

第7章 Ping程序

7.1 引言

“ping”这个名字源于声纳定位操作。Ping程序由Mike Muuss编写，目的是为了测试另一台主机是否可达。该程序发送一份ICMP回显请求报文给主机，并等待返回ICMP回显应答（图6-3列出了所有的ICMP报文类型）。

一般来说，如果不能Ping到某台主机，那么就不能Telnet或者FTP到那台主机。反过来，如果不能Telnet到某台主机，那么通常可以用Ping程序来确定问题出在哪里。Ping程序还能测出这台主机的往返时间，以表明该主机离我们有“多远”。

在本章中，我们将使用Ping程序作为诊断工具来深入剖析ICMP。Ping还给我们提供了检测IP记录路由和时间戳选项的机会。文献[Stevens 1990]的第11章提供了Ping程序的源代码。

几年前我们还可以作出这样没有限定的断言，如果不能Ping到某台主机，那么就不能Telnet或FTP到那台主机。随着Internet安全意识的增强，出现了提供访问控制清单的路由器和防火墙，那么像这样没有限定的断言就不再成立了。一台主机的可达性可能不只取决于IP层是否可达，还取决于使用何种协议以及端口号。Ping程序的运行结果可能显示某台主机不可达，但我们可以用Telnet远程登录到该台主机的25号端口（邮件服务器）。

7.2 Ping程序

我们称发送回显请求的ping程序为客户，而称被ping的主机为服务器。大多数的TCP/IP实现都在内核中直接支持Ping服务器——这种服务器不是一个用户进程（在第6章中描述的两类ICMP查询服务，地址掩码和时间戳请求，也都是直接在内核中处理的）。

ICMP回显请求和回显应答报文如图7-1所示。

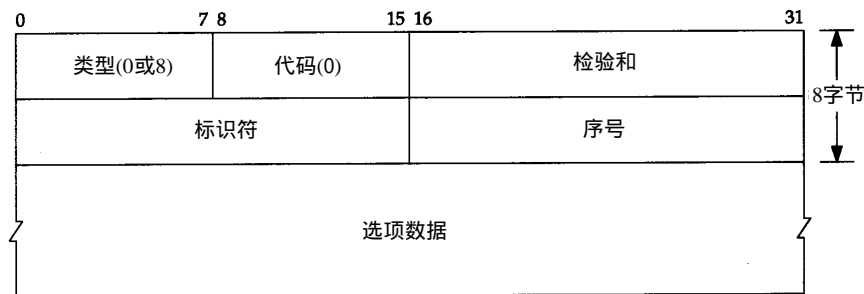


图7-1 ICMP回显请求和回显应答报文格式

对于其他类型的ICMP查询报文，服务器必须响应标识符和序列号字段。另外，客户发送的选项数据必须回显，假设客户对这些信息都会感兴趣。

Unix系统在实现ping程序时是把ICMP报文中的标识符字段置成发送进程的ID号。这样即使在同一台主机上同时运行了多个ping程序实例, ping程序也可以识别出返回的信息。

序列号从0开始, 每发送一次新的回显请求就加1。ping程序打印出返回的每个分组的序列号, 允许我们查看是否有分组丢失、失序或重复。IP是一种最好的数据报传递服务, 因此这三个条件都有可能发生。

旧版本的ping程序曾经以这种模式运行, 即每秒发送一个回显请求, 并打印出返回的每个回显应答。但是, 新版本的实现需要加上-s选项才能以这种模式运行。默认情况下, 新版本的ping程序只发送一个回显请求。如果收到回显应答, 则输出“host is alive”; 否则, 在20秒内没有收到应答就输出“no answer (没有回答)”。

7.2.1 LAN输出

在局域网运行ping程序的结果输出一般有如下格式:

```
bsdi % ping svr4
PING svr4 (140.252.13.34): 56 data bytes
64 bytes from 140.252.13.34: icmp_seq=0 ttl=255 time=0 ms
64 bytes from 140.252.13.34: icmp_seq=1 ttl=255 time=0 ms
64 bytes from 140.252.13.34: icmp_seq=2 ttl=255 time=0 ms
64 bytes from 140.252.13.34: icmp_seq=3 ttl=255 time=0 ms
64 bytes from 140.252.13.34: icmp_seq=4 ttl=255 time=0 ms
64 bytes from 140.252.13.34: icmp_seq=5 ttl=255 time=0 ms
64 bytes from 140.252.13.34: icmp_seq=6 ttl=255 time=0 ms
64 bytes from 140.252.13.34: icmp_seq=7 ttl=255 time=0 ms
^?                               键入中断键来停止显示
--- svr4 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 0/0/0 ms
```

