CHAPTER

Routing

8

What You Will Learn

In this chapter, you will learn how routing works. We'll look at both direct delivery of packets to a destination without a router and indirect delivery through a router, both of which happen all the time. Routers provide indirect delivery between LANs while bridges essentially provide direct delivery only. Packet switching, on the other hand, is a related form of indirect delivery that will be explored in a later chapter.

You will learn about the role of *routing tables* and *forwarding tables* in the routing process. Technically, routers use the information in the routing table to create a forwarding table to forward packets to the *next hop* based on a metric, but many people use the terms *routing* and *forwarding* loosely, often using one term for both. We'll try to use the terms as defined here consistently in this chapter, but there is no real formal definition of either term.

The Internet is the largest router-based network in the world. Router-based networks, as we'll see in this chapter, are characterized by certain features and methods of operation. The most obvious feature of a router-based network is that the most essential network nodes are routers and not bridges or switches or more exotic devices. This does not mean that there are no bridges, switches, and other types of network devices. It just means that routing is the most important function in moving packets from source to destination. This chapter is an introduction to routing as a process.

Figure 8.1 shows the areas of the Illustrated Network we will be investigating in this chapter. The LANs and customer-edge routers are highlighted, but the other routers play a large but unseen part in this chapter. We'll look at the role of the service-provider routers in the chapters on routing protocols. For now, we'll focus on how sending devices decide whether the destination is on their own network or whether the packets must be sent to a router for forwarding through a routing network.

We'll talk about forwarding tables in later chapters that investigate routing and routers more deeply. For now, let's take a look at the simple routing tables that are used on the Illustrated Network's hosts and routers.

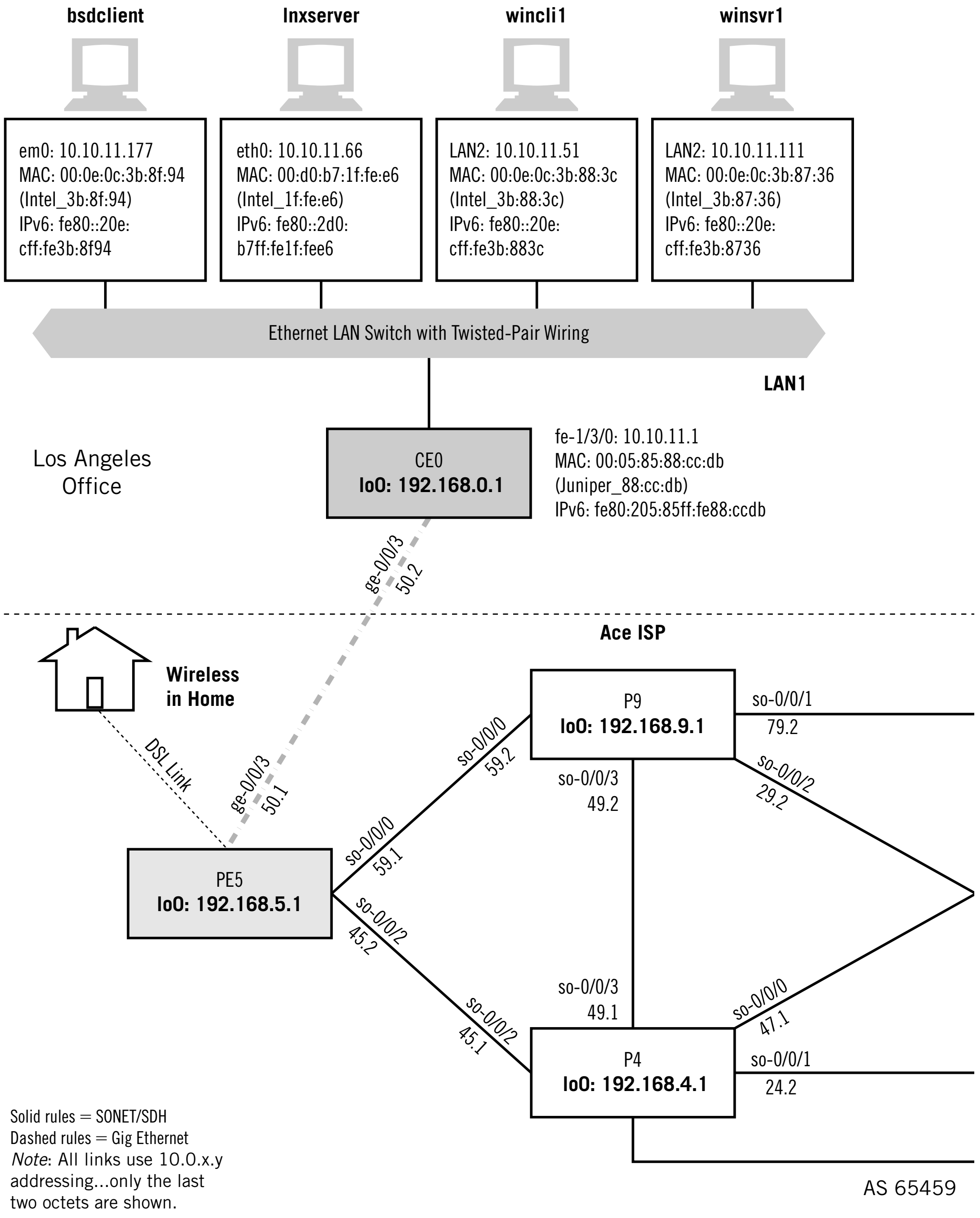
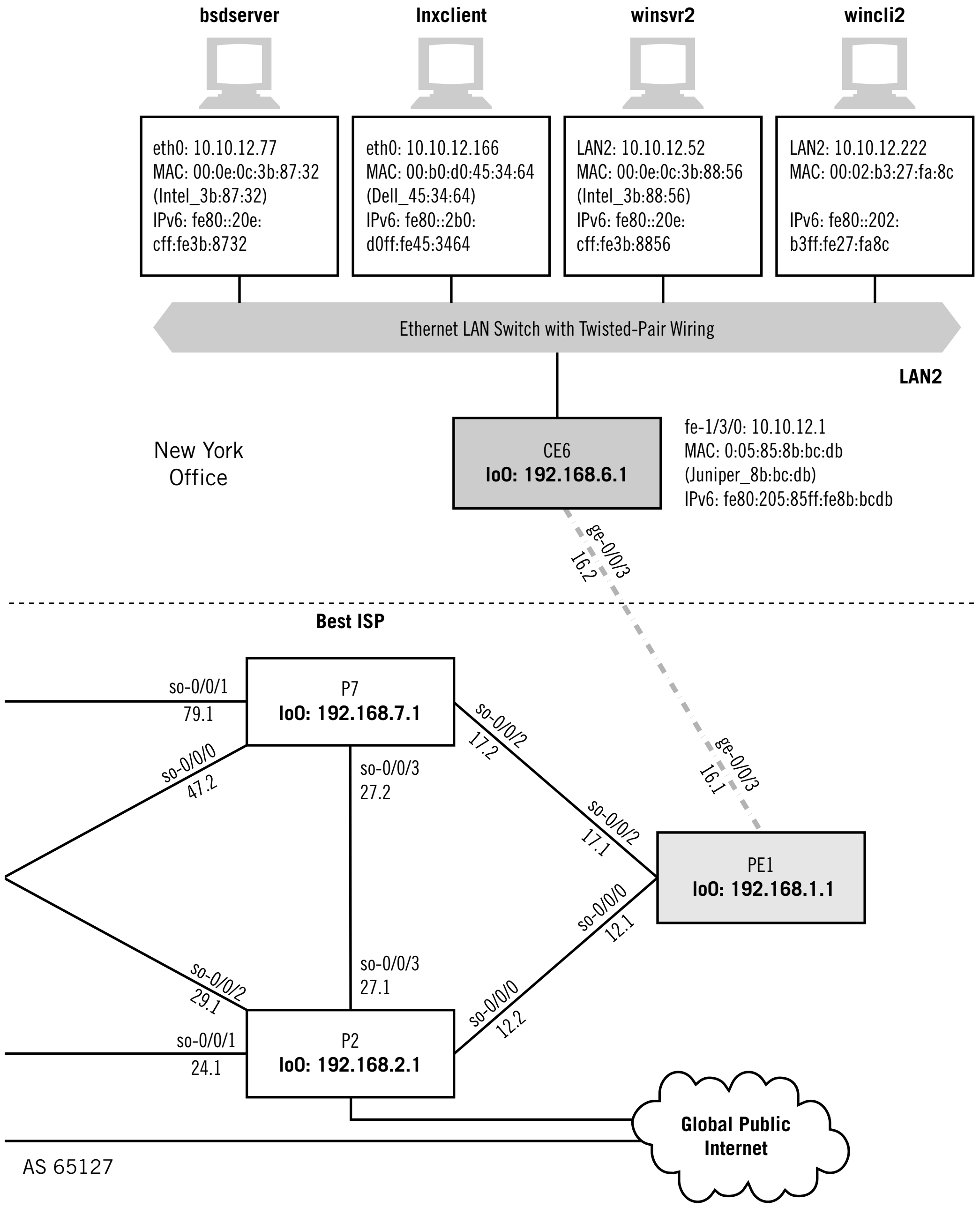


