9

IP Routing

9.1 Introduction

Routing is one of the most important functions of IP. Figure 9.1 shows a simplified view of the processing done at the IP layer. Datagrams to be routed can be generated either on the local host or on some other host. In the latter case this host must be configured to act as a router, or datagrams received through the network interfaces that are not ours are dropped (i.e., silently discarded).

In Figure 9.1 we also show a routing daemon, which is normally a user process. The most common daemons used on Unix systems are the programs `routed` and `gated`. (The term *daemon* means the process is running "in the background," carrying out operations on behalf of the whole system. Daemons are normally started when the system is bootstrapped and run as long as the system is up.) The topics of which routing protocol to use on a given host, how to exchange routing information with adjacent routers, and how the routing protocols work are complex and can fill an entire book of their own. (Interested readers are referred to [Perlman 1992] for many of the details.) We'll look briefly at dynamic routing and the Routing Information Protocol (RIP) in Chapter 10. Our main interest in the current chapter is how a single IP layer makes its routing decisions.

The routing table that we show in Figure 9.1 is accessed frequently by IP (on a busy host this could mean hundreds of times a second) but is updated much less frequently by a routing daemon (possibly about once every 30 seconds). The routing table can also be updated when ICMP "redirect" messages are received, something we'll look at in Section 9.5, and by the `route` command. This command is often executed when the system is bootstrapped, to install some initial routes. We'll also use the `netstat` command in this chapter to display the routing table.

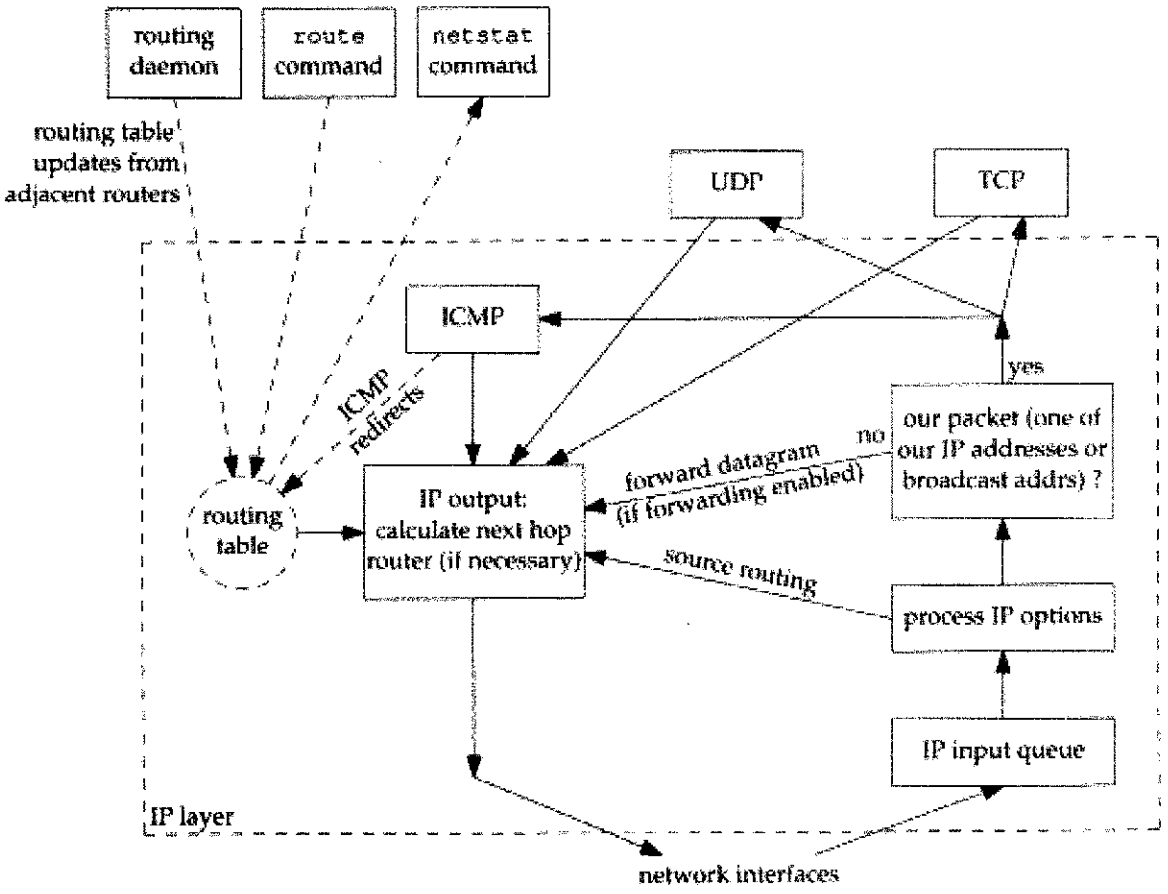

Figure 9.1 Processing done at the IP layer.

9.2 Routing Principles

The place to start our discussion of IP routing is to understand what is maintained by the kernel in its routing table. The information contained in the routing table drives all the routing decisions made by IP.