当返回ICMP回显应答时, 要打印出序列号和TTL, 并计算往返时间(TTL位于IP首部中的生存时间字段。当前的BSD系统中的ping程序每次收到回显应答时都打印出收到的TTL——有些系统并不这样做。我们将在第8章中通过traceroute程序来介绍TTL的用法)。

从上面的输出中可以看出, 回显应答是以发送的次序返回的(0, 1, 2等)。

ping程序通过在ICMP报文数据中存放发送请求的时间值来计算往返时间。当应答返回时, 用当前时间减去存放在ICMP报文中的时间值, 即是往返时间。注意, 在发送端bsdi上, 往返时间的计算结果都为0 ms。这是因为程序使用的计时器分辨率低的原因。BSD/386版本0.9.4系统只能提供10 ms级的计时器(在附录B中有更详细的介绍)。在后面的章节中, 当我们在具有较高分辨率计时器的系统上(Sun)查看tcpdump输出时会发现, ICMP回显请求和回显应答的时间差在4 ms以下。

输出的第一行包括目的主机的IP地址, 尽管指定的是它的名字(svr4)。这说明名字已经经过解析器被转换成IP地址了。我们将在第14章介绍解析器和DNS。现在, 我们发现, 如果敲入ping命令, 几秒钟过后会在第1行打印出IP地址, DNS就是利用这段时间来确定主机名所对应的IP地址。

本例中的tcpdump输出如图7-2所示。

从发送回显请求到收到回显应答, 时间间隔始终为3.7 ms。还可以看到, 回显请求大约每隔1秒钟发送一次。

通常, 第1个往返时间值要比其他的大。这是由于目的端的硬件地址不在ARP高速缓存中

```

1 0.0          bsdi > svr4: icmp: echo request
2 0.003733 (0.0037) svr4 > bsdi: icmp: echo reply
3 0.998045 (0.9943) bsdi > svr4: icmp: echo request
4 1.001747 (0.0037) svr4 > bsdi: icmp: echo reply
5 1.997818 (0.9961) bsdi > svr4: icmp: echo request
6 2.001542 (0.0037) svr4 > bsdi: icmp: echo reply
7 2.997610 (0.9961) bsdi > svr4: icmp: echo request
8 3.001311 (0.0037) svr4 > bsdi: icmp: echo reply
9 3.997390 (0.9961) bsdi > svr4: icmp: echo request
10 4.001115 (0.0037) svr4 > bsdi: icmp: echo reply
11 4.997201 (0.9961) bsdi > svr4: icmp: echo request
12 5.000904 (0.0037) svr4 > bsdi: icmp: echo reply
13 5.996977 (0.9961) bsdi > svr4: icmp: echo request
14 6.000708 (0.0037) svr4 > bsdi: icmp: echo reply
15 6.996764 (0.9961) bsdi > svr4: icmp: echo request
16 7.000479 (0.0037) svr4 > bsdi: icmp: echo reply

```

图7-2 在LAN上运行ping程序的结果

的缘故。正如我们在第4章中看到的那样，在发送第一个回显请求之前要发送一个 ARP请求并接收ARP应答，这需要花费几毫秒的时间。下面的例子说明了这一点：

```
sun % arp -a
```

保证ARP高速缓存是空的

```
sun % ping svr4
```

```
PING svr4: 56 data bytes
```

```
64 bytes from svr4 (140.252.13.34): icmp_seq=0. time=7. ms
```

```
64 bytes from svr4 (140.252.13.34): icmp_seq=1. time=4. ms
```

```
64 bytes from svr4 (140.252.13.34): icmp_seq=2. time=4. ms
```

```
64 bytes from svr4 (140.252.13.34): icmp_seq=3. time=4. ms
```

```
^?
```

键入中断键来停止显示

```
----svr4 PING Statistics----
```

```
4 packets transmitted, 4 packets received, 0% packet loss
```

```
round-trip (ms)  min/avg/max = 4/4/7
```

