

[Figures are not included in this sample chapter]

# CCIE Professional Development: Routing TCP/IP, Volume I

## - 3 -

### Static Routing

- The Route Table
- Configuring Static Routes
- Case Study: Simple Static Routes
- Case Study: Summary Routes
- Case Study: Alternative Routes
- Case Study: Static Floating Routes
- Case Study: Load Sharing
- Case Study: Recursive Table Lookups
- Troubleshooting Static Routes
- Case Study: Tracing a Failed Route
- Case Study: A Protocol Conflict

An important observation from Chapter 2, "TCP/IP Review," is that the data link/physical layers and the transport/network layers, as defined by the OSI model, perform very similar duties: They provide the means for conveying data from a source to a destination across some path. The difference is that the data link/physical layers provide communications across a physical path, whereas the transport/network layers provide communications across a logical or virtual path made up of a series of data links.

Further, Chapter 2 showed that for communications to take place across a physical path, certain information about data link identifiers and encapsulations must be acquired and stored in a database such as the ARP cache. Similarly, information that the transport/network layers require to do their job must also be acquired and stored. This information is stored in the route table, also known as the forwarding database.

This chapter examines what sort of information is required to route a packet, how that information is stored in the route table, how to enter the information into the database, and some techniques for building a routed internetwork by entering the proper information into the proper routers' route tables.

## The Route Table

To understand the kind of information that exists in the route table, it is useful to begin with an examination of what happens when a framed packet arrives at one of a router's interfaces. The data-link identifier in the frame's destination address field is examined. If it contains either the identifier of the router's interface or a broadcast identifier, the router strips off the frame and passes the enclosed packet to the network layer. At the network layer, the destination address of the packet is examined. If the destination address is either the IP address of the router's interface or an all-hosts broadcast address, the protocol field of the packet is examined and the enclosed data is sent to the appropriate internal process.

Any other destination address calls for routing. The address may be for a host on another network to which the router is attached (including the router interface attached to that network) or for a host on a network not directly connected to the router. The address may also be a directed broadcast, in which there is a distinct network or subnet address, and the remaining host bits are all ones. These addresses are also routable.

If the packet is to be routed, the router will do a route table lookup to acquire the correct route. At a minimum, each route entry in the database must contain two items:

- A destination address. This is the address of the network the router can reach. As this chapter explains, the router may have more than one route to the same address and/or a group of subnets of the same or of varying lengths grouped under the same major IP network address.
- A pointer to the destination. This pointer either will indicate that the destination network is directly connected to the router or it will indicate the address of another router on a directly connected network. That router, which will be one router hop closer to the destination, is a next-hop router.

The router will match the most specific address it can. In descending order of specificity, the address may be one of the following:

- A host address (a host route)
- A subnet
- A group of subnets (a summary route)
- A major network number
- A group of major network numbers (a supernet)
- A default address

This chapter provides examples of the first four types. Supernetting is covered in Chapter 7, "Routing Information Protocol Version 2." A default address is considered a least-specific address and is matched only if no other match can be found. Default addressing is the topic of Chapter 12, "Default Routes and On-Demand Routing."

If the destination address of the packet cannot be matched to any route table entry, the packet is dropped and a Destination Unreachable ICMP message is sent to the source address.

Figure 3.1 shows a simple internetwork and the route table entries required by each router. Of

primary importance here is the "big picture," seeing how the route tables work as a whole to transport packets correctly and efficiently. The destination addresses that the router can reach are listed in the Network column of the route tables. The pointers to the destinations are in the Next Hop column.

If router Carroll in Figure 3.1 receives a packet with a source address of 10.1.1.97 and a destination address of 10.1.7.35, a route table lookup determines that the best match for the destination address is subnet 10.1.7.0, reachable via next-hop address 10.1.2.2, on interface S0. The packet is sent to that next router (Dahl), which does a lookup in its own table and sees that network 10.1.7.0 is reachable via next-hop address 10.1.4.2, out interface S1. The process continues until the packet reaches router Baum. That router, receiving the packet on its interface S0, does a lookup, and sees that the destination is on one of its directly connected networks, out E0. Routing is completed, and the packet is delivered to host 10.1.7.35 on the Ethernet link.

