。if2和if3有着同样的网络号, 而另两个接口则有着不同的网络号。

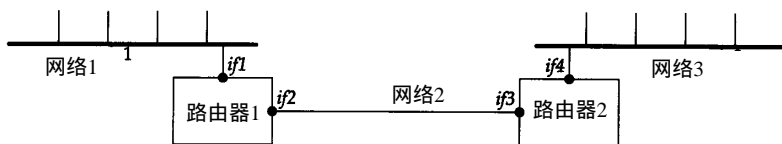


图8-3 traceroute 程序打印出的接口标识

最后，在广域网情况下，如果 traceroute 程序的输出是可读的域名形式，而不是 IP 地址形式，那么会更好理解一些。但是由于 traceroute 程序接收到 ICMP 报文时，它所获得的唯一信息就是 IP 地址，因此，在给定 IP 地址的情况下，它做一个“反向域名查看”工作来获得域名。这就需要路由器或主机的管理员正确配置其反向域名查看功能（并非所有的情况下都是如此）。我们将在 14.5 节描述如何使用 DNS 将一个 IP 地址转换成域名。

8.4 广域网输出

前面所给出的小互联网的输出例子对于查看协议运行过程来说是足够了，但对于像全球互联网这样的大互联网来说，应用 traceroute 程序就需要一些更为实际的东西。

图8-4是从sun主机到NIC (Network Information Center)的情况。

```
sun % traceroute nic.ddn.mil
traceroute to nic.ddn.mil (192.112.36.5), 30 hops max, 40 byte packets
 1  netb.tuc.noao.edu (140.252.1.183)  218 ms  227 ms  233 ms
 2  gateway.tuc.noao.edu (140.252.1.4)  233 ms  229 ms  204 ms
 3  butch.telcom.arizona.edu (140.252.104.2)  204 ms  228 ms  234 ms
 4  Gabby.Telcom.Arizona.EDU (128.196.128.1)  234 ms  228 ms  204 ms
 5  NSIgate.Telcom.Arizona.EDU (192.80.43.3)  233 ms  228 ms  234 ms
 6  JPL1.NSN.NASA.GOV (128.161.88.2)  234 ms  590 ms  262 ms
 7  JPL3.NSN.NASA.GOV (192.100.15.3)  238 ms  223 ms  234 ms
 8  GSFC3.NSN.NASA.GOV (128.161.3.33)  293 ms  318 ms  324 ms
 9  GSFC8.NSN.NASA.GOV (192.100.13.8)  294 ms  318 ms  294 ms
10  SURA2.NSN.NASA.GOV (128.161.166.2)  323 ms  319 ms  294 ms
11  nsn-FIX-pe.sura.net (192.80.214.253)  294 ms  318 ms  294 ms
12  GSI.NSN.NASA.GOV (128.161.252.2)  293 ms  318 ms  324 ms
13  NIC.DDN.MIL (192.112.36.5)  324 ms  321 ms  324 ms
```

图8-4 从sun主机到nic.ddn.mil 的traceroute 程序

由于运行的这个例子包含文本，非 DDN 站点（如，非军方站点）的 NIC 已经从 nic.ddn.mil 转移到 rs.internic.net，即新的“InterNIC”。

一旦数据报离开 tuc.noao.edu 网，它们就进入了 telcom.arizona.edu 网络。然后这些数据报进入 NASA Science Internet，nsn.nasa.gov。TTL 字段为 6 和 7 的路由器位于 JPL (Jet Propulsion Laboratory) 上。TTL 字段为 11 所输出的 sura.net 网络位于 Southeastern Universities Research Association Network 上。TTL 字段为 12 的域名 GSI 是 Government Systems, Inc., NIC 的运营者。

TTL 字段为 6 的第 2 个 RTT (590) 几乎是其他两个 RTT 值 (234 和 262) 的两倍。它表明 IP 路由的动态变化。在发送主机和这个路由器之间发生了使该数据报速度变慢的事件。同样，我们不能区分是发出的数据报还是返回的 ICMP 差错报文被拦截。