第1个RTT中多出的3 ms很可能就是因为发送ARP请求和接收ARP应答所花费的时间。

这个例子运行在sun主机上，它提供的是具有微秒级分辨率的计时器，但是 ping程序只能打印出毫秒级的往返时间。在前面运行于BSD/386 0.9.4版上的例子中，打印出来的往返时间值为0 ms，这是因为计时器只能提供10 ms的误差。下面的例子是BSD/386 1.0版的输出，它提供的计时器也具有微秒级的分辨率，因此，ping程序的输出结果也具有较高分辨率。

```
bsdi % ping svr4
```

```
PING svr4 (140.252.13.34): 56 data bytes
```

```
64 bytes from 140.252.13.34: icmp_seq=0 ttl=255 time=9.304 ms
```

```
64 bytes from 140.252.13.34: icmp_seq=1 ttl=255 time=6.089 ms
```

```
64 bytes from 140.252.13.34: icmp_seq=2 ttl=255 time=6.079 ms
```

```
64 bytes from 140.252.13.34: icmp_seq=3 ttl=255 time=6.096 ms
```

```
^?
```

键入中断键来停止显示

```
--- svr4 ping statistics ---
```

```
4 packets transmitted, 4 packets received, 0% packet loss
```

```
round-trip min/avg/max = 6.079/6.880/9.304 ms
```

7.2.2 WAN输出

在一个广域网上，结果会有很大的不同。下面的例子是在某个工作日的下午即 Internet具

有正常通信量时的运行结果：

```
gemini % ping vangogh.cs.berkeley.edu
PING vangogh.cs.berkeley.edu: 56 data bytes
64 bytes from (128.32.130.2): icmp_seq=0. time=660. ms
64 bytes from (128.32.130.2): icmp_seq=5. time=1780. ms
64 bytes from (128.32.130.2): icmp_seq=7. time=380. ms
64 bytes from (128.32.130.2): icmp_seq=8. time=420. ms
64 bytes from (128.32.130.2): icmp_seq=9. time=390. ms
64 bytes from (128.32.130.2): icmp_seq=14. time=110. ms
64 bytes from (128.32.130.2): icmp_seq=15. time=170. ms
64 bytes from (128.32.130.2): icmp_seq=16. time=100. ms
^?
键入中断来停止显示

----vangogh.CS.Berkeley.EDU PING Statistics----
17 packets transmitted, 8 packets received, 52% packet loss
round-trip (ms)  min/avg/max = 100/501/1780
```

这里，序列号为1、2、3、4、6、10、11、12和13的回显请求或回显应答在某个地方丢失了。另外，我们注意到往返时间发生了很大的变化（像 52%这样高的分组丢失率是不正常的。即使是在工作日的下午，对于Internet来说也是不正常的）。

通过广域网还有可能看到重复的分组（即相同序列号的分组被打印两次或更多次），失序的分组（序列号为 $N+1$ 的分组在序列号为 N 的分组之前被打印）。

7.2.3 线路SLIP链接

让我们再来看看SLIP链路上的往返时间，因为它们经常运行于低速的异步方式，如 9600 b/s或更低。回想我们在 2.10节计算的串行线路吞吐量。针对这个例子，我们把主机 bsd1和slip之间的SLIP链路传输速率设置为1200 b/s。

下面我们可以来估计往返时间。首先，从前面的 Ping程序输出例子中可以注意到，默认情况下发送的ICMP报文有56个字节。再加上20个字节的IP首部和8个字节的ICMP首部，IP数据报的总长度为84字节（我们可以运行tcpdump-e命令查看以太网数据帧来验证这一点）。另外，从2.4节可以知道，至少要增加两个额外的字节：在数据报的开始和结尾加上END字符。此外，SLIP帧还有可能再增加一些字节，但这取决于数据报中每个字节的值。对于1200 b/s这个速率来说，由于每个字节含有8 bit数据、1 bit起始位和1 bit结束位，因此传输速率是每秒120个字节，或者说每个字节8.33 ms。所以我们可以估计需要1433（ $86 \times 8.33 \times 2$ ）ms（乘2是因为我们计算的是往返时间）。

