D.1. route

In the same way that **ifconfig** is the venerable utility for IP address management, **route** is a tremendously useful command for manipulating and displaying IP routing tables.

Here we'll look at several tasks you can perform with **route**. You can display routes, add routes (most importantly, the default route), remove routes, and examine the routing cache. I will switch between traditional and CIDR notation for network addressing in this (and subsequent) sections, so the reader unaware of these notations is encouraged to refer liberally to the links provided in Section I.1.3, "General IP Networking Resources".

When using **route** and **ip route** on the same machine, it is important to understand that not all routing table entries can be shown with **route**. The key distinction is that **route** only displays information in the main routing table. NAT routes, and routes in tables other than the main routing table must be managed and viewed separately with the **ip route** tool.

D.1.1. Displaying the routing table with route

By far the simplest and most common task one performs with **route** is viewing the routing table. On a single-homed desktop like `tristan`, the routing table will be very simple, probably comprised of only a few routes. Compare this to a complex routing table on a host with multiple interfaces and static routes to internal networks, such as `masq-gw`. It is by using the **route** command that you can determine where a packet goes when it leaves your machine.

Example D.1. Viewing a simple routing table with route

```
[root@tristan]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.99.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         192.168.99.254  0.0.0.0         UG    0      0        0 eth0
```

In the simplest routing tables, as in `tristan`'s case, you'll see three separate routes. The route which is customarily present on all machines (and which I'll not remark on after this) is the route to the loopback interface. The loopback interface is an IP interface completely local to the host itself. Most commonly, loopback is configured as a single IP address in a class A-sized network. This entire network has been set aside for use on loopback devices. The address used is usually 127.0.0.1/8, and the device name under all default installations of linux I have seen is **lo**. It is not at all unheard of for people to host services on loopback which are intended only for consumption on that machine, e.g., SMTP on tcp/25.

The remaining two lines define how `tristan` should reach any other IP address anywhere on the Internet. These two routing table entries divide the world into two different categories: a locally reachable network (192.168.99.0/24) and everything else. If an address falls within

the 192.168.99.0/24 range, `tristan` knows it can reach the IP range directly on the wire, so any packets bound for this range will be pushed out onto the local media.

If the packet falls in any other range `tristan` will consult its routing table and find no single route that matches. In this case, the default route functions as a terminal choice. If no other route matches, the packet will be forwarded to this destination address, which is usually a router to another set of networks and routers (which eventually lead to the Internet).

Viewing a complex routing table is no more difficult than viewing a simple routing table, although it can be a bit more diffiult to read, interpret, and sometimes even find the route you wish to examine.

Example D.2. Viewing a complex routing table with route

```
[root@masq-gw]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.100.0   0.0.0.0         255.255.255.252 U     0      0        0 eth3
205.254.211.0   0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.100.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.99.0    0.0.0.0         255.255.255.0   U     0      0        0 eth2
192.168.98.0    192.168.99.1    255.255.255.0   UG    0      0        0 eth2
10.38.0.0       192.168.100.1   255.255.0.0     UG    0      0        0 eth3
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         205.254.211.254 0.0.0.0         UG    0      0        0 eth1
```

The above routing table shows a more complex set of static routes than one finds on a single-homed host. By comparing the network mask of the routes above, we can see that the network mask is listed from the most specific to the least specific. Refer to Section 4.5, “Route Selection” for more discussion.

A quick glance down this routing table also provides us with a good deal of knowledge about the topology of the network. Immediately we can identify four separate Ethernet interfaces, 3 locally connected class C sized networks, and one tiny subnet (192.168.100.0/30). We can also determine that there are two networks reachable via static routes behind internal routers.

Now that we have taken a quick glance at the output from the route command, let's examine a bit more systematically what it's reporting to us.

D.1.2. Reading route's output

For this discussion refer to the network map in the appendix, and also to Example D.2, “Viewing a complex routing table with route”. **route** is a venerable command, one which can manipulate routing tables for protocols other than IP. If you wish to know what other protocols are supported, try `route --help` at your leisure. Fortunately, **route** defaults to inet (IPv4) routes if no other address family is specified.

By combining the values in columns one and three you can determine the destination network or host address. The first line in `masq-gw`'s routing table shows 192.168.100.0/255.255.255.252, which is more conveniently written in CIDR notation as

192.168.100.0/30. This is the smallest possible network according to RFC 1878. The only two useable addresses are 192.168.100.1 (`service-router`) and 192.168.100.2 (`masq-gw`).