The routing process, as explained, assumes that the router can match its listed next-hop addresses to its interfaces. For example, router Dahl must know that Lewis's address 10.1.4.2 is reachable via interface S1. Dahl will know from the IP address and subnet mask assigned to S1 that S1 is directly connected to subnet 10.1.4.0. It then knows that 10.1.4.2, a member of the same subnet, must be connected to the same network.

Notice that every router must have consistent and accurate information for correct packet switching to occur. For example, in Figure 3.1 an entry for network 10.1.1.0 is missing from Dahl's route table. A packet from 10.1.1.97 to 10.1.7.35 will be delivered, but when a reply is sent from 10.1.7.35 to 10.1.1.97, the packet is passed from Baum to Lewis to Dahl. Dahl does a lookup and finds that it has no entry for subnet 10.1.1.0, so the packet is dropped, and an ICMP Destination Unreachable message is sent to host 10.1.7.35.

Figure 3.2 shows the actual Cisco route table from router Lewis of Figure 3.1. The command for examining the IP route table of a Cisco router is `show ip route`.

Examine the contents of this database and compare it with the generic table shown for Lewis in Figure 3.1. A key at the top of the table explains the letters down the left side of the table. These letters indicate how each route entry was learned; in Figure 3.2 all routes are tagged with either a C for "directly connected," or an S for "static entry." The statement "gateway of last resort is not set" refers to a default route.

At the top of the table is a statement indicating that the route table knows of seven subnets of the major network address 10.0.0.0, subnetted with a 24-bit mask. For each of the seven route entries, the destination subnet is shown; for the entries that are not directly connected--routes for which the packet must be forwarded to a next-hop router--a bracketed tuple indicates [administrative distance/metric] for that route. Administrative distances are introduced later in this chapter and are covered in detail in Chapter 11, "Route Redistribution."

Metrics, discussed in greater detail in Chapter 4, "Dynamic Routing Protocols," are a way for routes to be rated by preference--the lower the metric, the "shorter" the path. Notice that the static routes shown in Figure 3.2 have a metric of 0. Finally, either the address of the directly connected interface of the next-hop router or the interface to which the destination is connected is shown.

## Configuring Static Routes

The route table acquires information in two ways. The information may be entered manually, by means of a static route entry, or automatically by one of several systems of automatic information discovery and sharing known as dynamic routing protocols. The bulk of this book concerns dynamic IP routing protocols, but this discussion of static route configuration will prepare you to understand the subsequent chapters.

More to the point, static routing is preferred over dynamic routing in certain circumstances. As with any process, the more automatic it is, the less control we have over it. Although dynamic (automatic) routing requires much less human intervention, static routing allows very precise control over the routing behavior of an internetwork. The price to be paid for this precision is the necessity of manual reconfiguration any time the topology of the network changes.

## Case Study: Simple Static Routes

Figure 3.3 shows an internetwork with four routers and six networks. Notice that the subnets of network 10.0.0.0 are discontinuous--there is a different major network subnet (192.168.1.192, in the Tiger-to-Piglet link) separating 10.1.0.0 from the other 10.0.0.0 subnets. The subnets of 10.0.0.0 are also variably subnetted--the subnet masks are not consistent throughout the internetwork. Finally, the subnet address of Pooh's Ethernet link is an all-zero subnet. Later chapters demonstrate that an addressing scheme with these characteristics causes problems for simpler, classful routing protocols such as RIP and IGRP; static routes work fine here.

The procedure for statically routing an internetwork has three steps:

1. For each data link within the internetwork, identify all addresses (subnet or network).
2. For each router, identify all data links not directly connected to that router.
3. For each router, write a route statement for each data link not directly connected to it.

Writing route statements for a router's directly connected data links is unnecessary, because the addresses and masks configured on the router's interfaces cause those networks to be recorded in its route table.

For example, the internetwork in Figure 3.3 has six subnets:

- 10.1.0.0/16
- 10.4.6.0/24
- 10.4.7.0/24
- 192.168.1.192/27
- 192.168.1.64/27
- 192.168.1.0/27

To configure static routes for Piglet, the subnets that are not directly connected are identified as follows:

- 10.4.6.0/24

- 10.4.7.0/24
- 192.168.1.64/27
- 192.168.1.0/27

These are the subnets for which static routes must be written. The routing commands themselves are easily read if the reader remembers that each command describes a route table entry. The command is `ip route`, followed by the address to be entered into the table, a mask for determining the network portion of the address, and the address of the directly connected interface of the next-hop router.