In Section 3.3 we listed the steps that IP performs when it searches its routing table.

1. Search for a matching host address.
2. Search for a matching network address.
3. Search for a default entry. (The default entry is normally specified in the routing table as a network entry, with a network ID of 0.)

A matching host address is always used before a matching network address.

The routing done by IP, when it searches the routing table and decides which interface to send a packet out, is a *routing mechanism*. This differs from a *routing policy*, which is a set of rules that decides which routes go into the routing table. IP performs the routing mechanism while a routing daemon normally provides the routing policy.

Simple Routing Table

Let's start by looking at some typical host routing tables. On the host `svr4` we execute the `netstat` command with the `-r` option to list the routing table and the `-n` option, which prints IP addresses in numeric format, rather than as names. (We do this because some of the entries in the routing table are for networks, not hosts. Without the `-n` option, the `netstat` command searches the file `/etc/networks` for the network names. This confuses the discussion by adding another set of names—network names in addition to hostnames.)

```
svr4 % netstat -rn
Routing tables
Destination       Gateway           Flags    Refcnt  Use      Interface
140.252.13.65     140.252.13.35     UGH      0       0        emd0
127.0.0.1         127.0.0.1         UH       1       0        lo0
default           140.252.13.33     UG       0       0        emd0
140.252.13.32     140.252.13.34     U        4       25043    emd0
```

The first line says for destination 140.252.13.65 (host `slip`) the gateway (router) to send the packet to is 140.252.13.35 (`bsdi`). This is what we expect, since the host `slip` is connected to `bsdi` with a SLIP link, and `bsdi` is on the same Ethernet as this host.

There are five different flags that can be printed for a given route.

- `U` The route is up.
- `G` The route is to a gateway (router). If this flag is not set, the destination is directly connected.
- `H` The route is to a host, that is, the destination is a complete host address. If this flag is not set, the route is to a network, and the destination is a network address: a net ID, or a combination of a net ID and a subnet ID.
- `D` The route was created by a redirect (Section 9.5).
- `M` The route was modified by a redirect (Section 9.5).

The `G` flag is important because it differentiates between an *indirect route* and a *direct route*. (The `G` flag is not set for a direct route.) The difference is that a packet going out a direct route has both the IP address and the link-layer address specifying the destination (Figure 3.3, p. 40). When a packet is sent out an indirect route, the IP address specifies the final destination but the link-layer address specifies the gateway (that is, the next-hop router). We saw an example of this in Figure 3.4 (p. 41). In this routing table example we have an indirect route (the `G` flag is set) so the IP address of a packet using this route is the final destination (140.252.13.65), but the link-layer address must correspond to the router 140.252.13.35.

It's important to understand the difference between the `G` and `H` flags. The `G` flag differentiates between a direct and an indirect route, as described above. The `H` flag, however, specifies that the destination address (the first column of `netstat` output) is a complete host address. The absence of the `H` flag means the destination address is a network address (the host ID portion will be 0). When the routing table is searched for

a route to a destination IP address, a host address entry must match the destination address completely, while a network address only needs to match the network ID and any subnet ID of the destination address. Also, some versions of the `netstat` command print all the host entries first, followed by the network entries.

The reference count column gives the number of active uses for each route. A connection-oriented protocol such as TCP holds on to a route while the connection is established. If we established a Telnet connection between the two hosts `svr4` and `slip`, we would see the reference count go to 1. With another Telnet connection the reference count would go to 2, and so on.

The next column ("use") displays the number of packets sent through that route. If we are the only users of the route and we run the `ping` program to send 5 packets, the count goes up by 5. The final column, the interface, is the name of the local interface.

The second line of output is for the loopback interface (Section 2.7), always named `lo0`. The `G` flag is not set, since the route is not to a gateway. The `H` flag indicates that the destination address (127.0.0.1) is a host address, and not a network address. When the `G` field is not set, indicating a direct route, the gateway column gives the IP address of the outgoing interface.

The third line of output is for the default route. Every host can have one or more default routes. This entry says to send packets to the router 140.252.13.33 (`sun`) if a more specific route can't be found. This means the current host (`svr4`) can access other systems across the Internet through the router `sun` (and its SLIP link), using this single routing table entry. Being able to establish a default route is a powerful concept. The flags for this route (`UG`) indicate that it's a route to a gateway, as we expect.

> Here we purposely call `sun` a router and not a host because when it's used as a default router, its IP forwarding function is being used, not its host functionality.
>
> The Host Requirements RFC specifically states that the IP layer must support multiple default routes. Many implementations, however, don't support this. When multiple default routes exist, a common technique is to round robin among them. This is what Solaris 2.2 does, for example.

The final line of output is for the attached Ethernet. The `H` flag is not set, indicating that the destination address (140.252.13.32) is a network address with the host portion set to 0. Indeed, the low-order 5 bits are 0 (Figure 3.11, p. 47). Since this is a direct route (the `G` flag is not set) the gateway column specifies the IP address of the outgoing interface.