FIGURE 8.1

The Illustrated Network LAN internetworking, showing how the routers are connected and the links available to forward (route) packets through the network.


bsdserver
lnxclient
winsvr2
wincli2
eth0: 10.10.12.77
MAC: 00:0e:0c:3b:87:32
(Intel_3b:87:32)
IPv6: fe80::20e:
cff:fe3b:8732
eth0: 10.10.12.166
MAC: 00:b0:d0:45:34:64
(Dell_45:34:64)
IPv6: fe80::2b0:
d0ff:fe45:3464
LAN2: 10.10.12.52
MAC: 00:0e:0c:3b:88:56
(Intel_3b:88:56)
IPv6: fe80::20e:
cff:fe3b:8856
LAN2: 10.10.12.222
MAC: 00:02:b3:27:fa:8c
IPv6: fe80::202:
b3ff:fe27:fa8c
Ethernet LAN Switch with Twisted-Pair Wiring
LAN2
New York Office
CE6
lo0: 192.168.6.1
fe-1/3/0: 10.10.12.1
MAC: 0:05:85:8b:bc:db
(Juniper_8b:bc:db)
IPv6: fe80:205:85ff:fe8b:bcdb
ge-0/0/3
16.2
Best ISP
so-0/0/1
79.1
P7
lo0: 192.168.7.1
so-0/0/2
17.2
ge-0/0/3
16.1
so-0/0/0
47.2
so-0/0/3
27.2
so-0/0/2
17.1
PE1
lo0: 192.168.1.1
so-0/0/0
12.1
so-0/0/2
29.1
so-0/0/3
27.1
so-0/0/0
12.2
so-0/0/1
24.1
P2
lo0: 192.168.2.1
Global Public Internet
AS 65127

Routing Table and Forwarding Table

There are really two different types of network tables used in routers and hosts, and we'll distinguish them in this chapter. The routing table holds all of the information that a device knows about network addresses and interfaces, and is usually held in a fairly user-friendly format such as a standard set of tables or even a database, often with metrics (costs) associated with each route.

A forwarding table, on the other hand, is usually a machine-coded internal one that contains the routes actually used by the device to reach destinations. In most cases, the routing one holds more information than is distilled in the forwarding table.

ROUTERS AND ROUTING TABLES

The router that attaches LAN1 to the world is CE0, a Juniper Networks router. Let's look at the information in the routing table on CE0.

```
admin@CE0> show route
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 3d 02:59:20
                 > via ge-0/0/3.0
10.0.50.0/24       *[Direct/0] 2d 14:25:52
                 > via ge-0/0/3.0
10.0.50.1/32       *[Local/0] 2d 14:25:52
                  Local via ge-0/0/3.0
10.10.11.0/24      *[Direct/0] 2d 14:25:52
                 > via fe-1/3/0.0
10.10.11.1/32      *[Local/0] 2d 14:25:52
                  Local via fe-1/3/0.0

inet6.0: 5 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

::/0               *[Static/5] 2d 13:50:23
                    > via ge-0/0/3.0
fe80::/64          *[Direct/0] 2d 14:25:53
                    > via fe-1/3/0.0
fe80::205:85ff:fe88:ccdb/128
                   *[Local/0] 2d 14:25:53
                     Local via fe-1/3/0.0
fc00:fe67::/32     *[Static/5] 2d 13:50:23
                    > via ge-0/0/3.0
fc00:ffb3:d4:b::/64*[Direct/0] 2d 10:45:08
                    > via fe-1/3/0.0
fc00:ffb3:d4:b:205:85ff:fe88:ccdb/128
                   *[Local/0] 2d 10:45:08
                     Local via fe-1/3/0.0
```

Because both IPv4 and IPv6 addresses are configured, we have both IPv4 and IPv6 routing tables. There's a lot of information here that we'll detail in later chapters on routing protocols, so let's just look at the basics of `CE0`'s routing tables. Only physical addresses are used for now, on the LAN1 interface `fe-1/3/0` and the Gigabit Ethernet link to the provider routers, `ge-0/0/3`. Later, we'll also assign an address to the router's *loopback* interface, but not in this example.

In both tables, there are local, direct, and static entries. Local entries are the full 32- or 128-bit addresses configured on the interfaces. Direct entries are for the network portions of the interface address, so they have prefixes shorter than 32 or 128 bits. For example, the entry for the `fe-1/3/0` interface has a local entry of `10.10.11.1/32` and a direct entry of `10.10.11.0/24`. Both were derived from the configuration of the address string `10.10.11.1/24` to the interface (technically, a string like `10.10.11.1/24` is neither 32-bit host address nor 24-bit network address, but a concatenation of address and network mask).

Static entries are entries that are placed in the routing table by the network administrator, and they stay there no matter what else the router learns about the network. In this case, the static entry is also the ***default route***, a type of "router of last resort" that is used if no other entry in the routing table seems to represent the correct place to forward the packet. The default route matches the entire IPv4 address space, so nothing escapes the default. Note that the highlighted default route for IPv4 is `0.0.0.0/0` (or `0/0`) and sends packets out via interface `ge-0/0/3` onto the service provider router network.