The second column holds the IP address of the gateway to the destination if the destination is not a locally connected network. If there is a value other than 0.0.0.0 in this field, the kernel will address the outbound packet for this device (a router of some kind) rather than directly for the destination. The column after the netmask column (Flags) should always contain a **G** for destination not locally connected to the linux machine.

The fields Metric, Ref and Use are not generally used in simple or even moderately complex routing tables, however, we will discuss the Use column further in Section D.1.3, "Using **route** to display the routing cache" .

The final field in the **route** output contains the name of the interface through which the destination is reachable. This can be any interface known to the kernel which has an IP address. In Example D.2, "Viewing a complex routing table with route" we can learn immediately that 192.168.98.0/24 is reachable through interface `eth2`.

After this brief examination of the commonest of output from **route**, let's look at some of the other things we can learn from **route** and also how we can change the routing table.

D.1.3. Using route to display the routing cache

The routing cache is used by the kernel as a lookup table analogous to a quick reference card. It's faster for the kernel to refer to the cache (internally implemented as a hash table) for a recently used route than to lookup the destination address again. Routes existing in the route cache are periodically expired. If you need to clean out the routing cache entirely, you'll want to become familiar with **ip route flush cache**.

At first, it might surprise you to learn that there are no entries for locally connected networks in a routing cache. After a bit of reflection, you come to realize that there is on need to cache an IP route for a locally connected network because the machine is connected to the same Ethernet. So, any given destination has an entry in either the arp table or in the routing cache. For a clearer picture of the differences between each of the cached routse, I'd suggest adding a `-e` switch.

Example D.3. Viewing the routing cache with route

```
[root@tristan]# route -Cen
Kernel IP routing cache
Source          Destination     Gateway         Flags   MSS Window  irtt Iface
194.52.197.133  192.168.99.35   192.168.99.35     l      40 0          0 lo
192.168.99.35   194.52.197.133  192.168.99.254           1500 0         29 eth0
192.168.99.35   192.168.99.254  192.168.99.254           1500 0          0 eth0
192.168.99.254  192.168.99.35   192.168.99.35     il     40 0          0 lo
192.168.99.35   192.168.99.35   192.168.99.35     l   16436 0          0 lo
192.168.99.35   194.52.197.133  192.168.99.254           1500 0          0 eth0
192.168.99.35   192.168.99.254  192.168.99.254           1500 0          0 eth0
```

FIXME! I don't really know why there are three entries in the routing cache for each destination. Here, for example, we see three entries in the routing cache for 194.52.197.133 (a Swedish destination).

The MSS column tells us what the path MTU discovery has determined for a maximum segment size for the route to this destination. By discovering the proper segment size for a route and caching this information, we can make most efficient use of bandwidth to the destination, without incurring the overhead of packet fragmentation enroute. See Section 4.10.1, "MTU, MSS, and ICMP" for a more complete discussion of MSS and MTU.

FIXME! There has to be more we can say about the routing cache here.

D.1.4. Creating a static route with route add

Static routes are explicit routes to non-local destinations through routers or gateways which are not the default gateway. The case of the routing table on `tristan` is a classic example of the need for a static route. There are two routers in the same network, `masq-gw` and `isdn-router`. If `tristan` has packets for the 192.168.98.0/24 network, they should be routed to 192.168.99.1 (`isdn-router`). Refer also to Section 1.3.3, "Adding and removing a static route" for this example.

As with **ifconfig**, **route** has a syntax unlike most standard unix command line utilities, mixing options and arguments with less regularity. Note the mandatory `-net` or `-host` options when adding or removing any route other than the default route.

In order to add a static route to the routing table, you'll need to gather several pieces of information about the remote network.

In our example network, `masq-gw` can only reach 10.38.0.0/16 through `service-router`. Let's add a static route to the masquerading firewall to ensure that 10.38.0.0/16 is reachable. Our intended routing table will look like the routing table in Example D.2, "Viewing a complex routing table with route". Let's also view the output if we mistype the IP address of the default gateway and specify an address which is not a locally reachable address.