An alternative configuration command for static routes specifies the interface out of which a network is reached instead of the address of the next-hop router. For example, the route entries for Tigger could be as follows:

```
Tigger(config)# ip route 192.168.1.0 255.255.255.224 S0
```

```
Tigger(config)# ip route 10.1.0.0 255.255.0.0 E0
```

```
Tigger(config)# ip route 10.4.7.0 255.255.255.0 S1
```

Figure 3.4 compares the route table resulting from this configuration with the route table resulting from entries pointing to a next-hop router. Notice that a certain inaccuracy is introduced; all networks specified with a static route referring to an exit interface are entered into the table as if they are directly connected to that interface. The implications for route redistribution are discussed in Chapter 11.

A point of interest in Figure 3.4 is that the header for the 10.0.0.0 subnets indicates the variable subnet masks used in the internetwork. Variable Length Subnet Masking (VLSM) can be a useful tool and is discussed at length in Chapter 7.

## Case Study: Summary Routes

A summary route is an address that encompasses several more specific addresses in a route table. It is the address mask used with a route entry that makes static routes as flexible as they are; by using an appropriate address mask, it is sometimes possible to create a single summary route for several destination addresses.

For example, the preceding case study uses a separate entry for each data link. The mask of each entry corresponds to the address mask used on the device interfaces connected to that data link. Looking again at Figure 3.3, you can see that subnets 10.4.6.0/24 and 10.4.7.0/24 could be specified to Piglet with a single entry of 10.4.0.0/16, reachable via Tigger. Likewise, subnets 192.168.1.0/27 and 192.168.1.64/27 could be accounted for in its route table with a single entry pointing to 192.168.1.0/24, also reachable via Tigger. These two route entries, 10.4.0.0/16 and 192.16.1.0/24, are summary routes.

All subnets of network 10.0.0.0 are reachable from Pooh via Tigger, so a single entry to that major network address and a corresponding mask is all that is needed:

From Eeyore, all destination addresses beginning with 192 are reachable via Tigger. The single route entry does not even have to specify all of the class C address bits:

```
Eeyore(config)# ip route 192.0.0.0 255.0.0.0 10.4.6.1
```

```
Eeyore(config)# ip route 10.1.0.0 255.255.0.0 10.4.6.1
```

By summarizing a group of subnets or even major networks, the number of static route entries may be reduced drastically--in this example, by more than one-third. However, caution must be used when summarizing addresses; when done incorrectly, unexpected routing behavior may occur (see "Case Study: Tracing a Failed Route," later in this chapter). Summarization and the problems that can develop from incorrect summarization are examined in more depth in Chapters 8, "Enhanced Interior Gateway Routing Protocol (EIGRP)," and 9, "Open Shortest Path First."

## Case Study: Alternative Routes

In Figure 3.5, a new link has been added between Pooh and Eeyore. All packets from Pooh to the 10.0.0.0 networks will take this new path with the exception of packets destined for the host 10.4.7.25; a policy is in place stating that traffic to this host must go through Tigger. The static route commands at Pooh will be:

The first two route entries are the same as before except that the second path now points to the new interface 192.168.1.34 at Eeyore. The third entry is a host route, pointing to the single host 10.4.7.25 and made possible by setting the address mask to all ones. Notice that unlike the entry for the other 10.0.0.0 subnets, this host route points to Tigger's interface 192.168.1.66.

The debugging function `debug ip packet` is turned on in Pooh (Figure 3.6) to observe the paths packets take from the router as a result of the new route entries. A packet is sent from a host 192.168.1.15 to host 10.4.7.25. The first two debug trap messages show that the packet is routed from interface E0 to the next-hop router 192.168.1.66 (Tigger) out interface S0, as required, and that the reply packet was received on S0 and routed to the host 192.168.1.15 out E0.

Next a packet is sent from host 192.168.1.15 to host 10.4.7.100. Packets destined for any host on 10.0.0.0 subnets, other than host 10.4.7.25, should be routed across the new link to Eeyore's interface 192.186.1.34. The third debug message verifies that this is indeed happening. However, the fourth message shows something that may initially be surprising. The response from 10.4.7.100 to 192.168.1.15 arrived on Pooh's interface S0 from Tigger.