下面的输出证实了我们的计算：

```
svr4 % ping -s slip
PING slip: 56 data bytes
64 bytes from slip (140.252.13.65): icmp_seq=0. time=1480. ms
64 bytes from slip (140.252.13.65): icmp_seq=1. time=1480. ms
64 bytes from slip (140.252.13.65): icmp_seq=2. time=1480. ms
64 bytes from slip (140.252.13.65): icmp_seq=3. time=1480. ms
^?
----slip PING Statistics----
5 packets transmitted, 4 packets received, 20% packet loss
round-trip (ms)  min/avg/max = 1480/1480/1480
```

（对于SVR4来说，如果每秒钟发送一次请求则必须带-s选项）。往返时间大约是1.5秒，但是程序仍然每间隔1秒钟发送一次ICMP回显请求。这说明在第1个回显应答返回之前（1.480秒时刻）就已经发送了两次回显请求（分别在0秒和1秒时刻）。这就是为什么总结行指

出丢失了一个分组。实际上分组并未丢失，很可能仍然在返回的途中。

我们在第8章讨论traceroute程序时将回头再讨论这种低速的SLIP链路。

7.2.4 拨号SLIP链路

对于拨号SLIP链路来说，情况有些变化，因为在链路的两端增加了调制解调器。用在sun和netb系统之间的调制解调器提供的是V.32调制方式（9600 b/s）、V.42错误控制方式（也称作LAP-M）以及V.42bis数据压缩方式。这表明我们针对线路链路参数进行的简单计算不再准确了。

很多因素都有可能影响。调制解调器带来了时延。随着数据的压缩，分组长度可能会减小，但是由于使用了错误控制协议，分组长度又可能会增加。另外，接收端的调制解调器只能在验证了循环检验字符（检验和）后才能释放收到的数据。最后，我们还要处理每一端的计算机异步串行接口，许多操作系统只能在固定的时间间隔内，或者收到若干字符后才去读这些接口。

作为一个例子，我们在sun主机上ping主机gemini，输出结果如下：

```
sun % ping gemini
PING gemini: 56 data bytes
64 bytes from gemini (140.252.1.11): icmp_seq=0. time=373. ms
64 bytes from gemini (140.252.1.11): icmp_seq=1. time=360. ms
64 bytes from gemini (140.252.1.11): icmp_seq=2. time=340. ms
64 bytes from gemini (140.252.1.11): icmp_seq=3. time=320. ms
64 bytes from gemini (140.252.1.11): icmp_seq=4. time=330. ms
64 bytes from gemini (140.252.1.11): icmp_seq=5. time=310. ms
64 bytes from gemini (140.252.1.11): icmp_seq=6. time=290. ms
64 bytes from gemini (140.252.1.11): icmp_seq=7. time=300. ms
64 bytes from gemini (140.252.1.11): icmp_seq=8. time=280. ms
64 bytes from gemini (140.252.1.11): icmp_seq=9. time=290. ms
64 bytes from gemini (140.252.1.11): icmp_seq=10. time=300. ms
64 bytes from gemini (140.252.1.11): icmp_seq=11. time=280. ms
---gemini PING Statistics---
12 packets transmitted, 12 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 280/314/373
```

注意，第1个RTT不是10 ms的整数倍，但是其他行都是10 ms的整数倍。如果我们运行该程序若干次，发现每次结果都是这样（这并不是由sun主机上的时钟分辨率造成的结果，因为根据附录B中的测试结果可以知道它的时钟能提供毫秒级的分辨率）。

另外还要注意，第1个RTT要比其他的大，而且依次递减，然后徘徊在280~300 ms之间。我们让它运行1~2分钟，RTT一直处于这个范围，不会低于260 ms。如果我们以9600 b/s的速率计算RTT（习题7.2），那么观察到的值应该大约是估计值的1.5倍。

如果运行ping程序60秒钟并计算观察到的RTT的平均值，我们发现在V.42和V.42bis模式下平均值为277 ms（这比上个例子打印出来的平均值要好，因为运行时间较长，这样就把开始较长的时间分摊了）。如果我们关闭V.42bis数据压缩方式，平均值为330 ms。如果我们关闭V.42错误控制方式（它同时也关闭了V.42bis数据压缩方式），平均值为300 ms。这些调制解调器的参数对RTT的影响很大，使用错误控制和数据压缩方式似乎效果最好。

