7

Ping Program

7.1 Introduction

The name "ping" is taken from the sonar operation to locate objects. The Ping program was written by Mike Muuss and it tests whether another host is reachable. The program sends an ICMP echo request message to a host, expecting an ICMP echo reply to be returned. (Figure 6.3 lists all the ICMP message types.)

Normally if you can't Ping a host, you won't be able to Telnet or FTP to that host. Conversely, if you can't Telnet to a host, Ping is often the starting point to determine what the problem is. Ping also measures the round-trip time to the host, giving us some indication of how "far away" that host is.

In this chapter we'll use Ping as a diagnostic tool and to further explore ICMP. Ping also gives us an opportunity to examine the IP record route and timestamp options. Chapter 11 of [Stevens 1990] provides the source code for the Ping program.

> Years ago we could make the unqualified statement that if we can't Ping a host, we can't Telnet or FTP to that host. With the increased awareness of security on the Internet, routers that provide access control lists, and firewall gateways, unqualified statements like this are no longer true. Reachability of a given host may depend not only on reachability at the IP layer, but also on what protocol is being used, and the port numbers involved. Ping may show a host as being unreachable, yet we might be able to Telnet to port 25 (the mail server).

7.2 Ping Program

We call the `ping` program that sends the echo requests the *client*, and the host being pinged the *server*. Most TCP/IP implementations support the Ping server directly in the kernel—the server is not a user process. (The two ICMP query services that we described in Chapter 6, the address mask and timestamp requests, are also handled directly by the kernel.)

Figure 7.1 shows the ICMP echo request and echo reply messages.

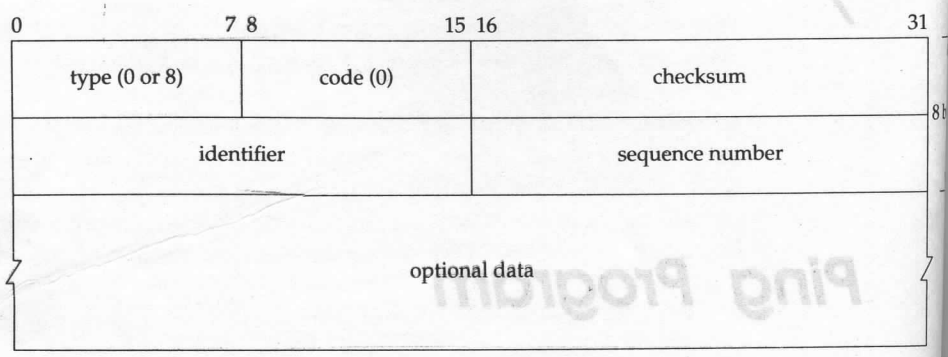

Figure 7.1 Format of ICMP message for echo request and echo reply.

As with other ICMP query messages, the server must echo the *identifier* and *sequence number* fields. Also, any optional data sent by the client must be echoed. These are presumably of interest to the client.

Unix implementations of `ping` set the *identifier* field in the ICMP message to the process ID of the sending process. This allows `ping` to identify the returned response if there are multiple instances of `ping` running at the same time on the same host.

The *sequence number* starts at 0 and is incremented every time a new echo request is sent. `ping` prints the sequence number of each returned packet, allowing us to see if packets are missing, reordered, or duplicated. IP is a best effort datagram delivery service, so any of these three conditions can occur.

Historically the `ping` program has operated in a mode where it sends an echo request once a second, printing each echo reply that is returned. Newer implementations, however, require the `-s` option to operate this way. By default, these newer implementations send only a single echo request and output "*host* is alive" if an echo reply is received, or "no answer" if no reply is received within 20 seconds.

LAN Output

`ping` output on a LAN normally looks like the following:

```
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
^?                                    type interrupt key to stop
--- svr4 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 0/0/0 ms
```

When the ICMP echo reply is returned, the sequence number is printed, followed by the TTL, and the round-trip time is calculated. (TTL is the time-to-live field in the IP header. The current BSD `ping` program prints the received TTL each time an echo reply is received—some implementations don't do this. We examine the usage of the TTL in Chapter 8 with the `traceroute` program.)

As we can see from the output above, the echo replies were returned in the order sent (0, 1, 2, and so on).

`ping` is able to calculate the round-trip time by storing the time at which it sends the echo request in the data portion of the ICMP message. When the reply is returned it subtracts this value from the current time. Notice that on the sending system, `bsdi`, the round-trip times are all calculated as 0 ms. This is because of the low-resolution timer available to the program. The BSD/386 Version 0.9.4 system only provides a 10-ms timer. (We talk more about this in Appendix B.) We'll see later that when looking at the `tcpdump` output from this `ping` example on a system with a finer resolution clock (the Sun) the time difference between the ICMP echo request and its echo reply is just under 4 ms.

The first line of output contains the IP address of the destination host, even though we specified its name (`svr4`). This implies that the name has been converted to the IP address by a resolver. We examine resolvers and the DNS in Chapter 14. For now realize that if we type a `ping` command, and a few seconds pass before the first line of output with the IP address is printed, this is the time required for the DNS to determine the IP address corresponding to the hostname.

Figure 7.2 shows the `tcpdump` output for this example.

```
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
```

Figure 7.2 `ping` output across a LAN.

The time between sending the echo request and receiving the echo reply is always 3.7 ms. We can also see that echo requests are sent approximately 1 second apart.