Remember that the route entries in the other routers have not changed from the original example. This result may or may not be desired, but it does illustrate two characteristics of static routes. First, if the internetwork topology changes, the routers that are required to know about those changes must be reconfigured; and second, static routes can be used to create very specific routing behavior. In this example, perhaps it is desirable to have traffic taking one path in one direction and another path in the opposite direction.

A final observation about this example is that packets routed from Pooh to subnet 10.1.5.0 take a less-than-optimal route, from Pooh to Eeyore to Tigger instead of directly from Pooh to Tigger. A more efficient configuration is:

The third entry will now send all packets for subnet 10.1.5.0 directly to Tigger.

## Case Study: Floating Static Routes

Unlike other static routes, a floating static route is not permanently entered in the route table; it appears only under the special circumstance of the failure of a more-preferred route.

In Figure 3.7, a new router (Rabbit) is connected to Piglet via their respective Serial 0 interfaces, but a new connection has been added between the two Serial 1 interfaces. This second link has been added for redundancy: If the primary link 10.1.10.0 fails, floating static routes will direct traffic across the backup link 10.1.20.0.

Additionally, the mask on Piglet's Ethernet interface has changed from 10.1.5.1/16 to 10.1.5.1/24. This change allows the single route entry at Tigger ip route 10.1.0.0 255.255.0.0 192.168.1.194 to point not only to 10.1.5.0 but also to all of the new subnets used in association with the new router.

To create the floating static route, Piglet's route entries are as follows:

```
ip route 192.168.1.0 255.255.255.0 192.168.1.193
ip route 10.4.0.0 255.255.0.0 192.168.1.193
ip route 10.1.30.0 255.255.255.0 10.1.10.1
ip route 10.1.30.0 255.255.255.0 10.1.20.1 50
```

And Rabbit's route entries are:

```
ip route 10.4.0.0 255.255.0.0 10.1.10.1
ip route 10.4.0.0 255.255.0.0 10.1.20.1 50
ip route 10.1.5.0 255.255.255.0 10.1.10.1
ip route 10.1.5.0 255.255.255.0 10.1.20.1 50
ip route 192.168.0.0 255.255.0.0 10.1.10.1
ip route 192.168.0.0 255.255.0.0 10.1.20.1 50
```

Two entries at Piglet point to Rabbit's network 10.1.30.0; one specifies a next-hop address of Rabbit's S0 interface, and the other specifies a next-hop address of Rabbit's S1 interface. Rabbit has similar double entries for every route.

Notice that all static routes using subnet 10.1.20.0 are followed by a 50. This number specifies an administrative distance, which is a measure of preferability; when duplicate paths to the same network are known, the router will prefer the path with the lower administrative distance. At first this idea sounds like a metric; however, a metric specifies the preferability of a route, whereas an administrative distance specifies the preferability of the means by which the route was discovered.

For example, static routes pointing to a next-hop address have an administrative distance of 1, and

static routes referencing an exit interface have an administrative distance of 0. If two static routes point to the same destination, but one references a next-hop address and one references an exit interface, the latter--with the lower administrative distance--will be preferred.

By increasing the administrative distances of the static routes traversing subnet 10.1.20.0 to 50, they become less preferred than the routes traversing subnet 10.1.10.0. Figure 3.8 shows three iterations of Rabbit's route table. In the first table, all routes to nonconnected networks use a next-hop address of 10.1.10.1. The bracketed numbers associated with each route indicate an administrative distance of 1 and a metric of 0 (because no metrics are associated with static routes).

Next, trap messages announce that the state of the primary link connected to Serial 0 has changed to "down," indicating a failure. A look at the second iteration of the route table shows that all nonconnected routes now point to a next-hop address of 10.1.20.1. Because the more-preferred entry is no longer available, the router has switched to the less-preferred backup link, with the administrative distance of 50 indicated in the brackets. And because subnet 10.1.10.0 has failed, it no longer shows up in the route table as a directly connected network.

Before the third iteration of the route table, trap messages indicate that the state of the primary link has changed to "up." The route table then shows that subnet 10.1.10.0 is again in the table, and the router is again using the next-hop address of 10.1.10.1.

Chapter 11 discusses the administrative distances associated with the various dynamic routing protocols, but it can be said here that the administrative distances of all dynamic routing protocols are substantially higher than 1. Therefore, by default a static route to a network will always be preferred over a dynamically discovered route to the same network.