TTL 字段为 3 的第 1 个 RTT 探测值 (204) 比 TTL 字段为 2 的第 1 个探测值 (233) 值还小。由

于每个打印出来的RTT值是从发送主机到路由器的总时间, 因此这种情况是可能发生的。

图8-5的例子是从sun主机到作者出版商之间的运行例子。

```
sun % traceroute aw.com
traceroute to aw.com (192.207.117.2), 30 hops max, 40 byte packets

 1  netb.tuc.noao.edu (140.252.1.183)  227 ms  227 ms  234 ms
 2  gateway.tuc.noao.edu (140.252.1.4)  233 ms  229 ms  234 ms

 3  butch.telcom.arizona.edu (140.252.104.2)  233 ms  229 ms  234 ms
 4  Gabby.Telcom.Arizona.EDU (128.196.128.1)  264 ms  228 ms  234 ms
 5  Westgate.Telcom.Arizona.EDU (192.80.43.2)  234 ms  228 ms  234 ms

 6  uu-ua.AZ.westnet.net (192.31.39.233)  263 ms  258 ms  264 ms
 7  enss142.UT.westnet.net (192.31.39.21)  263 ms  258 ms  264 ms

 8  t3-2.Denver-cnss97.t3.ans.net (140.222.97.3)  293 ms  288 ms  275 ms
 9  t3-3.Denver-cnss96.t3.ans.net (140.222.96.4)  283 ms  263 ms  261 ms
10  t3-1.St-Louis-cnss80.t3.ans.net (140.222.80.2)  282 ms  288 ms  294 ms
11  t3-1.Chicago-cnss24.t3.ans.net (140.222.24.2)  293 ms  288 ms  294 ms
12  t3-2.Cleveland-cnss40.t3.ans.net (140.222.40.3)  294 ms  288 ms  294 ms
13  t3-1.New-York-cnss32.t3.ans.net (140.222.32.2)  323 ms  318 ms  324 ms
14  t3-1.Washington-DC-cnss56.t3.ans.net (140.222.56.2)  323 ms  318 ms  324 ms
15  t3-0.Washington-DC-cnss58.t3.ans.net (140.222.58.1)  324 ms  318 ms  324 ms
16  t3-0.enss136.t3.ans.net (140.222.136.1)  323 ms  318 ms  324 ms

17  Washington.DC.ALTER.NET (192.41.177.248)  323 ms  377 ms  324 ms
18  Boston.MA.ALTER.NET (137.39.12.2)  324 ms  347 ms  324 ms
19  AW-gw.ALTER.NET (137.39.62.2)  353 ms  378 ms  354 ms

20  aw.com (192.207.117.2)  354 ms  349 ms  354 ms
```

图8-5 从sun.tuc.noao.edu 主机到aw.com 的traceroute 程序

在这个例子中, 数据报离开 telcom.arizona.edu网络后就进行了地区性的网络 westnet.net (TTL字段值为6和7)。然后进行了由 Advanced Network & Services 运营的 NSFNET主干网, t3.ans.net, (T3是对于主干网采用的45 Mb/s电话线的一般缩写。)最后的网络是alter.net, 即aw.com与互联网的连接点。

8.5 IP源站选路选项

通常IP路由是动态的, 即每个路由器都要判断数据报下面该转发到哪个路由器。应用程序对此不进行控制, 而且通常也并不关心路由。它采用类似 Traceroute程序的工具来发现实际的路由。

源站选路(source routing)的思想是由发送者指定路由。它可以采用以下两种形式:

- 严格的源路由选择。发送端指明IP数据报所必须采用的确切路由。如果一个路由器发现源路由所指定的下一个路由器不在其直接连接的网络上, 那么它就返回一个“源站路由失败”的ICMP差错报文。
- 宽松的源站选路。发送端指明了一个数据报经过的IP地址清单, 但是数据报在清单上指明的任意两个地址之间可以通过其他路由器。