7.3 IP记录路由选项

ping程序为我们提供了查看IP记录路由（RR）选项的机会。大多数不同版本的ping程

序都提供 -R 选项, 以提供记录路由的功能。它使得 ping 程序在发送出去的 IP 数据报中设置 IP RR 选项 (该 IP 数据报包含 ICMP 回显请求报文)。这样, 每个处理该数据报的路由器都把它的 IP 地址放入选项字段中。当数据报到达目的端时, IP 地址清单应该复制到 ICMP 回显应答中, 这样返回途中所经过的路由器地址也被加入清单中。当 ping 程序收到回显应答时, 它就打印出这份 IP 地址清单。

这个过程听起来简单, 但存在一些缺陷。源端主机生成 RR 选项, 中间路由器对 RR 选项的处理, 以及把 ICMP 回显请求中的 RR 清单复制到 ICMP 回显应答中, 所有这些都是选项功能。幸运的是, 现在的大多数系统都支持这些选项功能, 只是有一些系统不把 ICMP 请求中的 IP 清单复制到 ICMP 应答中。

但是, 最大的问题是 IP 首部中只有有限的空间来存放 IP 地址。我们从图 3-1 可以看到, IP 首部中的首部长度字段只有 4 bit, 因此整个 IP 首部最长只能包括 15 个 32 bit 长的字 (即 60 个字节)。由于 IP 首部固定长度为 20 字节, RR 选项用去 3 个字节 (下面我们再讨论), 这样只剩下 37 个字节 (60 - 20 - 3) 来存放 IP 地址清单, 也就是说只能存放 9 个 IP 地址。对于早期的 ARPANET 来说, 9 个 IP 地址似乎是很多了, 但是现在看来是非常有限的 (在第 8 章中, 我们将用 Traceroute 工具来确定数据报的路由)。除了这些缺点, 记录路由选项工作得很好, 为详细查看如何处理 IP 选项提供了一个机会。

IP 数据报中的 RR 选项的一般格式如图 7-3 所示。

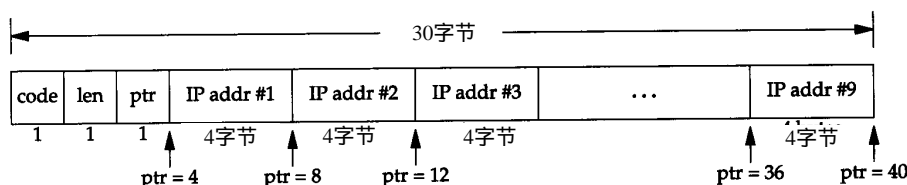


图7-3 IP首部中的记录路由选项的一般格式

code 是一个字节, 指明 IP 选项的类型。对于 RR 选项来说, 它的值为 7。len 是 RR 选项总字节长度, 在这种情况下为 39 (尽管可以为 RR 选项设置比最大长度小的长度, 但是 ping 程序总是提供 39 字节的选项字段, 最多可以记录 9 个 IP 地址。由于 IP 首部中留给选项的空间有限, 它一般情况都设置成最大长度)。

ptr 称作指针字段。它是一个基于 1 的指针, 指向存放下一个 IP 地址的位置。它的最小值为 4, 指向存放第一个 IP 地址的位置。随着每个 IP 地址存入清单, ptr 的值分别为 8, 12, 16, 最大到 36。当记录下 9 个 IP 地址后, ptr 的值为 40, 表示清单已满。

当路由器 (根据定义应该是多穴的) 在清单中记录 IP 地址时, 它应该记录哪个地址呢? 是入口地址还是出口地址? 为此, RFC 791 [Postel 1981a] 指定路由器记录出口 IP 地址。我们在后面将看到, 当原始主机 (运行 ping 程序的主机) 收到带有 RR 选项的 ICMP 回显应答时, 它也要把它的入口 IP 地址放入清单中。

7.3.1 通常的例子

我们举一个用 RR 选项运行 ping 程序的例子, 在主机 svr4 上运行 ping 程序到主机 slip。一个中间路由器 (bsdi) 将处理这个数据报。下面是 svr4 的输出结果:

```

svr4 % ping -R slip
PING slip (140.252.13.65): 56 data bytes
64 bytes from 140.252.13.65: icmp_seq=0 ttl=254 time=280 ms
RR:      bsd1 (140.252.13.66)
         slip (140.252.13.65)
         bsd1 (140.252.13.35)
         svr4 (140.252.13.34)
64 bytes from 140.252.13.65: icmp_seq=1 ttl=254 time=280 ms (same route)
64 bytes from 140.252.13.65: icmp_seq=2 ttl=254 time=270 ms (same route)
^?
--- slip ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 270/276/280 ms