The local and direct entries for the `ge-0/0/3` interface make up the last two entries in this simple five-entry routing table. The default entry basically says to the router, "If you don't know where else to forward the packet, send it out here." This seems trivial, but only because router `CE0` has only two interfaces. Backbone routers can have very complicated routing tables.

Each route in the table has a *preference* associated with the route. A lower value means the route is somehow "better" than another route to the same place having a higher value. The value of `0` associated with local/direct entries means that no other route can be a better way of reaching the locally attached interface, which only makes sense.

Routing table entries often have a *metric* associated with them. Why do routes need both preferences and metrics? Preference indicates *how* the router knows about a route; the metric assigns a *cost* of using the route, no matter how it was learned. Both preference and metric are considered in determining the active route to a destination. Generally, only active routes are loaded into the forwarding table. We'll look at this process more closely in the later chapters of routing. An asterisk (*) marks routes that are both currently active and have been active the last time the router recomputed its routes to use in the forwarding table.

There are no metrics in the `CE0` routing tables. Why? Because metrics are usually assigned by routing protocols and we don't have any routing protocols running yet on `CE0`. Static routes can be configured with metrics, but they still work fine without them.

The six entries in the IPv6 routing table mimic the five entries in the IPv4 table, and the default `::0` static route is highlighted. The only unassigned or "extra" entry is the `fe80::/64` direct route (which is generated automatically) for the link-local prefix for LAN1.

HOSTS AND ROUTING TABLES

Routers are not the only network devices that have routing tables. Hosts have them as well. It's how they know whether to send a packet inside a frame directly to the destination or to send the packet and frame to a router so it can be forwarded to its destination.

The following code block shows what the routing table on `bsdserver` looks like. We can display it with the `netstat -r` command (the `r` option displays network statistics about the routing table). We'll use `netstat -nr` in this chapter because the `n` option forces the output to use IP addresses instead of DNS names. This is a good practice because when trouble strikes the network, chances are that DNS will be down (or provides the wrong information), so it's best to get used to seeing IP addresses in these reports.

```
bsdserver# netstat -nr
Routing tables

Internet:
Destination      Gateway        Flags       Refs      Use       Netif Expire
default          10.10.12.1     UGSc        0         0         em0
10.10.12/24      link#1         UC          0         0         em0
localhost        localhost      UH          0         144       lo0

Internet6:
Destination            Gateway                Flags     Netif Expire
localhost.booklab.     localhost.booklab.     UH        lo0
fe80::%em0             link#1                 UC        em0
fe80::20e:cff:fe3b     00:0e:0c:3b:87:32      UHL       lo0
fe80::%lo0             fe80::1%lo0            Uc        lo0
fe80::1%lo0            link#4                 UHL       lo0
fc00::                 link#1                 UC        em0
fc00::20e:cff:fe3b     00:0e:0c:3b:87:32      UHL       lo0
fc00:fe67:d4:b::       link#1                 UC        em0
fc00:fe67:d4:b:205     00:05:85:8b:bc:db      UHLW      em0
fc00:fe67:d4:b:20e     00:0e:0c:3b:87:32      UHL       lo0
ff01::                 localhost.booklab.     U         lo0
ff02::%em0             link#1                 UC        em0
ff02::%lo0             localhost.booklab.     UC        lo0
```

The IPv4 routing table is even simpler than the `CE0` router's, which we might have expected, because the host only has one interface (`em0`). The third entry (`localhost`) is for the loopback interface (`lo0`), so there are really only two. The `10.10.12/24` entry points to `link#1`, which is the `em0` interface that attaches `bsdserver` to LAN1. It says `Gateway` above the column, but it really means "what is the next hop for this packet?"

Why does it say "gateway" and not "router"? Because technically it *is* a gateway, not a router. A gateway, as mentioned before, connects one or more LANs to the Internet (and can route from LAN to LAN, not just onto or off of the Internet). A router, on the other hand, can have nothing but other routers connected to it. People speak very loosely, of course, and usually the terms "gateway" or "router" can be used without confusion.

So the default entry does point to a router, in this case `CE6`, which is the gateway to the world on LAN2. The `Refs` and `Use` columns are usage indicators, and there is no Expire value because this information, as on router `CE0`, was not learned via a routing protocol and therefore will not get "stale" and need to be refreshed.

The flags commonly seen in FreeBSD follow:

- U (Up)—The route is the active route.
- H (Host)—The route destination is a single host.
- G (Gateway)—Send packets for this destination here, and it will figure out where to forward it.
- S (Static)—A manually configured route that was not generated by protocol or other means.
- C (Clone)—Generates a new route based on this one for devices that we connect to. Normally used for the local network(s).
- W (Was cloned)—A route that was autoconfigured based on a LAN clone route.
- L (Link)—The route references hardware.

Although listed as `default`, the actual entry value for the default route is `0.0.0.0/0` or `0/0`. We can force numeric displays in `netstat` by using the `n` option, but we won't use that here (generally, the fewer options you have to remember to use, the better).

Where's the Metric?

Note the `netstat -nr` on the host did not display any metric values, and `show route` on the router didn't either. In the case of `CE0`, that was explained by the fact that we have no routing protocol running to provide metrics for routes (destination networks). But even if a routing protocol were running, `netstat` never shows any metrics associated with routes. Does that mean hosts have no metrics or do not bother to compute them? Not necessarily, as we'll soon see in the case of Windows XP.

Why is the Internet6 routing table so much larger than either the `Internet` (IPv4) table on `bsdserver` or the tables on router `CE0`? It is larger because of the IPv6 neighbor discovery feature that populates the table with all of the local IPv6 hosts on LAN2. An easy way to spot them is by their MAC addresses in the `Gateway` column. There are also number link-local (`fe80`) and private (`fc00`) entries absent in IPv4, as well as multicast addresses beginning with `ff`.

Let's look at the routing table on `lnxclient` for comparison. We don't have IPv6 running, so the table includes the IPv4 address only. Most of the information is the same as in FreeBSD, just arranged differently.

```
[root@lnxclient admin]# netstat -nr
Kernel IP routing table
Destination    Gateway       Genmask         Flags   MSS Window    irtt Iface
10.10.12.0     *             255.255.255.0   U       0 0           0 eth0
127.0.0.0      *             255.0.0.0       U       0 0           0 lo
default        10.10.12.1    0.0.0.0         UG      0 0           0 eth0
[root@lnxclient admin]#
```

The `Gateway` column has asterisks because we don't have DNS running and the address is the same as the `Destination`. Only the default gateway entry (`10.10.12.1`) is different than the entry (`0.0.0.0/0`). Instead of prefixes, `lnxclient` uses netmask (`Genmask`) notation for the table entries, but either way, the network is `10.10.12.0/24`.