Traceroute程序提供了一个查看源站选路的方法, 我们可以在选项中指明源站路由, 然后检查其运行情况。

一些公开的Traceroute程序源代码包中包含指明宽松的源站选路的补丁。但是在标准版中通常并不包含此项。这些补丁的解释是“Van Jacobson的原始Traceroute程序

(1988年春)支持该特性,但后来因为有人提出会使网关崩溃而将此功能去除。”对于本章中所给出的例子,作者将这些补丁安装上去,并将它们设置成允许宽松的源站选路和严格的源站选路。

图8-6给出了源站路由选项的格式。

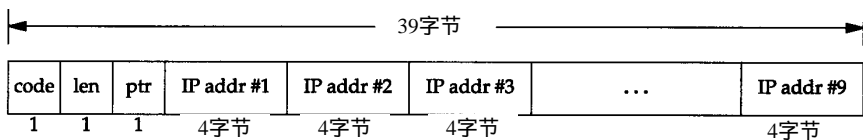


图8-6 IP首部源站路由选项的通用格式

这个格式与我们在图7-3中所示的记录路由选项格式基本一致。不同之处是,对于源站选路,我们必须在发送IP数据报前填充IP地址清单;而对于记录路由选项,我们需要为IP地址清单分配并清空一些空间,并让路由器填充该清单中的各项。同时,对于源站选路,只要为所需要的IP地址数分配空间并进行初始化,通常其数量小于9。而对于记录路由选项来说,必须尽可能地分配空间,以达到9个地址。

对于宽松的源站选路来说,code字段的值是0x83;而对于严格的源站选路,其值为0x89。len和ptr字段与7.3节中所描述的一样。

源站路由选项的实际称呼为“源站及记录路由”(对于宽松的源站选路和严格的源站选路,分别用LSRR和SSRR表示),这是因为在数据报沿路由发送过程中,对IP地址清单进行了更新。下面是其运行过程:

- 发送主机从应用程序接收源站路由清单,将第1个表项去掉(它是数据报的最终目的地),将剩余的项移到1个项中(如图8-6所示),并将原来的目的地址作为清单的最后一项。指针仍然指向清单的第1项(即,指针的值为4)。
- 每个处理数据报的路由器检查其是否为数据报的最终地址。如果不是,则正常转发数据报(在这种情况下,必须指明宽松源站选路,否则就不能接收到该数据报)。
- 如果该路由器是最终目的,且指针不大于路径的长度,那么(1)由ptr所指定的清单中的下一个地址就是数据报的最终目的地;(2)由外出接口(outgoing interface)相对应的IP地址取代刚才使用的源地址;(3)指针加4。

可以用下面这个例子很好地解释上述过程。在图8-7中,我们假设主机S上的发送应用程序发送一份数据报给D,指定源路由为R1,R2和R3。

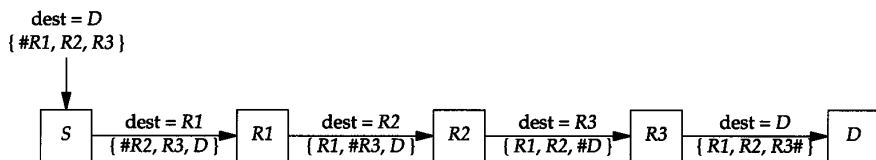


图8-7 IP源路由示例

在上图中,#表示指针字段,其值分别是4、8、12和16。长度字段恒为15(三个IP地址加上三个字节首部)。可以看出,每一跳IP数据报中的目的地址都发生改变。

当一个应用程序接收到由信源指定路由的数据时,在发送应答时,应该读出接收到的路由值,并提供反向路由。

Host Requirements RFC指明, TCP客户必须能指明源站选路, 同时, TCP服务器必须能够接收源站选路, 并且对于该 TCP连接的所有报文段都能采用反向路由。如果 TCP服务器下面接收到一个不同的源站选路, 那么新的源站路由将取代旧的源站路由。

8.5.1 宽松的源站选路的traceroute程序示例

使用traceroute程序的-g选项, 可以为宽松的源站选路指明一些中间路由器。采用该选项可以最多指定8个中间路由器(其个数是8而不是9的原因是, 所使用的编程接口要求最后的表目是目的主机)。