## Case Study: Load Sharing

The problem with the configuration used in the previous section is that under normal circumstances the second link is never utilized. The bandwidth available on the link is wasted. Load sharing, also known as load balancing, allows routers to take advantage of multiple paths to the same destination by sending packets over all the available routes.

Load sharing can be equal cost or unequal cost, where cost is a generic term referring to whatever metric (if any) is associated with the route.

- Equal-cost load sharing distributes traffic equally among multiple paths with equal metrics.
- Unequal-cost load sharing distributes packets among multiple paths with different metrics. The traffic is distributed inversely proportional to the cost of the routes. That is, paths with lower costs are assigned more traffic, and paths with higher costs are assigned less traffic.

Some routing protocols support both equal-cost and unequal-cost load sharing, whereas others support only equal cost. Static routes, which have no metric, support only equal-cost load sharing.

To configure the parallel links in Figure 3.7 for load sharing using static routes, Piglet's route entries are:

```
ip route 192.168.1.0 255.255.255.0 192.168.1.193
```

```
ip route 10.4.0.0 255.255.0.0 192.168.1.193
ip route 10.1.30.0 255.255.255.0 10.1.10.1
ip route 10.1.30.0 255.255.255.0 10.1.20.1
```

and Rabbit's route entries are:

```
ip route 10.4.0.0 255.255.0.0 10.1.10.1
ip route 10.4.0.0 255.255.0.0 10.1.20.1
ip route 10.1.5.0 255.255.255.0 10.1.10.1
ip route 10.1.5.0 255.255.255.0 10.1.20.1
ip route 192.168.0.0 255.255.0.0 10.1.10.1
ip route 192.168.0.0 255.255.0.0 10.1.20.1
```

These entries were also used in the preceding section for floating static routes, except both links now use the default administrative distance of 1. Rabbit's route table, shown in Figure 3.9, now has two routes to each destination.

Load sharing is also either per destination or per packet.

### **Per Destination Load Sharing and Fast Switching**

Per destination load balancing distributes the load according to destination address. Given two paths to the same network, all packets for one destination on the network may travel over the first path, all packets for a second destination on the same network may travel over the second path, all packets for a third destination may again be sent over the first path, and so on. This type of load balancing occurs in Cisco routers when they are fast switching, the default Cisco switching mode.

Fast switching works as follows: When a router switches the first packet to a particular destination, a route table lookup is performed and an exit interface is selected. The necessary data-link information to frame the packet for the selected interface is then retrieved (from the ARP cache, for instance), and the packet is encapsulated and transmitted. The retrieved route and data-link information is then entered into a fast switching cache, and as subsequent packets to the same destination enter the router, the information in the fast cache allows the router to immediately switch the packet without performing another route table and ARP cache lookup.

While switching time and processor utilization are decreased, fast switching means that all packets to a specific destination are routed out the same interface. When a packet addressed to a different host on the same network enters the router and an alternate route exists, the router may send all packets for that destination on the alternate route. Therefore, the best the router can do is balance traffic on a per destination basis.

### **Per Packet Load Sharing and Process Switching**

Per packet load sharing means that one packet to a destination is sent over one link, the next packet to the same destination is sent over the next link, and so on, given equal-cost paths. If the paths are unequal cost, the load balancing may be one packet over the higher-cost link for every three packets over the lower-cost link, or some other proportion depending upon the ratio of costs. Cisco routers will do per packet load balancing when they are process switching.

Process switching simply means that for every packet, the router performs a route table lookup, selects an interface, and then looks up the data link information. Because each routing decision is independent for each packet, all packets to the same destination are not forced to use the same interface. To enable process switching on an interface, use the command `no ip route-cache`.

In Figure 3.10, host 192.168.1.15 has sent six pings to host 10.1.30.25. Using `debug ip packet`, the ICMP echo request and echo reply packets are observed at Piglet. Looking at the exit interfaces and the forwarding addresses, it can be observed that both Piglet and Rabbit are using S0 and S1 alternately. Note that the command `debug ip packet` allows only process switched packets to be observed. Fast switched packets are not displayed.

Like many design choices, per packet load balancing has a price. The traffic may be distributed more evenly among the various links than with per destination load balancing, but the lower switching time and processor utilization of fast switching are lost.