The flags used in Linux follow (note the slightly different meanings compared to FreeBSD):

- G (Gateway)—The route uses a gateway.
- U (Up)—The interface to be used is up.
- H (Host)—Only a single host can be reached by the route.
- D (Dynamic)—The route is not a static route, but a dynamic route learned by a routing protocol.
- M (Modified)—This flag is set if the entry was changed by an ICMP redirect message.
- ! (Exclamation)—The route will reject (drop) all packets sent to it.

Linux hosts have the maximum segment size (`MSS`), `Window` size, and initial round-trip time (`irtt`) lists associated with the route, but these are not IP parameters. They're most useful for TCP, and we'll talk about them in the TCP chapter. And confusingly, a value of 0 in these columns does not mean that their *values* are zero (which would make for an interesting network), but that the *defaults* are used. The `Iface` column shows the interface used to reach the destination address space, with `lo` being loopback.

Finally, Windows hosts have routing tables as well. You can display the routing table contents with the `route print` command or with the same `netstat -nr` command using in Unix-based systems. This output is from `wincli1` and lists only the IPv4 routes.

```
C:\Documents and Settings\Owner>route print
Route Table
===========================================================================
Interface List
0x1 . . . . . . . . . . . . . . . MS TCP Loopback interface
0x2 . . . .00 0e 0c 3b 88 3c. . . Intel(R) PRO/1000 MT Desktop Adapter -
Packet Scheduler Miniport
===========================================================================
===========================================================================
Active Routes:
Network  Destination      Netmask          Gateway      Interface    Metric
         0.0.0.0          0.0.0.0          10.10.11.1   10.10.11.51  10
         10.10.11.51      255.255.255.255  127.0.0.1    127.0.0.1    10
         10.255.255.255   255.255.255.255  10.10.11.51  10.10.11.51  1
         127.0.0.0        255.0.0.0        127.0.0.1    127.0.0.1    1
         224.0.0.0        240.0.0.0        10.10.11.51  10.10.11.51  10
         255.255.255.255  255.255.255.255  127.0.0.1    127.0.0.1    1
Default Gateway:       10.10.11.1
===========================================================================
Persistent Routes:
Network Address    Netmask          Gateway Address    Metric
     10.10.12.0    255.255.255.0    10.10.11.1         1
```

The table looks different, yet is still very familiar. There is an entry for the default gateway (`10.10.11.1`), which is also listed separately for emphasis. One oddity is the classful broadcast address entry (`10.255.255.255`), but this can be changed. There are explicit loopback (`127.0.0.0/8`) and multicast (`224.0.0.0/4`) entries, and a `255.255.255.255/32` entry, as well as for the host itself (`10.10.11.51/32`), which point to the loopback interface.

Instead of relying on a flag, Windows just shows you `Active Routes`. But there is also a `Persistent Route` that is always in the table, no matter what. This was entered in the table manually, like a static route, and makes sure that any packets sent to LAN2 go to the router at `10.10.11.1`. It would still work with only a default route, but this shows how a static route shows up in Windows.

Note that even though no routing protocol is running in the host, `wincli1` assigns metrics to all the routes. These can be changed, but they are always there. But what about when `netstat -nr` is used on the Windows host? We didn't see any metrics on the Unix-based systems. Take a look at what we get with `netstat -nr`.

This output is from `wincli1` and lists only the IPv4 routes.

```
C:\Documents and Settings\Owner>netstat -nr
Route Table
===========================================================================
Interface List
0x1 . . . . . . . . . . . . . . MS TCP Loopback interface
0x2 . . .00 0e 0c 3b 88 3c. . . Intel(R) PRO/1000 MT Desktop Adapter -
Packet Scheduler Miniport
===========================================================================
===========================================================================
Active Routes:
Network   Destination     Netmask          Gateway      Interface    Metric
          0.0.0.0         0.0.0.0          10.10.11.1   10.10.11.51  10
          10.10.11.51     255.255.255.255  127.0.0.1    127.0.0.1    10
          10.255.255.255  255.255.255.255  10.10.11.51  10.10.11.51  1
          127.0.0.0       255.0.0.0        127.0.0.1    127.0.0.1    1
```

```
         224.0.0.0       240.0.0.0         10.10.11.51  10.10.11.51  10
         255.255.255.255  255.255.255.255   127.0.0.1    127.0.0.1    1
Default Gateway:        10.10.11.1
=========================================================================
Persistent Routes:
Network Address    Netmask         Gateway Address    Metric
     10.10.12.0    255.255.255.0   10.10.11.1         1
```

That's right—the output is identical, and *does* show the metrics. However, Windows appears to be the only implementation that shows the metrics associated with routes when `netstat` is used.

Let's take a more detailed look at how routing tables are used to determine whether packets should be sent to the destination directly or to a router for forwarding. We'll see how IP and MAC addresses are used in the packets and frames as well.

DIRECT AND INDIRECT DELIVERY

When routers are used to connect or segment Ethernet LANs, the Ethernet frame that leaves a source may or may not be the same frame that arrives at the destination. If the source and destination host are on the same LAN, then a method sometimes known as *direct delivery* is used and the frame is delivered locally. This means that the source and destination MAC addresses are the same in the frame that is sent from the source and in the frame that arrives at the destination.

Let's see if we can verify that frames are delivered locally, without a router, when the IP address prefix is the same on the destination and on the source. In this case, the MAC addresses on the frame that leave the source and the ones in the frame that arrive at the destination should be the same.

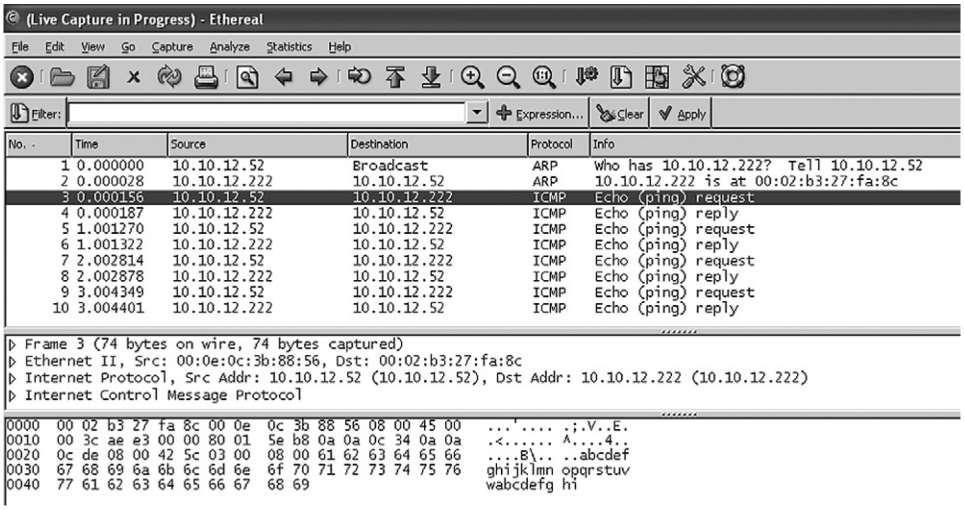
We can also check and make sure that the frames use different MAC addresses when the source and destination hosts are on different IP networks and the frames pass through a router. We can even check and make sure that the frames came from the router.