在图8-4中, 去往NIC, 即nic.ddn.mil的路由经过NASA Science Internet。在图8-8中, 我们通过指定路由器 enss142.UT.westnet.net (192.31.39.21) 作为中间路由器来强制数据报通过NSFNET:

```
sun % traceroute -g 192.31.39.21 nic.ddn.mil
traceroute to nic.ddn.mil (192.112.36.5), 30 hops max, 40 byte packets
 1  netb.tuc.noao.edu (140.252.1.183)  259 ms  256 ms  235 ms
 2  butch.telcom.arizona.edu (140.252.104.2)  234 ms  228 ms  234 ms
 3  Gabby.Telcom.Arizona.EDU (128.196.128.1)  234 ms  257 ms  233 ms
 4  enss142.UT.westnet.net (192.31.39.21)  294 ms  288 ms  295 ms
 5  t3-2.Denver-cnss97.t3.ans.net (140.222.97.3)  294 ms  286 ms  293 ms
 6  t3-3.Denver-cnss96.t3.ans.net (140.222.96.4)  293 ms  288 ms  294 ms
 7  t3-1.St-Louis-cnss80.t3.ans.net (140.222.80.2)  294 ms  318 ms  294 ms
 8  * t3-1.Chicago-cnss24.t3.ans.net (140.222.24.2)  318 ms  295 ms
 9  t3-2.Cleveland-cnss40.t3.ans.net (140.222.40.3)  319 ms  318 ms  324 ms
10  t3-1.New-York-cnss32.t3.ans.net (140.222.32.2)  324 ms  318 ms  324 ms
11  t3-1.Washington-DC-cnss56.t3.ans.net (140.222.56.2)  353 ms  348 ms  325 ms
12  t3-0.Washington-DC-cnss58.t3.ans.net (140.222.58.1)  348 ms  347 ms  325 ms
13  t3-0.enss145.t3.ans.net (140.222.145.1)  353 ms  348 ms  325 ms
14  nsn-FIX-pe.sura.net (192.80.214.253)  353 ms  348 ms  325 ms
15  GSI.NSN.NASA.GOV (128.161.252.2)  353 ms  348 ms  354 ms
16  NIC.DDN.MIL (192.112.36.5)  354 ms  347 ms  354 ms
```

图8-8 采用宽松源站选路通过NSFNET到达nic.ddn.mil 的traceroute 程序

在这种情况下, 看起来路径中共有 16跳, 其平均RTT大约是350 ms。而图8-4的通常选路则只有13跳, 其平均RTT约为322 ms。默认路径看起来更好一些(在建立路径时, 还需要考虑其他的一些因素。其中一些必须考虑的因素是所包含网络的组织及政治因素)。

前面我们说看起来有 16跳, 这是因为将其输出结果与前面的通过 NSFNET(图8-5)的示例比较, 发现在本例采用宽松源路由, 选择了 3个路由器(这可能是因为路由器对源站选路数据报产生ICMP超时差错报文上存在一些差错)。在netb和butch路由器之间的gateway.tuc.noao.edu路由器丢失了, 同时, 位于 Gabby和enss142.UT.west.net之间的Westgate.Telcom.Arizona.edu和uu-ua.AZ.westnet.net两个路由器也丢失了。在这些丢失的路由器上可能发生了与接收到宽松的源站选路选项数据报有关的程序问题。实际上, 当采用NSFNET时, 信源和NIC之间的路径有19跳。本章习题8.5继续对这些丢失路由器进行讨论。

同时本例也指出了另一个问题。在命令行, 我们必须指定路由器 enss142.UT.westnet.net的点分十进制IP地址, 而不能以其域名代替。这是因为, 反向域名解析(14.5节中描述的通过IP

地址返回域名)将域名与IP地址相关联,但是前向解析(即给出域名返回IP地址)则无法做到。在DNS中,前向映射和反向映射是两个独立的文件,而并非所有的管理者都同时拥有这两个文件。因此,在一个方向是工作正常而另一个方向却失败的情况并不少见。