Implied in this final entry, but not shown by the `netstat` output, is the mask associated with this destination address (140.252.13.32). If this destination is being compared against the IP address 140.252.13.33, the address is first logically ANDed with the mask associated with the destination (the subnet mask of the interface, `0xffffffe0`, from Section 3.7) before the comparison. For a network route to a directly connected network, the routing table mask defaults to the subnet mask of the interface. But in general the routing table mask can assume any 32-bit value. A value other than the default can be specified as an option to the `route` command.

The complexity of a host's routing table depends on the topology of the networks to which the host has access.

1. The simplest (but least interesting) case is a host that is not connected to any networks at all. The TCP/IP protocols can still be used on the host, but only to communicate with itself! The routing table in this case consists of a single entry for the loopback interface.
2. Next is a host connected to a single LAN, only able to access hosts on that LAN. The routing table consists of two entries: one for the loopback interface and one for the LAN (such as an Ethernet).
3. The next step occurs when other networks (such as the Internet) are reachable through a single router. This is normally handled with a default entry pointing to that router.
4. The final step is when other host-specific or network-specific routes are added. In our example the route to the host `slip`, through the router `bsdi`, is an example of this.

Let's follow through the steps IP performs when using this routing table to route some example packets on the host `svr4`.

1. Assume the destination address is the host `sun`, 140.252.13.33. A search is first made for a matching host entry. The two host entries in the table (`slip` and `localhost`) don't match, so a search is made through the routing table again for a matching network address. A match is found with the entry 140.252.13.32 (the network IDs and subnet IDs match), so the `emd0` interface is used. This is a direct route, so the link-layer address will be the destination address.
2. Assume the destination address is the host `slip`, 140.252.13.65. The first search through the table, for a matching host address, finds a match. This is an indirect route so the destination IP address remains 140.252.13.65, but the link-layer address must be the link-layer address of the gateway 140.252.13.35, and the interface is `emd0`.
3. This time we're sending a datagram across the Internet to the host `aw.com` (192.207.117.2). The first search of the routing table for a matching host address fails, as does the second search for a matching network address. The final step is a search for a default entry, and this succeeds. The route is an indirect route through the gateway 140.252.13.33 using the interface `emd0`.
4. In our final example we send a datagram to our own host. There are four ways to do this, using either the hostname, the host IP address, the loopback name, or the loopback IP address:

```
ftp svr4
ftp 140.252.13.34

ftp localhost
ftp 127.0.0.1
```

In the first two cases, the second search of the routing table yields a network match with 140.252.13.32, and the packet is sent down to the Ethernet driver. As we showed in Figure 2.4 (p. 28) it will be seen that this packet is destined for the host's own IP address, and the packet is sent to the loopback driver, which sends it to the IP input queue.

In the latter two cases, specifying the name of the loopback interface or its IP address, the first search of the routing table finds the matching host address entry, and the packet is sent to the loopback driver, which sends it to the IP input queue.

In all four cases the packet is sent to the loopback driver, but two different routing decisions are made.

Initializing a Routing Table

We never said how these routing table entries are created. Whenever an interface is initialized (normally when the interface's address is set by the `ifconfig` command) a direct route is automatically created for that interface. For point-to-point links and the loopback interface, the route is to a host (i.e., the `H` flag is set). For broadcast interfaces such as an Ethernet, the route is to that network.

Routes to hosts or networks that are not directly connected must be entered into the routing table somehow. One common way is to execute the `route` command explicitly from the initialization files when the system is bootstrapped. On the host `svr4` the following two commands were executed to add the entries that we showed earlier:

```
route add default sun 1
route add slip bsdi 1
```

The third arguments (`default` and `slip`) are the destinations, the fourth argument is the gateway (router), and the final argument is a routing metric. All that the `route` command does with this metric is install the route with the `G` flag set if the metric is greater than 0, or without the `G` flag if the metric is 0.

> Unfortunately, few systems agree on which start-up file contains the `route` commands. Under 4.4BSD and BSD/386 it is `/etc/netstart`, under SVR4 it is `/etc/inet/rc.inet`, under Solaris 2.x it is `/etc/rc2.d/S69inet`, SunOS 4.1.x uses `/etc/rc.local`, and AIX 3.2.2 uses `/etc/rc.net`.

Some systems allow a default router to be specified in a file such as `/etc/defaultrouter`, and this default is added to the routing table on every reboot.

Other ways to initialize a routing table are to run a routing daemon (Chapter 10) or to use the newer router discovery protocol (Section 9.6).

A More Complex Routing Table

The host `sun` is the default router for all the hosts on our subnet, since it has the dialup SLIP link that connects to the Internet (see the figure on the inside front cover).

```
sun % netstat -rn
Routing tables
Destination       Gateway           Flags    Refcnt Use      Interface
140.252.13.65     140.252.13.35     UGH      0      171      le0
127.0.0.1         127.0.0.1         UH       1      766      lo0
140.252.1.183     140.252.1.29      UH       0      0        sl0
default           140.252.1.183     UG       1      2955     sl0
140.252.13.32     140.252.13.33     U        8      99551    le0
```

The first two entries are identical to the first two for the host `svr4`: a host-specific route to `slip` through the router `bsdi`, and the loopback route.