```

分组所经过的四站如图 7-4 所示（每个方向各有两站），每一站都把自己的 IP 地址加入 RR 清单。

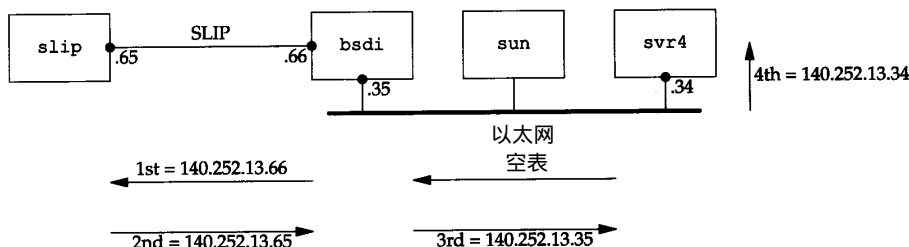


图7-4 带有记录路由选项的ping程序

路由器 bsd1 在不同方向上分别加入了不同的 IP 地址。它始终是把出口的 IP 地址加入清单。我们还可以看到，当 ICMP 回显应答到达原始系统（svr4）时，它把自己的入口 IP 地址也加入清单中。

还可以通过运行带有 -v 选项的 tcpdump 命令来查看主机 sun 上进行的分组交换（参见 IP 选项）。输出如图 7-5 所示。

```

1 0.0 svr4 > slip: icmp: echo request (ttl 32, id 35835,
    optlen=40 RR{39}= RR{#0.0.0.0/0.0.0.0/0.0.0.0/
    0.0.0.0/ 0.0.0.0/0.0.0.0/0.0.0.0/0.0.0.0/0.0.0.0} EOL)

2 0.267746 (0.2677) slip > svr4: icmp: echo reply (ttl 254, id 1976,
    optlen=40 RR{39}= RR{140.252.13.66/140.252.13.65/
    140.252.13.35/#0.0.0.0/0.0.0.0/0.0.0.0/0.0.0.0/
    0.0.0.0/0.0.0.0} EOL)

```

图7-5 记录路由选项的tcpdump 输出

输出中 optlen=40 表示在 IP 首部中有 40 个字节的选项空间（IP 首部长度必须为 4 字节的整数倍）。RR{39} 的意思是记录路由选项已被设置，它的长度字段是 39。然后是 9 个 IP 地址，符号“#”用来标记 RR 选项中的 ptr 字段所指向的 IP 地址。由于我们是在主机 sun 上观察这些分组（参见图 7-4），因此所能看到 ICMP 回显请求中的 IP 地址清单是空的，而 ICMP 回显应答中有 3 个 IP 地址。我们省略了 tcpdump 输出中的其他行，因为它们与图 7-5 基本一致。

位于路由信息末尾的标记 EOL 表示 IP 选项“end of list（清单结束）”的值。EOL 选项的值可以为 0。这时表示 39 个字节的 RR 数据位于 IP 首部中的 40 字节空间中。由于在数据报发送之前空间选项被设置为 0，因此跟在 39 个字节的 RR 数据之后的 0 字符就被解释为 EOL。这正是我

们所希望的结果。如果在 IP 首部中的选项字段中有多个选项, 在开始下一个选项之前必须填入空白字符, 另外还可以用另一个值为 1 的特殊字符 NOP (“no operation”)

在图 7-5 中, SVR4 把回显请求中的 TTL 字段设为 32, BSD/386 设为 255 (它打印出的值为 254 是因为路由器 bsd1 已经将其减去 1)。新的系统都把 ICMP 报文中的 TTL 设为最大值 (255)。

在作者使用的三个 TCP/IP 系统中, BSD/386 和 SVR4 都支持记录路由选项。这就是说, 当转发数据报时, 它们都能正确地更新 RR 清单, 而且能正确地把接收到的 ICMP 回显请求中的 RR 清单复制到出口 ICMP 回显应答中。虽然 SunOS 4.1.3 在转发一个数据报时能正确更新 RR 清单, 但是不能复制 RR 清单。Solaris 2.x 对这个问题已作了修改。