First, let's use the Windows client and server (which are located in pairs on the two LANs) to generate some packets to capture with Ethereal. We'll use a little utility called "ping" (discussed more fully in Chapter 7) to bounce some packets off the Windows IPv4 addresses.

Ethereal is running on `wincli2`. When we send some pings to the client (`10.10.12.222`) from the Windows server (`10.10.12.52`), what we see is shown in Figure 8.2.

The MAC address `00:02:b3:27:fa:8c` is associated with IPv4 address `10.10.12.222`, and the MAC layer address `00:0e:0c:3b:88:56` is associated with IPv4 address `10.10.12.52`. If we looked at the same stream of pings on the server, the MAC address and IP address associations would be the same. The frame sent is the same as the one that arrives.

What about a packet sent to other IP networks? We'll use a little "echo" client and server utility on the Linux hosts to generate the frames for this exercise. We'll say more


(Live Capture in Progress) - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No. -  Time      Source          Destination     Protocol  Info
     1 0.000000  10.10.12.52     Broadcast       ARP       Who has 10.10.12.222?  Tell 10.10.12.52
     2 0.000028  10.10.12.222    10.10.12.52     ARP       10.10.12.222 is at 00:02:b3:27:fa:8c
     3 0.000156  10.10.12.52     10.10.12.222    ICMP      Echo (ping) request
     4 0.000187  10.10.12.222    10.10.12.52     ICMP      Echo (ping) reply
     5 1.001270  10.10.12.52     10.10.12.222    ICMP      Echo (ping) request
     6 1.001322  10.10.12.222    10.10.12.52     ICMP      Echo (ping) reply
     7 2.002814  10.10.12.52     10.10.12.222    ICMP      Echo (ping) request
     8 2.002878  10.10.12.222    10.10.12.52     ICMP      Echo (ping) reply
     9 3.004349  10.10.12.52     10.10.12.222    ICMP      Echo (ping) request
    10 3.004401  10.10.12.222    10.10.12.52     ICMP      Echo (ping) reply

▷ Frame 3 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: 00:0e:0c:3b:88:56, Dst: 00:02:b3:27:fa:8c
▷ Internet Protocol, Src Addr: 10.10.12.52 (10.10.12.52), Dst Addr: 10.10.12.222 (10.10.12.222)
▷ Internet Control Message Protocol

0000  00 02 b3 27 fa 8c 00 0e  0c 3b 88 56 08 00 45 00   ...'.... .;.V..E.
0010  00 3c ae e3 00 00 80 01  5e b8 0a 0a 0c 34 0a 0a   .<...... ^....4..
0020  0c de 08 00 42 5c 03 00  08 00 61 62 63 64 65 66   ....B\.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

FIGURE 8.2

MAC addresses and direct delivery. Note that the MAC layer addresses in the frame that is sent are the same as in the frame that will arrive at the destination.

about where this little utility came from in the chapter on sockets (Chapter 12). For now, just note that this is *not* the usual Linux echo utility bundled with most distributions. With this utility, we can invoke the server on the `lnxserver` host and use the client to send a simple string to be echoed back by the server process. We'll use tethereal (the text version of Ethereal) this time, just to show that the same information is available in either the graphical or text-based version.

First, we'll run the Echo server process, which normally runs on port 7, on port 55555:

```
[root@lnxserver admin]# ./Echo 55555
```

We have to run tethereal on each end too, if we want to compare frames. The command is the same on the client and server. We'll use the verbose (`–V`) switch to see the MAC layer information as packets arrive.

```
[root@lnxclient admin]# /usr/sbin/tethereal-V
Capturing on eth0
```

Now we can invoke the Echo client to bounce the string `TESTING123` off the server process.

```
[root@lnxclient admin]# ./Echo 10.10.11.66 TESTING123 55555
Received: TESTING123
[root@lnxclient admin]#
```

What did we get? Let's look at the frames leaving the client. We only need to examine the Layer 2 and IP address information.

```
[root@lnxclient admin]# /usr/sbin/tethereal-V
Capturing on eth0
Frame 1 (74 bytes on wire, 74 bytes captured)
    Arrival Time: May 5, 2008 13:39:34.102363000
    Time delta from previous packet: 0.000000000 seconds
    Time relative to first packet: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 74 bytes
    Capture Length: 74 bytes
Ethernet II, Src: 00:b0:d0:45:34:64, Dst: 00:05:85:8b:bc:db
    Destination: 00:05:85:8b:bc:db (Juniper__8b:bc:db)
    Source: 00:b0:d0:45:34:64 (Dell_45:34:64)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.10.12.166 (10.10.12.166), Dst Addr: 10.10.11.66
(10.10.11.66)
    Version: 4
    Header length: 20 bytes... [much more information not shown]
```

We can see that the Ethernet frame leaving the Linux client has source MAC address `00:b0:d0:45:34:64` and destination MAC address `00:05:85:8b:bc:db`. The packet inside the frame has the source IPv4 address `10.10.12.166` and destination address `10.10.11.66`, as expected.

How do we know that the destination MAC address `00:05:85:8b:bc:db` is not associated with the destination address `10.10.11.66`? We can simply look at the frame that arrives at the Linux server.

```
[root@lnxserver admin]# /usr/sbin/tethereal -V
Capturing on eth0
Frame 1 (74 bytes on wire, 74 bytes captured)
    Arrival Time: May 5, 2008 13:39:34.104401000
    Time delta from previous packet: 0.000000000 seconds
    Time relative to first packet: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 74 bytes
    Capture Length: 74 bytes
Ethernet II, Src: 00:05:85:88:cc:db, Dst: 00:d0:b7:1f:fe:e6
    Destination: 00:d0:b7:1f:fe:e6 (Intel_1f:fe:e6)
    Source: 00:05:85:88:cc:db (Juniper__88:cc:db)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.10.12.166 (10.10.12.166), Dst Addr: 10.10.11.66
(10.10.11.66)
    Version: 4
    Header length: 20 bytes...(much more information not shown)
```

Note that the frame arriving at `10.10.11.66` has the MAC address `00:d0:b7:1f:fe:e6`, which is not the one used as the destination MAC address in the frame leaving the `10.10.12.166` client (that address is `00:b0:d0:45:34:64`).

Table 8.1 Frame IP and MAC Addresses

	MAC Source Address	**IP Source Address**	**MAC Destination Address**	**IP Destination Address**
Frame leaving client	`00:b0:d0:45:34:64` (Linux client)	`10.10.12.166` (Linux client)	`00:05:85:8b:bc:db` (Juniper router)	`10.10.11.66` (Linux server)
Frame arriving at server	`00:05:85:88:cc:db` (Juniper router)	`10.10.12.166` (Linux client)	`00:d0:b7:1f:fe:e6` (Linux server)	`10.10.11.66` (Linux server)

Now, if the MAC address associated with the frame leaving the `10.10.12.166` client is `00:bo:do:45:34:64`, then the MAC address associated with the same IP address on the server LAN cannot magically change to `00:05:85:88:cc:db`. As expected, the IP packet is identical (except for the decremented TTL field), but the *frame* is different. This is sometimes called *indirect delivery* of packets because the packet is sent through one or more network nodes and not directly to the destination.