还有一种以前没有碰到过的情况是在TTL字段为8的情况下,对于第一个RTT,打印一个星号。这表明,发生超时,在5秒内未收到本次探查的应答信号。

将本图与图8-4相比较,还可以得出一个结论,即路由器 nsn-FIX-pe.sura.net 同时与NSFNET和NASA Science Internet相连。

8.5.2 严格的源站选路的traceroute程序示例

在作者的traceroute程序版本中, - G选项与前面所描述的 - g选项是完全一样的,不过此时是严格的源站选路而不是宽松的源站选路。我们可以采用这个选项来观察在指明无效的严格的源站选路时其结果会是什么样的。从图 8-5可以看出来,从作者的子网发往NSFNET的数据报的正常路由器顺序是 netb, gateway, butch和gabby(为了便于查看,后面所有的输出结果中,均省略了域名后缀 .tuc.noao.edu和.telcom.arizona.edu)。我们指定了一个严格源路由,使其试图将数据报从 gateway直接发送到 gabby,而省略了 butch。我们可以猜测到其结果会是失败的,正如图 8-9所给出的结果。

```
sun % traceroute -G netb -G gateway -G gabby westgate
traceroute to westgate (192.80.43.2), 30 hops max, 40 byte packets
 1 netb (140.252.1.183)  272 ms  257 ms  261 ms
 2 gateway (140.252.1.4)  263 ms  259 ms  234 ms
 3 gateway (140.252.1.4)  263 ms !S *  235 ms !S
```

图8-9 采用严格源站路由失败的traceroute程序

这里的关键是在于TTL字段为3的输出行中,RTT后面的!S。这表明traceroute程序接收到ICMP“源站路由失败”的差错报文:即图6-3中type字段为3,而code字段为5。TTL字段为3的第二个RTT位置的星号表示未收到这次探查的应答信号。这与我们所猜想的一样, gateway不可能直接发送数据报给 gabby,这是因为它们之间没有直接连接。

TTL字段为2和3的结果都来自于 gateway,对于TTL字段为2的应答来自 gateway,是因为 gateway接收到TTL字段为1的数据报。在它查看到(无效的)严格的源站选路之前,就发现TTL已过期,因此发送回ICMP超时报文。TTL字段等于3的行,在进入 gateway时其TTL字段为2,因此,它查看严格的源站选路,发现它是无效的,因此发送回ICMP源站选路失败的差错报文。

图8-10给出了与本例相对应的tcpdump输出结果。该输出结果是在 sun和netb之间的SLIP链路上遇到的。我们必须在tcpdump中指定 - v选项以显示出源站路由信息。这样,会输出一些像数据报ID这样我们并不需要的结果,我们在给出结果中将这些不需要的结果删除掉。同样,用SSRR表示“严格的源站及记录路由”。

首先注意到, sun所发送的每个UDP数据报的目的地址都是 netb,而不是目的主机(westgate)。这一点可以用图8-7的例子来解释。类似地, - G选项所指定的另外两个路由器(gateway和gabby)以及最终目(westgate)成为第一跳的SSRR选项。