## Case Study: Recursive Table Lookups

All route entries do not necessarily need to point to the next-hop router. Figure 3.11 shows a simplified version of the internetwork of Figure 3.7. In this internetwork, Pooh is configured with:

```
ip route 10.1.30.0 255.255.255.0 10.1.10.2
ip route 10.1.10.0 255.255.255.0 192.168.1.194
ip route 192.168.1.192 255.255.255.224 192.168.1.66
```

If Pooh needs to send a packet to host 10.1.30.25, it will look into its route table and find that the subnet is reachable via 10.1.10.2. Because that address is not on a directly connected network, Pooh must again consult the table to find that network 10.1.10.0 is reachable via 192.168.1.194. That subnet is also not directly connected, so a third table lookup is called for. Pooh will find that 192.168.1.192 is reachable via 192.168.1.66, which is on a directly connected subnet. The packet can now be forwarded.

Because each table lookup costs processor time, under normal circumstances forcing a router to perform multiple lookups is a poor design decision. Fast switching significantly reduces these adverse effects by limiting the recursive lookups to the first packet to each destination, but a justification should still be identified before using such a design.

Figure 3.12 shows an example of an instance in which recursive lookups may be useful. Here, Sanderz reaches all networks via Heffalump. However, the network administrator plans to eliminate Heffalump and repoint all of Sanderz's routes through Woozle. The first 12 entries point not to Heffalump, but to the appropriate router attached to the 10.87.14.0 subnet. The last entry specifies

that the 10.87.14.0 subnet is reached via Heffalump.

With this configuration, all of Sanderz's entries can be repointed through Woozle simply by changing the last static entry.

Had all the static routes referenced 10.23.5.20 as the next-hop address, it would have been necessary to delete all 13 lines and type 13 new lines. Nevertheless, the effort saved in retyping static routes must be weighed carefully against the extra processing burden that recursive lookups put on the router.

## Troubleshooting Static Routes

Followers of the assorted American political scandals of the past 30 or so years will have heard a congressional investigator ask the question, "What did he know and when did he know it?" The same question serves an internetworking investigator well. When troubleshooting routing problems, the first step should almost always be to examine the route table. What does the router know? Does the router know how to reach the destination in question? Is the information in the route table accurate? Knowing how to trace a route is essential to successfully troubleshooting an internetwork.

### Case Study: Tracing a Failed Route

Figure 3.13 shows a previously configured internetwork, with each router's associated static routes. A problem has been discovered. Devices on subnet 192.168.1.0/27, connected to Pooh's Ethernet interface, can communicate with devices on subnet 10.1.0.0/16 just fine. However, when a ping is sent from Pooh itself to subnet 10.1.0.0/16, the ping fails (Figure 3.14). This seems strange. If packets being routed by Pooh successfully reach their destinations, why do packets originated by the same router fail?

Addressing this problem requires tracing the route of the ping. First, Pooh's route table is examined (Figure 3.15). The destination address of 10.1.5.1 matches the route entry for 10.0.0.0/8, which (according to the table) is reached via the next-hop address 192.168.1.34--one of Eeyore's interfaces.

Next the route table for Eeyore must be examined (Figure 3.16). The destination address 10.1.5.1 matches the entry 10.1.0.0/16, with a next-hop address of 10.4.6.1. This address is one of Tigger's interfaces.

Figure 3.17 shows Tigger's route table. The destination address matches the entry for 10.1.0.0/16 and will be forwarded to 192.168.1.194, which is at Piglet.

Piglet's route table (Figure 3.18) reveals that the target network, 10.1.0.0, is directly connected. In other words, the packet has arrived. The target address 10.1.5.1 is Piglet's own interface to that network. Because the path to the address has just been verified as good, we can assume that the ICMP echo packets from Pooh are reaching the destination.

The next step is to trace the path of the responding ICMP echo reply packets. To trace this path, you need to know the source address of the echo packet--that address will be the destination address of the echo reply packet. The source address of a packet that is originated from a router is the address of the interface from which the packet was transmitted. In this example, Pooh originally forwarded the

echo packet to 192.168.1.34. Figure 3.13 shows that the source address of that packet is 192.168.1.33. So this address is the destination address to which Piglet will send the echo reply.

Referring again to Piglet's route table in Figure 3.18, 192.168.1.33 will match the entry for 192.168.1.0/24 and be forwarded to 192.168.1.193, which is another of Tigger's interfaces. A reexamination of Tigger's route table in Figure 3.17 at first suggests that there is an entry for 192.168.1.0. However, care must be taken to correctly interpret the information that is actually there.