These relationships are displayed in Table 8.1, which shows how the MAC addresses relate to the IP subnet addresses.

Tethereal not only gives the MAC addresses, but also parses the 24-bit OUI and helpfully lists Intel as the owner of `00:d0:b7` and Juniper as the owner of `00:05:85`. We can verify this on the Linux client or server. Let's look at the client's ARP cache.

```
[root@lnxclient admin]# /sbin/arp -a
? (10.10.12.1) at 00:05:85:8b:bc:db [ether] on eth0
[root@lnxclient admin]#
```

The question mark (?) just means that our routers do not have names in DNS.

The Illustrated Network uses two small LAN switches for LAN1 and LAN2, but the nodes used for internetworking are routers. Let's take a closer look at just what a router does and how it delivers packets from LAN to LAN over an internetwork.

Routing

Routing is done entirely with IP addresses, of course. Many books make extensive use of the concepts of *direct routing* and *indirect routing* of packets. This can be confusing, since direct "routing" of packets does not require a router. In this chapter, the terms *direct delivery* and *indirect delivery* are used instead. A host can use direct delivery to send packets directly to another host, perhaps using a VLAN, or use indirect delivery if the destination host is reachable only through a router.

How does the source host know whether the destination host is reachable through direct (local) delivery or indirect (remote) delivery through a router? The answer has a lot to do with the way bridges and routers differ in their fundamental operation, and how routers use the IP address to determine how to handle packets. Here's an example using the Illustrated Network's actual MAC and IP addresses.

Direct Delivery without Routing

Let's look at a packet sent from `wincli` on LAN1 to `winsvr1`. Both of these hosts are on LAN1, so no routing is needed. The IPv4 addresses are `10.10.11.51` for `wincli1` and `10.10.11.111` for `winsvr1`, and both use the same `255.255.255.0` mask. Therefore, both addresses have the same network portion of the IPv4 address, `10.10.11.0/24`.

The host software knows that no router is needed to handle a packet sent from the source host to the destination host because the IP addresses of the source and destination hosts have the *same IP network portion* (prefix) in both source and destination IP addresses. This is a simple and effective way to let hosts know whether they are on the same LAN. The packet can be placed in a frame and sent directly to the destination using the local link. This is shown in Figure 8.3.

In Figure 8.3, a packet is followed from client to server when both are on the same LAN segment and there is no router between client and server. All direct delivery means is that the packet and frame do not have to pass through a router on the way from source to destination.

The TCP/IP protocol stack on the client builds the TCP header and IP header. In Figure 8.3, the IP packet is placed inside an Ethernet MAC frame. The MAC source and destination addresses are shown as well. The client knows its own MAC address, and if

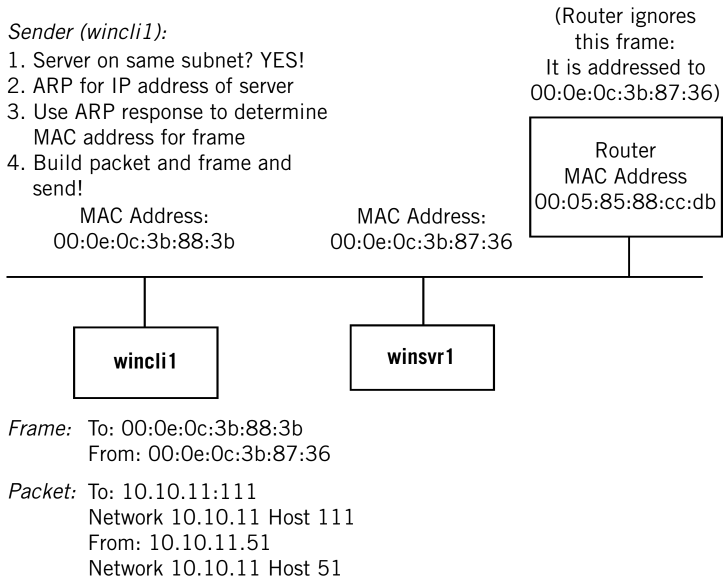

Sender (wincli1):
1. Server on same subnet? YES!
2. ARP for IP address of server
3. Use ARP response to determine MAC address for frame
4. Build packet and frame and send!
(Router ignores this frame: It is addressed to 00:0e:0c:3b:87:36)
Router MAC Address 00:05:85:88:cc:db
MAC Address: 00:0e:0c:3b:88:3b
MAC Address: 00:0e:0c:3b:87:36
wincli1
winsvr1
Frame: To: 00:0e:0c:3b:88:3b
From: 00:0e:0c:3b:87:36
Packet: To: 10.10.11:111
Network 10.10.11 Host 111
From: 10.10.11.51
Network 10.10.11 Host 51

FIGURE 8.3

Direct delivery of packets on a LAN. Note that the MAC address does not change from source to destination, and that the router ignores the frame.

the server's MAC address is not cached, an ARP broadcast message that asks, "Who has IP address `10.10.11.111`?," is used to determine the MAC address of the server.

The source host knew to ask for the MAC address of the destination host because the destination host is on the same LAN as the source. Hosts with the same IP network addresses must be on the same LAN segment. Destination hosts on the same LAN are simply "asked" to provide their MAC addresses. The destination MAC address in the frame is the MAC address that corresponds to the destination IP address in the IP packet inside the MAC frame.

What would be different when the client and server are on different LANs and must communicate through a router?

Indirect Delivery and the Router

It is one thing to say that the router is the network node of the Internet, but exactly what does this mean? What is the role of the router on the Internet? Routers route IP packets to perform *indirect delivery* (through the *forwarding*) of packets from source to destination.

Unlike direct delivery, where the packets are sent between devices on the same LAN, indirect delivery employs one or more routers to connect source and destination. The source and destination could be near in terms of distance, perhaps on separate floors of the same building. All that really matters is whether there is a router between source and destination or not.

Figure 8.4 shows a simple network consisting of two LANs connected by routers. The routers are connected by a serial link using PPP, but SONET would do just as well. Of course, the Internet consists of thousands of LANs and routers, but all of the essentials of routing can be illustrated with this simple network.

The routing network has been simplified to emphasize the architectural features without worrying about the details. The routers are just Router 1 and Router 2, not `CE0` and `CE6`. But the LANs are still LAN1 and LAN2, and we'll trace a packet from `wincli1` on LAN1 to `winsvr2` on LAN2.

Both LAN segments in Figure 8.4 are implemented with Ethernet hubs and unshielded twisted pair (UTP) wiring, but are shown as shared media cables, just to make the adjacencies clearer. Each host in the figure has a network interface card (NIC) installed. It is important to realize that it is the *interface* that has the IP address, not the entire host, but in this example each host has only one interface. However, the routers in the figure have more than one network interface and therefore more than one IP network address. A router is a network device that belongs to two or more networks at the same time, which is how they connect LANs. A typical router can have 2, 8, 16, or more interfaces. Each interface usually gets an IP address and typically represents a separate "network" as the term applies to IP, but there are exceptions.