7.3.2 异常的输出

下面的例子是作者观察到的, 把它作为第 9 章讨论 ICMP 间接报文的起点。在子网 140.252.1 上 ping 主机 aix (在主机 sun 上通过拨号 SLIP 连接可以访问), 并带有记录路由选项。在 slip 主机上运行有如下输出结果:

```
slip % ping -R aix
PING aix (140.252.1.92): 56 data bytes
64 bytes from 140.252.1.92: icmp_seq=0 ttl=251 time=650 ms
RR:      bsd1 (140.252.13.35)
          sun (140.252.1.29)
          netb (140.252.1.183)
          aix (140.252.1.92)
          gateway (140.252.1.4)      为什么用这个路由器?
          netb (140.252.1.183)
          sun (140.252.13.33)
          bsd1 (140.252.13.66)
          slip (140.252.13.65)
64 bytes from aix: icmp_seq=1 ttl=251 time=610 ms (same route)
64 bytes from aix: icmp_seq=2 ttl=251 time=600 ms (same route)
^?
--- aix ping statistics ---
4 packets transmitted, 3 packets received, 25% packet loss
round-trip min/avg/max = 600/620/650 ms
```

我们已经在主机 bsd1 上运行过这个例子。现在选择 slip 来运行它, 观察 RR 清单中所有的 9 个 IP 地址。

在输出中令人感到疑惑的是, 为什么传出的数据报 (ICMP 回显请求) 直接从 netb 传到 aix, 而返回的数据报 (ICMP 回显应答) 却从 aix 开始经路由器 gateway 再到 netb? 这里看到的正是下面将要描述的 IP 选路的一个特点。数据报经过的路由如图 7-6 所示。

问题是 aix 不知道要把目的地为子网 140.252.13 的 IP 数据报发到主机 netb 上。相反, aix 在它的路由表中有一个默认项, 它指明当没有明确某个目的主机的路由时, 就把所有的数据报发往默认项指定的路由器 gateway。路由器 gateway 比子网 140.252.1 上的任何主机都具备更强的选路能力 (在这个以太网上有超过 150 台主机, 每台主机的路由表中都有一个默认项指向路由器 gateway, 这样就不用每台主机上都运行一个选路守护程序)。

这里没有应答的一个问题是为什么 gateway 不直接发送 ICMP 报文重定向到 aix (9.5 节), 以更新它的路由表? 由于某种原因 (很可能是由于数据报产生的重定向是一份 ICMP 回显请求报文), 重定向并没有产生。但是如果我们用 Telnet 登录到 aix 上的 daytime 服务器, ICMP 就会

产生重定向，因而它在 aix 上的路由表也随之更新。如果接着执行 ping 程序并带有记录路由选项，其路由显示表明数据报从 netb 到 aix，然后返回 netb，而不再经过路由器 gateway。在 9.5 节中将更详细地讨论 ICMP 重定向的问题。