Compare the entries in Tigger's route table for the 10.0.0.0 subnets to those of the 192.168.1.0 subnets. The heading for the former says that 10.0.0.0 is variably subnetted; in other words, Tigger's static route to subnet 10.4.7.0 uses a 24-bit mask, and the static route to subnet 10.1.0.0 uses a 16-bit mask. The table records the correct mask at each subnet.

The heading for 192.168.1.0 is different. This heading states that Tigger knows of three subnets of 192.168.1.0 and that they all have a mask of 255.255.255.224. This mask will be used on the destination address of 192.168.1.33 to derive a destination network of 192.168.1.32/27. The route table has entries for 192.168.1.64/27, 192.168.1.0/27, and 192.168.1.192/27. There is no entry for 192.168.1.32/27, so the router does not know how to reach this subnet.

The problem, then, is that the ICMP echo reply packet is being dropped at Tigger. One solution is to create another static route entry for network 192.168.1.32, with a mask of 255.255.255.224 and pointing to a next hop of either 192.168.1.65 or 10.4.6.2. Another solution would be to change the mask in the existing static route entry for 192.168.1.0 from 255.255.255.224 to 255.255.255.0.

The moral of this story is that when you are tracing a route, you must consider the complete communication process. Verify not only that the path to a destination is good but also that the return path is good.

## Case Study: A Protocol Conflict

Figure 3.19 shows two routers connected by two Ethernet networks, one of which includes a bridge. This bridge handles traffic for several other links not shown and occasionally becomes congested. The host Milne is a mission-critical server; the network administrator is worried about traffic to Milne being delayed by the bridge, so a static host route has been added in Roo that will direct packets destined for Milne across the top Ethernet, avoiding the bridge.

This solution seems to be logical, but it isn't working. After the static route was added, packets routed through Roo no longer reach the server. Not only that, but packets routed through Kanga no longer reach the server, although no changes were made to that router.

The first step, as always, is to check the route table. Roo's route table (Figure 3.20) indicates that packets with a destination address of 172.16.20.75 will in fact be forwarded to Kanga's E1 interface, as desired. Kanga is directly connected to the destination network, so no more routing should be occurring; a quick check verifies that the Ethernet interfaces are functioning at both Kanga and at Milne.

In Figure 3.21, a trace is performed from Roo to Milne, and a symptom is found. Instead of delivering the packets to Milne, Kanga is forwarding them to Roo's E0 interface. Roo forwards the packets to Kanga's E1 interface, and Kanga sends the packets right back to Roo. A routing loop seems to have

occurred, but why?

The suspicious aspect of all of this is that Kanga should not be routing the packet, which appears to be the case. Kanga should recognize that the destination address of the packet is for its directly connected network 172.16.20.0 and should be using the data link to deliver the packet to the host. Therefore, suspicion should fall on the data link. Just as the route table should be examined to see whether the router has the correct information for reaching a network across a logical path, the ARP cache should be examined to see whether the router has the correct information for reaching a host across a physical path.

Figure 3.22 shows the ARP cache for Kanga. The IP address for Milne is in Kanga's cache as expected, with a MAC identifier of 00e0.1e58.dc39. When Milne's interface is checked, though, it shows that it has a MAC identifier of 0002.6779.0f4c; Kanga has somehow acquired incorrect information.

Another look at Kanga's ARP table reveals that the MAC identifier associated with Milne is suspiciously similar to the MAC identifier of Kanga's own Cisco interfaces (the MAC addresses with no ages associated with them are for the router's interfaces). Because Milne is not a Cisco product, the first three octets of its MAC identifier should be different from the first three octets of Kanga's MAC identifier. The only other Cisco product in the internetwork is Roo, so its ARP cache is examined (Figure 3.23); 00e0.1e58.dc39 is the MAC identifier of Roo's E0 interface.

So Kanga mistakenly believes that the E0 interface of Roo is Milne. It frames packets destined for Milne with a destination identifier of 00e0.1e58.dc39; Roo accepts the frame, reads the destination address of the enclosed packet, and routes the packet back to Kanga.

But how did Kanga get the wrong information? The answer is proxy ARP. When Kanga first receives a packet for Milne, it will ARP for that device's data link identifier. Milne responds, but Roo also hears the ARP Request on its E0 interface. Because Roo has a route to Milne on a different network from the one on which it received the ARP Request, it issues a proxy ARP in reply. Kanga receives Milne's ARP reply and enters it into the ARP cache. The proxy ARP reply from Roo arrives later because of the delay of the bridge. The original ARP cache entry is overwritten by what Kanga thinks is new information.