The third line is new. It is a direct route (the G flag is not set) to a host (the H flag is set) and corresponds to our point-to-point link, the SLIP interface. If we compare it to the output from the `ifconfig` command,

```
sun % ifconfig sl0
sl0: flags=1051<UP,POINTOPOINT,RUNNING>
        inet 140.252.1.29 --> 140.252.1.183 netmask ffffff00
```

we see that the destination address in the routing table is the other end of the point-to-point link (the router `netb`) and the gateway address is really the local IP address of the outgoing interface (140.252.1.29). (We said earlier that the gateway address printed by `netstat` for a direct route is the local IP address of the interface to use.)

The default entry is an indirect route (G flag) to a network (no H flag), as we expect. The gateway address is the address of the router (140.252.1.183, the other end of the SLIP link) and not the local IP address of the SLIP link (140.252.1.29). Again, this is because it is an indirect route, not a direct route.

We should also note that the third and fourth lines output by `netstat` (the ones with an interface of `sl0`) are created by the SLIP software being used when the SLIP line is brought up, and deleted when the SLIP link is brought down.

No Route to Destination

All our examples so far have assumed that the search of the routing table finds a match, even if the match is the default route. What if there is no default route, and a match isn't found for a given destination?

The answer depends on whether the IP datagram being routed was generated on the host or is being forwarded (e.g., we're acting as a router). If the datagram was generated on this host, an error is returned to the application that sent the datagram, either "host unreachable" or "network unreachable." If the datagram was being forwarded, an ICMP host unreachable error is sent back to original sender. We examine this error in the following section.

9.3 ICMP Host and Network Unreachable Errors

The ICMP "host unreachable" error message is sent by a router when it receives an IP datagram that it cannot deliver or forward. (Figure 6.10 shows the format of the ICMP

unreachable messages.) We can see this easily on our network by taking down the dialup SLIP link on the router sun, and trying to send a packet through the SLIP link from any of the other hosts that specify sun as the default router.

> Older implementations of the BSD TCP/IP software generated either a host unreachable, or a network unreachable, depending on whether the destination was on a local subnet or not. 4.4BSD generates only the host unreachable.

Recall from the `netstat` output for the router sun shown in the previous section that the routing table entries that use the SLIP link are added when the SLIP link is brought up, and deleted when the SLIP link is brought down. This means that when the SLIP link is down, there is no default route on sun. But we don't try to change all the other host's routing tables on our small network, having them also remove their default route. Instead we count on the ICMP host unreachable generated by sun for any packets that it gets that it cannot forward.

We can see this by running `ping` on `svr4`, for a host on the other side of the dialup SLIP link (which is down):

```
svr4 % ping gemini
ICMP Host Unreachable from gateway sun (140.252.13.33)
ICMP Host Unreachable from gateway sun (140.252.13.33)
^?                                        type interrupt key to stop
```

Figure 9.2 shows the `tcpdump` output for this example, run on the host `bsdi`.

```
1  0.0                  svr4 > gemini: icmp: echo request
2  0.00 (0.00)          sun > svr4: icmp: host gemini unreachable
3  0.99 (0.99)          svr4 > gemini: icmp: echo request
4  0.99 (0.00)          sun > svr4: icmp: host gemini unreachable
```

Figure 9.2 ICMP host unreachable in response to `ping`.

When the router sun finds no route to the host `gemini`, it responds to the echo request with a host unreachable.

If we bring the SLIP link to the Internet up, and try to ping an IP address that is not connected to the Internet, we expect an error. What is interesting is to see how far the packet gets into the Internet, before the error is returned:

```
sun % ping 192.82.148.1              this IP address is not connected to the Internet
PING 192.82.148.1: 56 data bytes
ICMP Host Unreachable from gateway enss142.UT.westnet.net (192.31.39.21)
 for icmp from sun (140.252.1.29) to 192.82.148.1
```

Looking at Figure 8.5 (p. 103) we see that the packet made it through six routers before detecting that the IP address was invalid. Only when it got to the border of the NSFNET backbone was the error detected. This implies that the six routers that forwarded the packet were doing so because of default entries, and only when it reached the NSFNET backbone did a router have complete knowledge of every network connected to the Internet. This illustrates that many routers can operate with just partial knowledge of the big picture.

[Ford, Rekhter, and Braun 1993] define a *top-level routing domain* as one that maintains routing information to most Internet sites and does not use default routes. They note that five of these top-level routing domains exist on the Internet: the NSFNET backbone, the Commercial Internet Exchange (CIX), the NASA Science Internet (NSI), SprintLink, and the European IP Backbone (EBONE).

9.4 To Forward or Not to Forward

We've mentioned a few times that hosts are not supposed to forward IP datagrams unless they have been specifically configured as a router. How is this configuration done?