Each NIC in a host or router has a MAC address, and these are given in Figure 8.4. The routers are only shown with network layers and IP layers, because that's all they need for packet forwarding (most routers do have application layers, as we have seen). Because the routers in this example are in different locations, they are connected by a

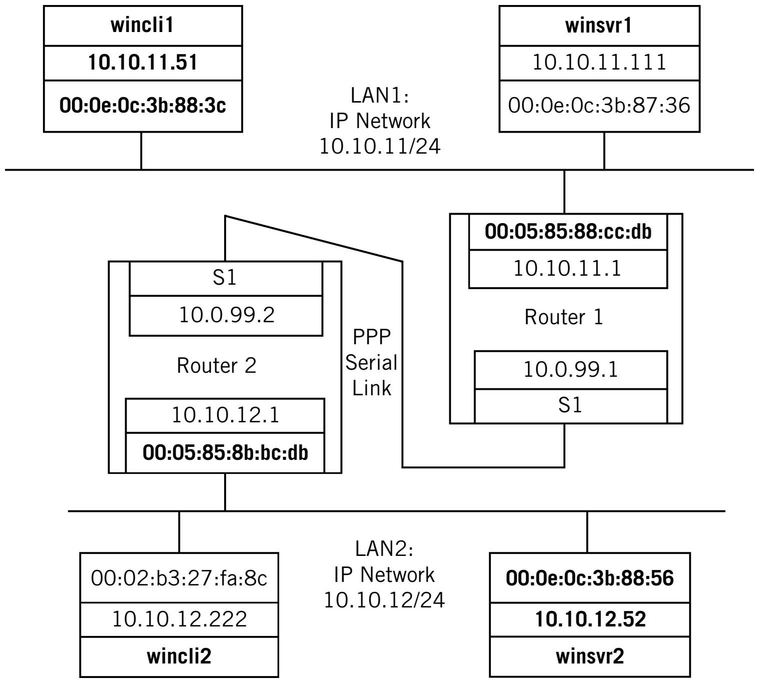


FIGURE 8.4

Indirect delivery using a router. Note the different MAC and link-level addresses in place between source and destination.

serial link. The serial link is running PPP and packets are placed inside PPP frames on this link between the routers. There is no need for global uniqueness on serial ports, since they are point-to-point links in the example, so each is called "S1" (Serial1) at the network layer. They don't even require IP addresses, but these are usually provided to make the link visible to network management and make routing and forwarding tables a lot simpler.

All of the pieces are now in place to follow a packet between client and server on the "internetwork" in Figure 8.4 using indirect delivery of packets with routers. Let's see what happens when a client process running on `wincli1` wants to send a packet to a server process running on `winsvr2`. The application is unimportant. What is important is that the source host knows that the destination host (server) is not on the same LAN. Once the IP address of the server is obtained, it is obvious to the source that the destination IP network address (`10.10.12.52`) is different than the source IP network address (`10.10.11.51`).

The source client software now knows that the packet going to `10.10.12.52` must be sent through at least one router, and probably several routers, using indirect delivery. It is called indirect delivery (or indirect routing) *because the packet destination*

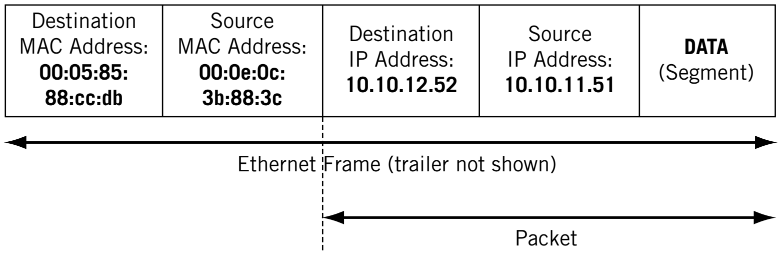

Destination MAC Address: 00:05:85: 88:cc:db
Source MAC Address: 00:0e:0c: 3b:88:3c
Destination IP Address: 10.10.12.52
Source IP Address: 10.10.11.51
DATA (Segment)
Ethernet Frame (trailer not shown)
Packet

FIGURE 8.5

Frame and packet sent to Router1, showing source and destination IP and MAC addresses.

address is the destination IP address of `winsvr2`***, but the initial frame destination address is the MAC address of the Router1.*** The packet is sent *indirectly* to the destination host inside a frame sent to the router. The address fields of the frame and packet constructed and sent on the LAN by `wincli1` are shown in Figure 8.5.

Note that the *frame* is sent to Router1's MAC address (`00:05:85:88:cc:db`), but the *packet* is sent to `10.10.12.52` (`winsvr2`). This is how routing works. (Bridges, or direct delivery even in routing, always has frames in which the destination MAC address is the same as the IP address it represents.)

How did the source host, `wincli1`, know the MAC address of the correct router? There could be several routers on a LAN, if for no other reason than redundancy. All that `wincli1` did was use the routing table to look up the IP address of the destination. But there's no specific entry for a network associated with `10.10.12.52`. However, TCP/IP configuration on a host often includes configuration of at least one *default gateway* to be used when packets must leave the local LAN. The default gateway (a router in this case) can be set statically, or dynamically using the Dynamic Host Configuration Protocol (DHCP), or even other ways. In this example network, the default gateway IP address has been entered statically when the host was configured for TCP/IP.

Since the default gateway is by definition on the same LAN as the source host (they share the same IP address prefix), the source host can just send an ARP to get the MAC address of the interface on the router attached to that LAN. Note that the IP address of the router is used only to get the MAC address of the router, not so that the source host `wincli1` can send packets to the router (the packets are being forwarded to `winsvr2`).

When this packet is sent, the router pays attention to the frame when it arrives, but `winsrv1` ignores it (the frame is not for `00:0e:0c:3b:87:36`). Router1 looks at the packet inside the frame and knows that the destination host is not directly connected to Router1. The *next hop* to the destination is another router. How does Router1 know? In much the same way as `wincli1`: Router1 compares the destination IP address to the IP addresses assigned to its local interfaces. These are `10.10.11.0/24` and `10.0.99.0/24`. The packet's destination IP address of `10.10.12.0/24` does not belong to either of the two networks local to Router1.

However, a router can have many interfaces, not just the two in this example. Which output port should the router use to forward the packet? The network portion of the IP

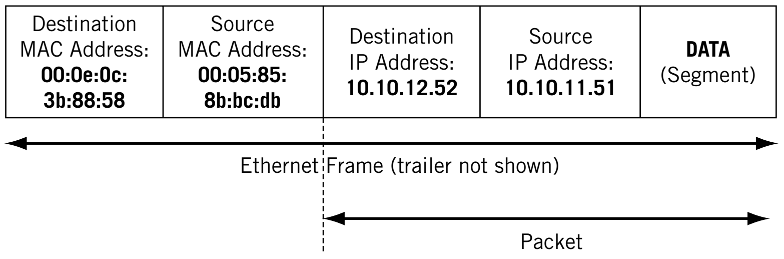

Destination MAC Address: 00:0e:0c:3b:88:58
Source MAC Address: 00:05:85:8b:bc:db
Destination IP Address: 10.10.12.52
Source IP Address: 10.10.11.51
DATA (Segment)
Ethernet Frame (trailer not shown)
Packet

FIGURE 8.6

Frame sent by Router2 to `winsvr2`, showing source and destination IP and MAC addresses.

address is looked up in the forwarding table according to certain rules to find out the IP address of the next-hop router and the output interface leading to this router. (In practice, Router1 might simply have a default route pointed at the serial WAN interface.) The rules used for these lookups will be discussed in more detail in a later chapter. For now, assume that Router1 finds out that the next hop for the packet to `winsvr2` is Router2, and that Router2 is reached on serial port S1.