There are two solutions to the problem. The first is to turn off proxy ARP at Roo's E0 with the command:

```
Roo(config)#interface e0
Roo(config-if)#no ip proxy-arp
```

The second solution is to configure a static ARP entry for Milne at Kanga with the command:

```
Kanga(config)#arp 172.16.20.75 0002.6779.0f4c arpa
```

This entry will not be overwritten by any ARP reply. Figure 3.24 shows the static ARP entry being entered and the resulting ARP cache at Kanga. Note that because the entry is static, no age is associated with it.

The circumstances of the network should help determine which of the two solutions is best. A static

ARP entry means that if the network interface at Milne is ever replaced, someone must remember to change the ARP entry to reflect the new MAC identifier. On the other hand, turning off proxy ARP is a good solution only if no hosts make use of it.

## Looking Ahead

For precise control over routing behavior in an internetwork, static routing is a powerful tool. However, if topology changes are prevalent, the demands of manual reconfiguration may make static routing administratively unfeasible. Dynamic routing protocols enable an internetwork to respond quickly and automatically to topology changes. Before examining the details of specific IP routing protocols, the general issues surrounding all dynamic protocols must be examined. The next chapter introduces dynamic routing protocols.

## Summary Table: Chapter 3

### Command Review

### Review Questions

1. What information must be stored in the route table?
2. What does it mean when a route table says that an address is variably subnetted?
3. What are discontinuous subnets?
4. What command is used to examine the route table in a Cisco router?
5. What are the two bracketed numbers associated with the non-directly connected routes in the route table?
6. When static routes are configured to reference an exit interface instead of a next-hop address, in what way will the route table be different?
7. What is a summary route? In the context of static routing, how are summary routes useful?
8. What is an administrative distance?
9. What is a floating static route?
10. What is the difference between equal-cost and unequal-cost load sharing?
11. How does the switching mode at an interface affect load sharing?
12. What is a recursive table lookup?

### Configuration Exercises

1. Configure static routes for each router of the internetwork shown in Figure 3.25. Write the routes so that every subnet of the internet has an individual entry.
2. Rewrite the static routes created in Configuration Exercise 1 to use the minimum possible route entries. (Hint: RTA will have only two static route entries.)
3. For the internetwork shown in Figure 3.26, write static routes for each router. Assume that all links are identical media. Use load balancing and floating static routes for maximum efficiency and redundancy.

## Troubleshooting Exercises

1. The static route configurations for the routers in Figure 3.27 are as follows:

### RTA

```
ip route 172.20.96.0 255.255.240.0 172.20.20.1
ip route 172.20.82.0 255.255.240.0 172.20.20.1
ip route 172.20.64.0 255.255.240.0 172.20.20.1
ip route 172.20.160.0 255.255.240.0 172.20.30.255
ip route 172.20.144.0 255.255.240.0 172.20.30.255
ip route 172.20.128.0 255.255.240.0 172.20.30.255
```

### RTB

```
ip route 172.20.192.0 255.255.240.0 172.20.16.50
ip route 172.20.224.0 255.255.240.0 172.20.16.50
ip route 172.20.128.0 255.255.240.0 172.20.16.50
ip route 172.20.160.0 255.255.240.0 172.20.30.255
ip route 172.20.144.0 255.255.240.0 172.20.30.255
ip route 172.20.128.0 255.255.240.0 172.20.30.255
```

### RTC

```
ip route 172.20.192.0 255.255.240.0 172.20.16.50
ip route 172.20.208.0 255.255.255.0 172.20.16.50
ip route 172.20.224.0 255.255.240.0 172.20.16.50
ip route 172.20.96.0 255.255.240.0 172.20.20.1
```

```
ip route 172.20.82.0 255.255.240.0 172.20.20.1
```

```
ip route 172.20.64.0 255.255.240.0 172.20.20.1
```

Users are complaining of several connectivity problems in this internet. Find the mistakes in the static route configurations.

**2.** Figure 3.28 shows another internetwork in which users are complaining of connectivity problems. Figures 3.29 through 3.32 show the route tables of the four routers. Find the mistakes in the static route configurations