从这个输出结果中,还可以看出, traceroute程序所采用的定时时间(第15行和16行

之间的时间差) 是5秒。

```

1 0.0 sun.33593 > netb.33435: udp 12 [ttl 1]
   (optlen=16 SSRR{#gateway gabby westgate} EOL)
2 0.270278 (0.2703) netb > sun: icmp: time exceeded in-transit
3 0.284784 (0.0145) sun.33593 > netb.33436: udp 12 [ttl 1]
   (optlen=16 SSRR{#gateway gabby westgate} EOL)
4 0.540338 (0.2556) netb > sun: icmp: time exceeded in-transit
5 0.550062 (0.0097) sun.33593 > netb.33437: udp 12 [ttl 1]
   (optlen=16 SSRR{#gateway gabby westgate} EOL)
6 0.810310 (0.2602) netb > sun: icmp: time exceeded in-transit
7 0.818030 (0.0077) sun.33593 > netb.33438: udp 12 (ttl 2,
   optlen=16 SSRR{#gateway gabby westgate} EOL)
8 1.080337 (0.2623) gateway > sun: icmp: time exceeded in-transit
9 1.092564 (0.0122) sun.33593 > netb.33439: udp 12 (ttl 2,
   optlen=16 SSRR{#gateway gabby westgate} EOL)
10 1.350322 (0.2578) gateway > sun: icmp: time exceeded in-transit
11 1.357382 (0.0071) sun.33593 > netb.33440: udp 12 (ttl 2,
   optlen=16 SSRR{#gateway gabby westgate} EOL)
12 1.590586 (0.2332) gateway > sun: icmp: time exceeded in-transit
13 1.598926 (0.0083) sun.33593 > netb.33441: udp 12 (ttl 3,
   optlen=16 SSRR{#gateway gabby westgate} EOL)
14 1.860341 (0.2614) gateway > sun:
   icmp: gateway unreachable - source route failed
15 1.875230 (0.0149) sun.33593 > netb.33442: udp 12 (ttl 3,
   optlen=16 SSRR{#gateway gabby westgate} EOL)
16 6.876579 (5.0013) sun.33593 > netb.33443: udp 12 (ttl 3,
   optlen=16 SSRR{#gateway gabby westgate} EOL)
17 7.110518 (0.2339) gateway > sun:
   icmp: gateway unreachable - source route failed

```

图8-10 失败的严格源站选路traceroute 程序的tcpdump 输出结果

8.5.3 宽松的源站选路traceroute程序的往返路由

我们在前面已经说过, 从A到B的路径并不一定与从B到A的路径完全一样。除非同时在两个系统中登录并在每个终端上运行 traceroute 程序, 否则很难发现两条路径是否不同。但是, 采用宽松的源站选路, 就可以决定两个方向上的路径。

这里的窍门就在于指定一个宽松的源站路由, 该路由的目的端和宽松路径一样, 但发送端为目的主机。例如, 在sun主机上, 我们可以查看到发往以及来自bruno.cs.colorado.edu的结果如图8-11所示。

发出路径 (TTL字段为1~11) 的结果与返回路径 (TTL字段为11~21) 不同, 这很好地说明了在Internet上, 选路可能是不对称的。

该输出同时还说明了我们在图8-3中所讨论的问题。比较TTL字段为2和19的输出结果: 它们都是路由器gateway.tuc.noao.edu, 但两个IP地址却是不同的。由于traceroute程序以进入接口作为其标识, 而我们从两条不同的方向经过该路由器, 一条是发出路径 (TTL字段为2), 另一条是返回路径 (TTL字段为19), 因此可以猜想到这个结果。通过比较TTL字段为3和18、4和17的结果, 可以看到同样的结果。

```
sun % traceroute -g bruno.cs.colorado.edu sun
traceroute to sun (140.252.13.33), 30 hops max, 40 byte packets
 1  netb.tuc.noao.edu (140.252.1.183)  230 ms  227 ms  233 ms
 2  gateway.tuc.noao.edu (140.252.1.4)  233 ms  229 ms  234 ms
 3  butch.telcom.arizona.edu (140.252.104.2)  234 ms  229 ms  234 ms
 4  Gabby.Telcom.Arizona.EDU (128.196.128.1)  233 ms  231 ms  234 ms
 5  NSigate.Telcom.Arizona.EDU (192.80.43.3)  294 ms  258 ms  234 ms
 6  JPL1.NSN.NASA.GOV (128.161.88.2)  264 ms  258 ms  264 ms
 7  JPL2.NSN.NASA.GOV (192.100.15.2)  264 ms  258 ms  264 ms
 8  NCAR.NSN.NASA.GOV (128.161.97.2)  324 ms * 295 ms
 9  cu-gw.ucar.edu (192.43.244.4)  294 ms  318 ms  294 ms
10  engr-gw.Colorado.EDU (128.138.1.3)  294 ms  288 ms  294 ms
11  bruno.cs.colorado.edu (128.138.243.151)  293 ms  317 ms  294 ms
12  engr-gw-ot.cs.colorado.edu (128.138.204.1)  323 ms  317 ms  384 ms
13  cu-gw.Colorado.EDU (128.138.1.1)  294 ms  318 ms  294 ms
14  enss.ucar.edu (192.43.244.10)  323 ms  318 ms  294 ms
15  t3-1.Denver-cnss97.t3.ans.net (140.222.97.2)  294 ms  288 ms  384 ms
16  t3-0.enss142.t3.ans.net (140.222.142.1)  293 ms  288 ms  294 ms
17  Gabby.Telcom.Arizona.EDU (192.80.43.1)  294 ms  288 ms  294 ms
18  Butch.Telcom.Arizona.EDU (128.196.128.88)  293 ms  317 ms  294 ms
19  gateway.tuc.noao.edu (140.252.104.1)  294 ms  289 ms  294 ms
20  netb.tuc.noao.edu (140.252.1.183)  324 ms  321 ms  294 ms
21  sun.tuc.noao.edu (140.252.13.33)  534 ms  529 ms  564 ms
```