Router1 now encapsulates the packet from `wincli1` to `winsvr2` inside a PPP frame for transport on the serial link. Another key feature distinguishing routers from bridges, as we have seen, is an IPv4 router's ability to *fragment* a packet for transport on an output link. Fragmentation depends on every router knowing the maximum transmission unit (MTU) frame size for the link types on all of the router's interfaces. Ethernet LANs, for example, all have an MTU size of 1500 bytes (1518 bytes, including the LAN frame header). Serial links usually have MTU sizes larger than that, so this example assumes that Router1 does not have to fragment the content of the packet it received from the LAN.

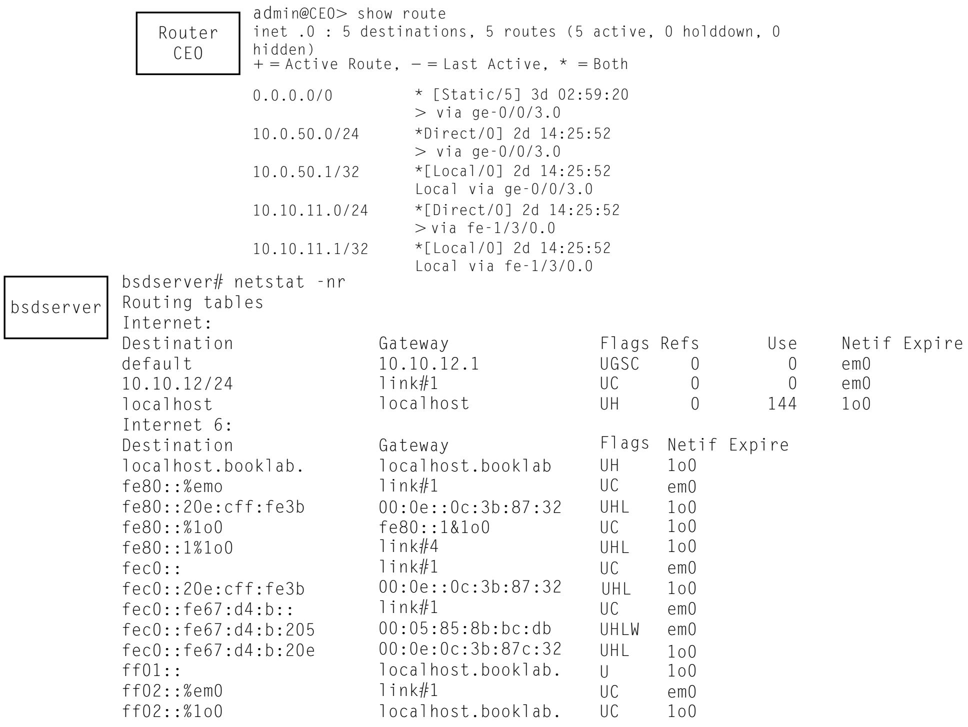
When the packet sent by `wincli1` to `winsvr2` arrives at Router2 on the serial link from Router1, Router2 knows that the next hop for this packet is *not* another router. Router2 can deliver the packet directly to `winsvr2` using direct delivery. How does it know? Because the network portion of the IP address in the packet destination, `10.10.12.52/24`, is on the same network as the router on one of its interfaces, `10.10.12.1/24`. In brief, it has a route that covers the destination network on one of its interfaces.

The frame containing the packet is sent onto the LAN with the structure shown in Figure 8.6. Note that in this case the MAC address of the *source* is Router2, and the MAC address of the destination is the MAC address of `winsrv2`. Again, Router2 can always use ARP to get the MAC address associated with IP address `10.10.12.52` if the MAC address of the destination host is not in the local ARP cache on the router. The source and destination IP addresses on the packet do not change in this example, of course. `Winsvr2` must be able to reply to the sender, `wincli1` in this case. (We'll talk about cases using NAT, when the source and destination packet addresses do and must change, in the chapter on NAT.)

It is assumed that there is no problem with MTU sizes in this example. However, MTU sizes are often important, especially when the operational differences between IPv4 and IPv6 routers, when it comes to fragmentation, are considered.

QUESTIONS FOR READERS

Figure 8.7 shows some of the concepts discussed in this chapter and can be used to help you answer the following questions.



```
admin@CEO> show route
inet .0 : 5 destinations, 5 routes (5 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          * [Static/5] 3d 02:59:20
                   > via ge-0/0/3.0
10.0.50.0/24       *Direct/0] 2d 14:25:52
                   > via ge-0/0/3.0
10.0.50.1/32       *[Local/0] 2d 14:25:52
                   Local via ge-0/0/3.0
10.10.11.0/24      *[Direct/0] 2d 14:25:52
                   >via fe-1/3/0.0
10.10.11.1/32      *[Local/0] 2d 14:25:52
                   Local via fe-1/3/0.0
```

```
bsdserver# netstat -nr
Routing tables
Internet:
Destination              Gateway                Flags Refs      Use    Netif Expire
default                  10.10.12.1             UGSC     0        0    em0
10.10.12/24              link#1                 UC       0        0    em0
localhost                localhost              UH       0      144    lo0
Internet 6:
Destination              Gateway                Flags  Netif Expire
localhost.booklab.       localhost.booklab      UH     lo0
fe80::%emo               link#1                 UC     em0
fe80::20e:cff:fe3b       00:0e::0c:3b:87:32     UHL    lo0
fe80::%lo0               fe80::1&lo0            UC     lo0
fe80::1%lo0              link#4                 UHL    lo0
fec0::                   link#1                 UC     em0
fec0::20e:cff:fe3b       00:0e::0c:3b:87:32     UHL    lo0
fec0::fe67:d4:b::        link#1                 UC     em0
fec0::fe67:d4:b:205      00:05:85:8b:bc:db      UHLW   em0
fec0::fe67:d4:b:20e      00:0e:0c:3b:87c:32     UHL    lo0
ff01::                   localhost.booklab.     U      lo0
ff02::%em0               link#1                 UC     em0
ff02::%lo0               localhost.booklab.     UC     lo0
```

FIGURE 8.7

The routing table output from router `CEO` (IPv4 only) and host `bsdserver`.

1. What is the difference between a routing table and a forwarding table?
2. In the IPv6 routing table for router `CEO`, what is the IPv6 address associated with interface `ge-0/0/3`?
3. In the IPv6 routing table for router `CEO`, what is the precise IP address value of the default route for IPv4 and IPv6?
4. Why are there so many entries in the IPv6 host routing table on `bsdserver`?
5. What is a "persistent" route? What is a "static" route?