Most Berkeley-derived implementations have a kernel variable named `ipforwarding`, or some similar name. (See Appendix E.) Some systems (BSD/386 and SVR4, for example) only forward datagrams if this variable is nonzero. SunOS 4.1.x allows three values for the variable: –1 means never forward and never change the value of the variable, 0 means don't forward by default but set this variable to 1 when two or more interfaces are up, and 1 means always forward. Solaris 2.x changes the three values to be 0 (never forward), 1 (always forward), and 2 (only forward when two or more interfaces are up).

Older 4.2BSD hosts forwarded datagrams by default, which caused lots of problems for systems configured improperly. That's why this kernel option must always default to "never forward" unless the system administrator specifically enables forwarding.

9.5 ICMP Redirect Errors

The ICMP redirect error is sent by a router to the sender of an IP datagram when the datagram should have been sent to a different router. The concept is simple, as we show in the three steps in Figure 9.3. The only time we'll see an ICMP redirect is when the host has a choice of routers to send the packet to. (Recall the earlier example of this we saw in Figure 7.6, p. 94.)

1. We assume that the host sends an IP datagram to R1. This routing decision is often made because R1 is the default router for the host.
2. R1 receives the datagram and performs a lookup in its routing table and determines that R2 is the correct next-hop router to send the datagram to. When it sends the datagram to R2, R1 detects that it is sending it out the same interface on which the datagram arrived (the LAN to which the host and the two routers are attached). This is the clue to a router that a redirect can be sent to the original sender.
3. R1 sends an ICMP redirect to the host, telling it to send future datagrams to that destination to R2, instead of R1.

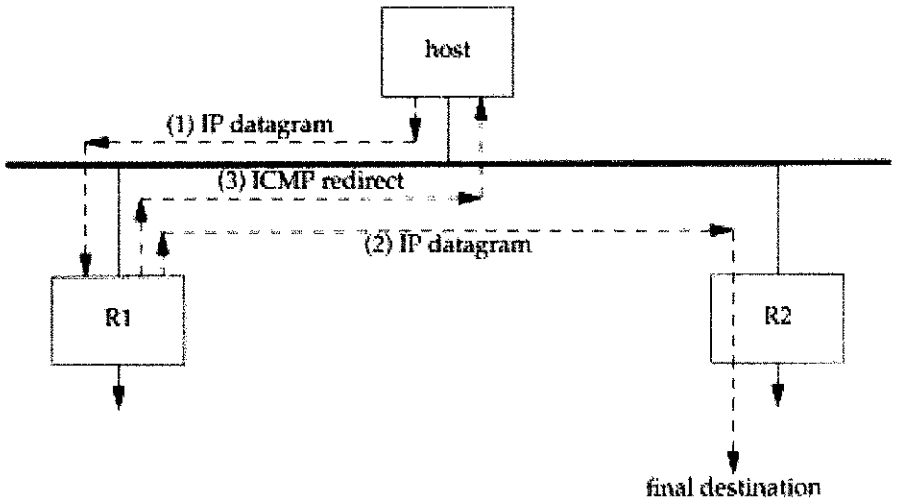

Figure 9.3 Example of an ICMP redirect.

A common use for redirects is to let a host with minimal routing knowledge build up a better routing table over time. The host can start with only a default route (either R1 or R2 from our example in Figure 9.3) and anytime this default turns out to be wrong, it'll be informed by that default router with a redirect, allowing the host to update its routing table accordingly. ICMP redirects allow TCP/IP hosts to be dumb when it comes to routing, with all the intelligence in the routers. Obviously R1 and R2 in our example have to know more about the topology of the attached networks, but all the hosts attached to the LAN can start with a default route and learn more as they receive redirects.

An Example

We can see ICMP redirects in action on our network (inside front cover). Although we show only three hosts (`aix`, `solaris`, and `gemini`) and two routers (`gateway` and `netb`) on the top network, there are more than 150 hosts and 10 other routers on this network. Most of the hosts specify `gateway` as the default router, since it provides access to the Internet.

How is the author's subnet (the bottom four hosts in the figure) accessed from the hosts on the 140.252.1 subnet? First recall that if only a single host is at the end of the SLIP link, proxy ARP is used (Section 4.6). This means nothing special is required for hosts on the top network (140.252.1) to access the host `sun` (140.252.1.29). The proxy ARP software in `netb` handles this.

When a network is at the other end of the SLIP link, however, routing becomes involved. One solution is for every host and router to know that the router `netb` is the gateway for the network 140.252.13. This could be done by either a static route in each host's routing table, or by running a routing daemon in each host. A simpler way (and the method actually used) is to utilize ICMP redirects.