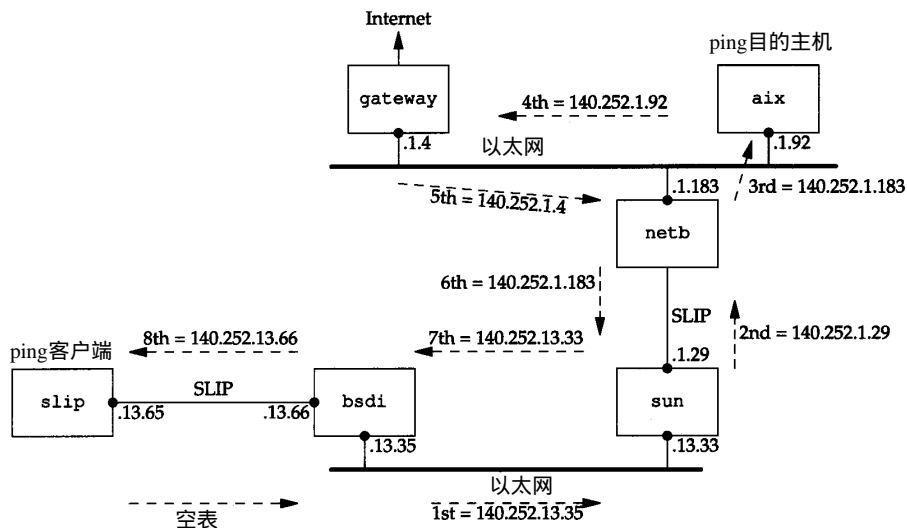


图7-6 运行带有记录路由选项的ping程序，显示IP选路的特点

7.4 IP时间戳选项

IP时间戳选项与记录路由选项类似。IP时间戳选项的格式如图 7-7 所示（请与图 7-3 进行比较）。

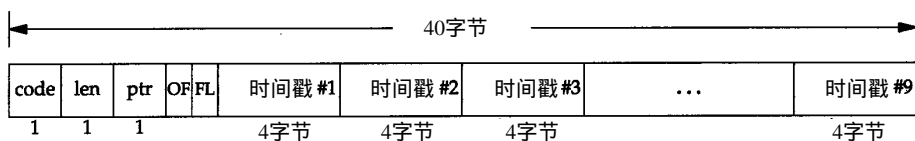


图7-7 IP首部中时间戳选项的一般格式

时间戳选项的代码为 0x44。其他两个字段 len 和 ptr 与记录路由选项相同：选项的总长度（一般为 36 或 40）和指向下一个可用空间的指针（5，9，13 等）。

接下来的两个字段是 4 bit 的值：OF 表示溢出字段，FL 表示标志字段。时间戳选项的操作根据标志字段来进行，如图 7-8 所示。

标 志	描 述
0	只记录时间戳，正如我们在图 7-7 看到的那样
1	每台路由器都记录它的 IP 地址和时间戳。在选项列表中只有存放 4 对地址和时间戳的空间
3	发送端对选项列表进行初始化，存放了 4 个 IP 地址和 4 个取值为 0 的时间戳值。只有当列表中的下一个 IP 地址与当前路由器地址相匹配时，才记录它的时间戳

图7-8 时间戳选项不同标志字段值的意义

如果路由器由于没有空间而不能增加时间戳选项，那么它将增加溢出字段的值。

时间戳的取值一般为自 UTC 午夜开始计的毫秒数, 与 ICMP 时间戳请求和应答相类似。如果路由器不使用这种格式, 它就可以插入任何它使用的时间表示格式, 但是必须打开时间戳中的高位以表明为非标准值。

与我们遇到的记录路由选项所受到的限制相比, 时间戳选项遇到情况要更坏一些。如果我们要同时记录 IP 地址和时间戳 (标志位为 1), 那么就可以同时存入其中的四对值。只记录时间戳是没有用处的, 因为我们没有标明时间戳与路由器之间的对应关系 (除非有一个永远不变的拓扑结构)。标志值取 3 会更好一些, 因为我们可以插入时间戳的路由器。一个更为基本的问题是, 很可能无法控制任何给定路由器上时间戳的正确性。这使得试图用 IP 选项来计算路由器之间的跳站数是徒劳的。我们将看到 (第 8 章) traceroute 程序可以提供一种更好的方法来计算路由器之间的跳站数。

7.5 小结

ping 程序是对两个 TCP/IP 系统连通性进行测试的基本工具。它只利用 ICMP 回显请求和回显应答报文, 而不用经过传输层 (TCP/UDP)。Ping 服务器一般在内核中实现 ICMP 的功能。

我们分析了在 LAN、WAN 以及 SLIP 链路 (拨号和线路) 上运行 ping 程序的输出结果, 并对串行线路上的 SLIP 链路吞吐量进行了计算。我们还讨论并使用了 ping 程序的 IP 记录路由选项。利用该 IP 选项, 可以看到它是如何频繁使用默认路由的。在第 9 章我们将再次回到这个讨论主题。另外, 还讨论了 IP 时间戳选项, 但它在实际使用时有所限制。

习题

- 7.1 请画出 7.2 节中 ping 输出的时间线。
- 7.2 若把 bsd1 和 slip 主机之间的 SLIP 链路设置为 9600 b/s, 请计算这时的 RTT。假定默认的数据是 56 字节。
- 7.3 当前 BSD 版中的 ping 程序允许我们为 ICMP 报文的数据部分指定一种模式 (数据部分的前 8 个字节不用来存放模式, 因为它要存放发送报文的时间)。如果我们指定的模式为 0xc0, 请重新计算上一题中的答案 (提示: 阅读 2.4 节)。
- 7.4 使用压缩 SLIP (CSLIP, 见 2.5 节) 是否会影响我们在 7.2 节中看到的 ping 输出中的时间值?
- 7.5 在图 2-4 中, ping 环回地址与 ping 主机以太网地址会出现什么不同?