Example D.4. Adding a static route to a network route add

```
[root@masq-gw]# route add -net 10.38.0.0 netmask 255.255.0.0 gw 192.168.109.1
SIOCADDRT: Network is unreachable
[root@masq-gw]# route add -net 10.38.0.0 netmask 255.255.0.0 gw 192.168.100.1
```

It should be clear now that the gateway address must be reachable on a locally connected network for a static route to be useable (or even make sense). In the first line, where we mistyped, the route could not be added to the routing table because the gateway address was not a reachable address.

Now, instead of sending packets with a destination of 10.38.0.0/16 to the default gateway, `wan-gw`, `masq-gw` will send this traffic to `service-router` at IP address 192.168.100.1.

The above is a simple example of routing a network to a separate gateway, a gateway other than the default gateway. This is a common need on networks central to an operation, and less common in branch offices and remote networks.

Occasionally, however, you'll have a single machine with an IP address in a different range on the same Ethernet as some other machines. Or you might have a single machine which

is reachable via a router. Let's look at these two scenarios to see how we can create static routes to solve this routing need.

Occasionally, you may have a desire to restrict communication from one network to another by not including routes to the network. In our sample network, `tristan` may be a workstation of an employee who doesn't need to reach any machines in the branch office. Perhaps this employee needs to periodically access some data or service supplied on 192.168.98.101. We'll need to add a static route to allow this machine to access this single host IP in the branch office network [47].

Here's a summary of the required data for our static route. The destination is 192.168.98.101/32 and the gateway is 192.168.99.1.

Example D.5. Adding a static route to a host with route add

```
[root@tristan]# route add -host 192.168.98.101 gw 192.168.99.1
[root@tristan]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.98.101  192.168.99.1    255.255.255.255 UG    0      0        0 eth0
192.168.99.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         192.168.99.254  0.0.0.0         UG    0      0        0 eth0
```

Now, we have successfully altered the routing table to include a host route for the single machine we want our employee to be able to reach.

Even rarer, you may encounter a situation where a single Ethernet network is used to host multiple IP networks. There are reasons people might do this, although I regard this is bad form. If possible, it is cleaner, more secure, and easier to troubleshoot if you do not share IP networks on the same media segment. With that said, you can still convince your linux box to be a part of each network [48].

Let's assume for the sake of this example that NAT is not an option for us, and we need to move the machine 205.254.211.184 into another network. Though it violates the concept of security partitioning, we have decided to put the server into the same network as `service-router`. Naturally, we'll need to modify the routing table on `masq-gw`.

Be sure to refer to Section 9.3, “Breaking a network in two with proxy ARP” for a complete discussion of this unusual networking scenario.

Example D.6. Adding a static route to a host on the same media with route add

```
[root@masq-gw]# route add -host 205.254.211.184 dev eth3
```

I'll leave as an exercise to the reader's imagination the question of how to send all traffic to a locally connected network to an interface. In light of the host route above, it should be a logical step for the reader to make.

The above are common examples of the usage of the **route** command.

D.1.5. Creating a default route with route add default

The default route is a special case of a static route. Any machine which is connected to the Internet has a default route. For the majority of smaller networks which are not running dynamic routing protocols, each machine on an internal network uses a router or firewall as its default gateway, forwarding all traffic to that destination. Typically, this router or firewall forwards the traffic to the next router or device via a static route until the traffic reaches the ISP's service access router. Many ISPs use dynamic routing internally to determine the best path out of their networks to remote destinations.

But we are only interested in adding a default route and understanding that packets are reaching the default gateway. Once the packets have reached the default gateway, we assume that the administrator of that device is monitoring its correct operation.

With this bit of background about the default route, it is easy to see why a default route is a key part of any networking device's configuration. If the machine is to reach machines other than the machines on the local network, it must know the address of the default gateway.

Because the default gateway is so important, there is particular support for adding a default route included in the **route** command. Refer to Example 1.8, "Adding a default route with route" for a simple example of adding a default route. The syntax of the command is as follows:

Example D.7. Setting the default route with route

```
[root@tristan]# route add default gw 192.168.99.254
```

This is the commonest method used for setting a default route, although the route can also be specified by the following command. I find the alternate method more explicit than the common method for setting default gateway, because the destination address and network mask are treated exactly like any other network address and netmask.