Let's run the `ping` program from the host `solaris` on the top network to the host `bsdi` (140.252.13.35) on the bottom network. Since the subnet IDs are different, proxy ARP can't be used. Assuming a static route has not been installed, the first packet sent will use the default route to the router `gateway`. Here is the routing table before we run `ping`:

```
solaris % netstat -rn
Routing Table:
  Destination        Gateway           Flags  Ref   Use   Interface
-------------------- ----------------- -----  ----- ------ ---------
127.0.0.1            127.0.0.1         UH         0    848  lo0
140.252.1.0          140.252.1.32      U          3  15042  le0
224.0.0.0            140.252.1.32      U          3      0  le0
default              140.252.1.4       UG         0   5747
```

(The entry for 224.0.0.0 is for IP multicasting. We describe it in Chapter 12.) If we specify the `-v` option to `ping`, we'll see any ICMP messages received by the host. We need to specify this to see the redirect message that's sent.

```
solaris % ping -sv bsdi
PING bsdi: 56 data bytes
ICMP Host redirect from gateway gateway (140.252.1.4)
 to netb (140.252.1.183) for bsdi (140.252.13.35)
64 bytes from bsdi (140.252.13.35): icmp_seq=0. time=383. ms
64 bytes from bsdi (140.252.13.35): icmp_seq=1. time=364. ms
64 bytes from bsdi (140.252.13.35): icmp_seq=2. time=353. ms
^?                                        type interrupt key to stop
----bsdi PING Statistics----
4 packets transmitted, 3 packets received, 25% packet loss
round-trip (ms)  min/avg/max = 353/366/383
```

Before we receive the first `ping` response, the host receives an ICMP redirect from the default router `gateway`. If we then look at the routing table, we'll see that the new route to the host `bsdi` has been inserted. (This new entry is shown in a bolder font.)

```
solaris % netstat -rn
Routing Table:
  Destination        Gateway           Flags  Ref   Use   Interface
-------------------- ----------------- -----  ----- ------ ---------
127.0.0.1            127.0.0.1         UH         0    848  lo0
140.252.13.35        140.252.1.183     UGHD       0      2
140.252.1.0          140.252.1.32      U          3  15045  le0
224.0.0.0            140.252.1.32      U          3      0  le0
default              140.252.1.4       UG         0   5749
```

This is the first time we've seen the `D` flag, which means the route was installed by an ICMP redirect. The G flag means it's an indirect route to a gateway (`netb`), and the `H` flag means it's a host route (as we expect), not a network route.

Since this is a host route, added by a host redirect, it handles only the host `bsdi`. If we then access the host `svr4`, another redirect is generated, creating another host route. Similarly, accessing the host `slip` creates another host route. The point here is that each redirect is for a single host, causing a host route to be added. All three hosts on the author's subnet (`bsdi`, `svr4`, and `slip`) could also be handled by a single network

route pointing to the router sun. But ICMP redirects create host routes, not network routes, because the router generating the redirect in this example (`gateway`) has no knowledge of the subnet structure on the 140.252.13 network.

More Details

Figure 9.4 shows the format of the ICMP redirect message.

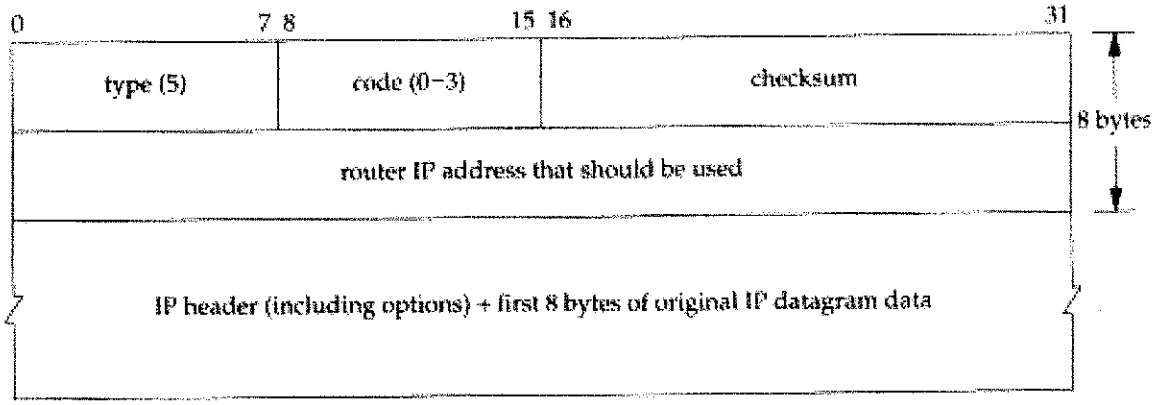

Figure 9.4 ICMP redirect message.

There are four different redirect messages, with different *code* values, as shown in Figure 9.5.

code	Description
0	redirect for network
1	redirect for host
2	redirect for type-of-service and network
3	redirect for type-of-service and host

Figure 9.5 Different *code* values for ICMP redirect.

There are three IP addresses that the receiver of an ICMP redirect must look at: (1) the IP address that caused the redirect (which is in the IP header returned as the data portion of the ICMP redirect), (2) the IP address of the router that sent the redirect (which is the source IP address of the IP datagram containing the redirect), and (3) the IP address of the router that should be used (which is in bytes 4–7 of the ICMP message).

There are numerous rules about ICMP redirects. First, redirects are generated only by routers, not by hosts. Also, redirects are intended to be used by hosts, not routers. It is assumed that routers participate in a routing protocol with other routers, and the routing protocol should obviate the need for redirects. (This means that in Figure 9.1 the routing table should be updated by either a routing daemon or redirects, but not by both.)