图8-11 显示非对称路径的traceroute 程序

8.6 小结

在一个TCP/IP网络中，traceroute程序是不可缺少的工具。其操作很简单：开始时发送一个TTL字段为1的UDP数据报，然后将TTL字段每次加1，以确定路径中的每个路由器。每个路由器在丢弃UDP数据报时都返回一个ICMP超时报文，而最终目的主机则产生一个ICMP端口不可达的报文。

我们给出了在LAN和WAN上运行traceroute程序的例子，并用它来考察IP源站选路。我们用宽松的源站选路来检测发往目的主机的路由是否与从目的主机返回的路由一样。

习题

- 8.1 当IP将接收到的TTL字段减1，发现它为0时，将会发生什么结果？
- 8.2 traceroute程序是如何计算RTT的？将这种计算RTT的方法与ping相比较。
- 8.3 （本习题与下一道习题是基于开发traceroute程序过程中遇到的实际问题，它们来自于traceroute程序源代码注释）。假设源主机和目的主机之间有三个路由器（R1、R2和R3），而中间的路由器（R2）在进入TTL字段为1时，将TTL字段减1，但却错误地将该IP数据报发往下一个路由器。请描述会发生什么结果。在运行traceroute程序时会看到什么样的现象？
- 8.4 同样，假设源主机和目的主机之间有三个路由器。由于目的主机上存在错误，因此，它总是将进入TTL值作为外出ICMP报文的TTL值。请描述这将发生什么结果，你会看到什么现象。

- 8.5 在图8-8运行例子中, 我们可以在 sun和netb之间的SLIP链路上运行tcpdump程序。如果指定 - v选项, 就可以看到返回 ICMP报文的TTL值。这样, 我们可以看到进入 netb、butch、Gabby和enss142.UT.westnet.net的TTL值分别为255、253、252和249。这是否为我们判断是否存在丢失路由器提供了额外的信息?
- 8.6 SunOS和SVR4都提供了带 - l选项的ping版本, 以提供松源选路。手册上说明, 该选项可以与 - R选项 (指定记录路由选项) 一起使用。如果已经进入到这些系统中, 请尝试同时用这两个选项。其结果是什么? 如果采用 tcpdump来观测数据报, 请描述其过程。
- 8.7 比较ping和traceroute程序在处理同一台主机上客户的多个实例的不同点。
- 8.8 比较ping和traceroute程序在计算往返时间上的不同点。
- 8.9 我们已经说过, traceroute程序选取开始UDP目的主机端口号为 33453, 每发送一个数据报将此数加 1。在 1.9节中, 我们说过暂时端口号通常是 1024~5000之间的值, 因此 traceroute程序的目的地主机端口号不可能是目的地主机上所使用的端口号。在 Solaris2.2系统中的情况也是如此吗? (提示: 查看 E.4节)
- 8.10 RFC 1393 [Malkin 1993b]提出了另一种判断到目的地主机路径的方法。请问其优缺点是什么?