Example D.8. An alternate method of setting the default route with route

```
[root@tristan]# route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.99.254
```

The alternate method of setting a default route specifies a network and netmask of 0, which is shorthand for all destinations. I'll reiterate that the kernel sees these two methods of setting the default route as identical. The resulting routing table is exactly the same. You may select whichever of these **route** invocations you find more comfortable.

Now that we have covered adding static routes and the special static route, the default route, let's try our hand at removing existing routes from routing tables.

D.1.6. Removing routes with route del

Any route can be removed from the routing table as easily as it can be added. The syntax of the command is exactly the same as the syntax of the **route add** command.

After we went to all of the trouble above to put our machine 205.254.211.184 into the network with `service-router`, we probably realize that from a security partitioning standpoint, it is not only stupid, but also foolhardy! So now, we conclude that we need to return 205.254.211.184 to its former network (the DMZ proper). We'll now remove the special host route for its IP, so the network route for 205.254.211.0/24 will now be used for reaching this host. (If you have questions about why, read Section 4.5, “Route Selection”.)

Example D.9. Removing a static host route with route del

```
[root@masq-gw]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
205.254.211.184 0.0.0.0         255.255.255.255 U     0      0        0 eth3
192.168.100.0   0.0.0.0         255.255.255.252 U     0      0        0 eth3
205.254.211.0   0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.100.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.99.0    0.0.0.0         255.255.255.0   U     0      0        0 eth2
192.168.98.0    192.168.99.1    255.255.255.0   UG    0      0        0 eth2
10.38.0.0       192.168.100.1   255.255.0.0     UG    0      0        0 eth3
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         205.254.211.254 0.0.0.0         UG    0      0        0 eth1
[root@masq-gw]# route del -host 205.254.211.184 dev eth3
```

Another possible example might be the prohibition of Internet traffic to a particular user. If a machine does not have a default route, but instead has a routing table populated only with routes to internal networks, then that machine can only reach IP addresses in networks to which it has a routing table entry. Let's say that you have a user who routinely spends work hours browsing the Internet, fetching mail from a POP account outside your network, and in short wastes time on the Internet. You can easily prevent this user from reaching anything except your internal networks. Naturally, this sort of a problem employee should probably face some sort of administrative sanction to address the real problem, but as a technical component of the strategy to prevent this user from wasting time on the Internet, you could remove access to the Internet from this employee's machine.

In the below example, we'll use the **route** command a number of times for different operations, all of which you should be familiar with by now.

Example D.10. Removing the default route with route del

```
[root@morgan]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.98.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         192.168.98.254  0.0.0.0         UG    0      0        0 eth0
[root@morgan]# route del default gw 192.168.98.254
[root@morgan]# route add -net 192.168.99.0 netmask 255.255.255.0 gw 192.168.98.254
[root@morgan]# route add -net 192.168.100.0 netmask 255.255.255.0 gw 192.168.98.254
[root@morgan]# route add -net 205.254.211.0 netmask 255.255.255.0 gw 192.168.98.254
[root@morgan]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
205.254.211.0   192.168.98.254  255.255.255.0   U     0      0        0 eth0
192.168.100.0   192.168.98.254  255.255.255.0   U     0      0        0 eth0
192.168.99.0    192.168.98.254  255.255.255.0   U     0      0        0 eth0
192.168.98.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
```

Now, the user on `morgan` can only reach the specified networks. The networks we have entered here are all of our corporate networks. If the user tries to generate a packet to any other destination, the kernel is not going to know where to send it, so will return in error code to the application trying to make the network connection.

While this can be a very effective way to restrict access to an individual machine, it is an ineffective method of systems administration, since it requires that the user log in to the affected machine and make changes to the routing table on demand. A better solution would be to use packet filter rules.

---

[47] Though `tristan` does not have a direct route to the 192.168.98.0/24 network, it does have a default route which knows about this destination network. Therefore, for the purposes of this illustrative example, we'll assume that `masq-gw` is configured to drop or reject all traffic to 192.168.98.0/24 from 192.168.99.0/24 and vice versa. Effectively this means that the only path to reach the branch office from the main office is via `isdn-router`.

[48] There can potentially be routing problems with multiple IP networks on the same media segment, but if you can remember that IP routing is essentially stateless, you can plan around these routing problems and solve these problems. For a fuller discussion of these issues, see Section 9.4, “Multiple IPs on an Interface” and Section 9.2, “Multiple IP Networks on one Ethernet Segment”.