4.4BSD, when acting as a router, performs the following checks, all of which must be true before an ICMP redirect is generated.

1. The outgoing interface must equal the incoming interface.
2. The route being used for the outgoing datagram must not have been created or modified by an ICMP redirect, and must not be the router's default route.
3. The datagram must not be source routed.
4. The kernel must be configured to send redirects.

> The kernel variable is named `ip_sendredirects`, or something similar. (See Appendix E.) Most current systems (4.4BSD, SunOS 4.1.x, Solaris 2.x, and AIX 3.2.2, for example) enable this variable by default. Other systems such as SVR4 disable it by default.

Additionally, a 4.4BSD host that receives an ICMP redirect performs some checks before modifying its routing table. These are to prevent a misbehaving router or host, or a malicious user, from incorrectly modifying a system's routing table.

1. The new router must be on a directly connected network.
2. The redirect must be from the current router for that destination.
3. The redirect cannot tell the host to use itself as the router.
4. The route that's being modified must be an indirect route.

Our final point about redirects is that routers should send only host redirects (*codes* 1 or 3 from Figure 9.5) and not network redirects. Subnetting makes it hard to specify exactly when a network redirect can be sent instead of a host redirect. Some hosts treat a received network redirect as a host redirect, in case a router sends the wrong type.

9.6 ICMP Router Discovery Messages

We mentioned earlier in this chapter that one way to initialize a routing table is with static routes specified in configuration files. This is often used to set a default entry. A newer way is to use the ICMP router advertisement and solicitation messages.

The general concept is that after bootstrapping, a host broadcasts or multicasts a router solicitation message. One or more routers respond with a router advertisement message. Additionally, the routers periodically broadcast or multicast their router advertisements, allowing any hosts that are listening to update their routing table accordingly.

RFC 1256 [Deering 1991] specifies the format of these two ICMP messages. Figure 9.6 shows the format of the ICMP router solicitation message. Figure 9.7 shows the format of the ICMP router advertisement message sent by routers.

Multiple addresses can be advertised by a router in a single message. *Number of addresses* is the number. *Address entry size* is the number of 32-bit words for each router address, and is always 2. *Lifetime* is the number of seconds that the advertised addresses can be considered valid.

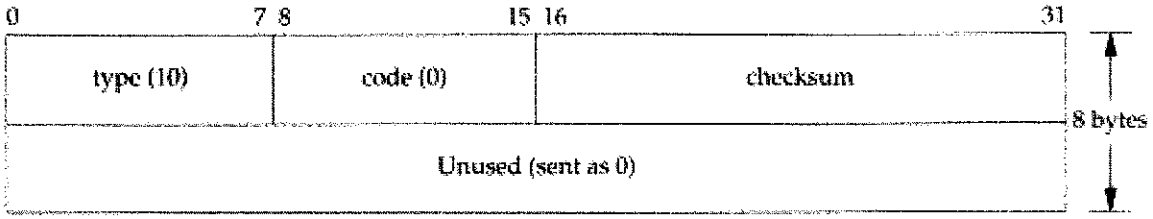

Figure 9.6 Format of ICMP router solicitation message.

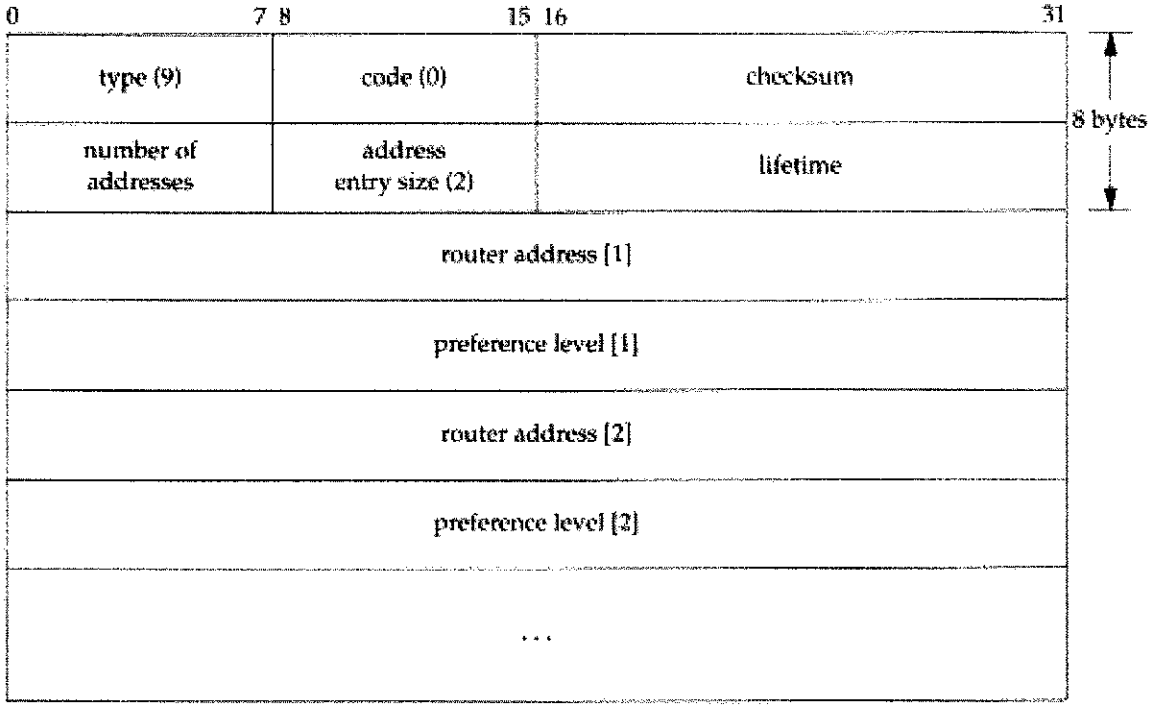

Figure 9.7 Format of ICMP router advertisement message.

One or more pairs of an IP address and a preference then follow. The IP address must be one of the sending router's IP addresses. The *preference level* is a signed 32-bit integer indicating the preference of this address as a default router address, relative to other router addresses on the same subnet. Larger values imply more preferable addresses. The preference level `0x80000000` means the corresponding address, although advertised, is not to be used by the receiver as a default router address. The default value of the preference is normally 0.

Router Operation

When a router starts up it transmits periodic advertisements on all interfaces capable of broadcasting or multicasting. These advertisements are not exactly periodic, but are

randomized, to reduce the probability of synchronization with other routers on the same subnet. The normal time interval between advertisements is between 450 and 600 seconds. The default lifetime for a given advertisement is 30 minutes.

Another use of the lifetime field occurs when an interface on a router is disabled. In that case the router can transmit a final advertisement on the interface with the lifetime set to 0.

In addition to the periodic, unsolicited advertisements, a router also listens for solicitations from hosts. It responds to these solicitations with a router advertisement.

If there are multiple routers on a given subnet, it is up to the system administrator to configure the preference level for each router as appropriate. For example, the primary default router would have a higher preference than a backup.

Host Operation

Upon bootstrap a host normally transmits three router solicitations, 3 seconds apart. As soon as a valid advertisement is received, the solicitations stop.

A host also listens for advertisements from adjacent routers. These advertisements can cause the host's default router to change. Also, if an advertisement is not received for the current default, that default can time out.

As long as the normal default router stays up, that router will send advertisements every 10 minutes, with a lifetime of 30 minutes. This means the host's default entry won't time out, even if one or two advertisements are lost.

Implementation

The router discovery messages are normally generated by and processed by a user process (a daemon). This adds yet another program updating the routing table in Figure 9.1, although it would only add or delete a default entry. The daemon would have to be configured to act as a router or a host.

These two ICMP messages are new and not supported by all systems. Solaris 2.x is the only system in our network that supports these messages (the `in.rdisc` daemon). Although the RFC recommends using IP multicasting whenever possible, router discovery can work using broadcast messages also.

9.7 Summary

The operation of IP routing is fundamental to a system running TCP/IP, be it a host or router. The routing table entries are simple: up to 5 flag bits, a destination IP address (host, network, or default), a next-hop router IP address (for an indirect route) or a local interface IP address (for a direct route), and a pointer to a local interface to use. Host entries have priority over network entries, which have priority over default entries.

A search of this routing table is made for every IP datagram that the system generates or forwards, and can be updated by either a routing daemon or ICMP redirects. By default a system should never forward a datagram unless it has specifically been

configured to do so. Static routes can be entered using the `route` command, and the newer ICMP router discovery messages can be used to initialize and dynamically update default entries. Hosts can start with a simple routing table that is updated dynamically by ICMP redirects from its default router.

Our discussion in this chapter has focused on how a single system uses its routing table. In the next chapter we examine how routers exchange routing information with each other.

Exercises

9.1 Why do you think both types of ICMP redirects—network and host—exist?

9.2 In the routing table for `svr4` shown at the beginning of Section 9.2, is a specific route to the host `slip` (140.252.13.65) necessary? What would change if this entry were removed from the routing table?

9.3 Consider a cable with both 4.2BSD hosts and 4.3BSD hosts. Assume the network ID is 140.1. The 4.2BSD hosts only recognize a host ID of all zero bits as the broadcast address (140.1.0.0), while the 4.3BSD hosts normally send a broadcast using a host ID of all one bits (140.1.255.255). Also, the 4.2BSD hosts by default will try to forward incoming datagrams, even if they have only a single interface.

Describe the events that happen when the 4.2BSD hosts receive an IP datagram with the destination address of 140.1.255.255.

9.4 Continue the previous exercise, assuming someone corrects this problem by adding an entry to the ARP cache on one system on the 140.1 subnet (using the `arp` command) saying that the IP address 140.1.255.255 has a corresponding Ethernet address of all one bits (the Ethernet broadcast). Describe what happens now.

9.5 Examine your system's routing table and describe each entry.