Introduction

It is essential to have an understanding of basic networking tools when administering and troubleshooting Linux servers. While some tools are made primarily for monitoring, other low-level utilities are used to configure the network connection itself and implement default settings.

Traditionally, a group of unrelated tools lumped together under the title of `net-tools` was used to do this. They were often packaged together to provide full functionality coverage, but their development and usage strategy varied from tool to tool.

Because of inconsistencies, as well as halted maintenance, a collection of tools known under the umbrella moniker `iproute2` has been used to replace these separate tools. They have been developed in tandem to share syntax and operate together efficiently.

In this guide, we will discuss how to use the iproute2 tools to configure, manipulate, and gather information about your network. We will be using an Ubuntu 12.04 VPS to demonstrate, but most modern Linux distributions should provide the same level of functionality.

While the querying commands can usually be executed as an unprivileged user, root privileges must be used to modify settings.

How To View Network Interfaces, Addresses, and Routes

One of the most fundamental responsibilities of the iproute2 suite is to manage actual interfaces.

Usually, the interfaces themselves will be named things like `eth0`, `eth1`, `lo`, etc. Traditionally, the `ifconfig` command was used to configure items in this area. Under the iproute2 system, the subcommands `ip addr` and `ip link` take care of these steps.

With ifconfig, you could gather information about the current state of your network interfaces by typing the command with no arguments:

```
ifconfig
```

```
eth0      Link encap:Ethernet  HWaddr 54:be:f7:08:c2:1b
          inet addr:192.168.56.126  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::56be:f7ff:fe08:c21b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:114273 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58866 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:73490903 (73.4 MB)  TX bytes:14294252 (14.2 MB)
          Interrupt:20 Memory:f7f00000-f7f20000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:3942 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3942 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:668121 (668.1 KB)  TX bytes:668121 (668.1 KB)
```

To get information about a single interface, you can always specify it as an argument:

<pre> ifconfig <span class="highlight">eth0</span> </pre> <pre> eth0 Link encap:Ethernet HWaddr 54:be:f7:08:c2:1b
inet addr:192.168.56.126 Bcast:192.168.56.255 Mask:255.255.255.0 inet6 addr: fe80::56be:f7ff:fe08:c21b/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:114829 errors:0 dropped:0 overruns:0 frame:0 TX packets:59007 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:73598364 (73.5 MB) TX bytes:14325245 (14.3 MB) Interrupt:20 Memory:f7f00000-f7f20000 </pre>

We can replicate this functionality with subcommands in the iproute2 suite.

To get an overview of the addresses attached to each interface, type `ip addr` in with no arguments:

```
ip addr
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 54:be:f7:08:c2:1b brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.126/24 brd 192.168.56.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::56be:f7ff:fe08:c21b/64 scope link
       valid_lft forever preferred_lft forever
```

To get a specific interface, you can use this syntax:

<pre> ip addr show <span class="highlight">eth0</span> </pre> <pre> 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000 link/ether 54:be:f7:08:c2:1b brd ff:ff:ff:ff:ff:ff inet 192.168.56.126/24 brd 192.168.56.255 scope global eth0 valid_lft forever preferred_lft forever inet6 fe80::56be:f7ff:fe08:c21b/64 scope link valid_lft forever preferred_lft forever </pre>

In fact, the `ip addr` command is just an alias for the `ip addr show` command.

If you are only concerned with the interfaces themselves and not the addresses, you can use the `ip link` command instead:

```
ip link
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 04:01:13:8a:a2:01 brd ff:ff:ff:ff:ff:ff
```

To get information about a specific interface, you'll need to add the keyword `show` followed by the interface name:

```
ip link show eth0
```

To get statistics about how an interface is communicating, you can query statistics from each interface by passing the `-s` option to the link subcommand:

```
ip -s link show eth0
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 04:01:13:8a:a2:01 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    853144     14672    0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    91257      537      0       0       0       0
```

So how do we find our routing table? The routing table contains kernel information about the paths to other network locations. We can print off the current routing table by typing:

```
ip route show
```

```
default via 107.170.58.1 dev eth0  metric 100
107.170.58.0/24 dev eth0  proto kernel  scope link  src 107.170.58.162
```

This shows us that the default route to the greater internet is available through the `eth0` interface and the address 107.170.58.1. We can access this server through that interface, where our own interface address is 107.170.58.162.

How To Configure Network Interfaces and Addresses

Now that you are familiar with how to get information about the interfaces and addresses associated with them, the next step is to find out how to modify their states.

The first step is to configure the interface itself. You can do this with the `ip link` subcommand again. This time, however, you pass the action `set` instead of show in order to modify values.

For instance, we can bring a network interface up or down by issuing these:

<pre> ip link set <span class="highlight">eth1</span> up ip link set <span class="highlight">eth1</span> down </pre>

Note: Be careful not to accidentally bring down the interface that you are connected to your server through.

You can also use the `ip link` subcommand to set attributes about the interface. For instance, if you would like to change the multicast flag on or off for your interface, you can type:

<pre> ip link set <span class="highlight">eth1</span> multicast on ip link set <span class="highlight">eth1</span> multicast off </pre>

You can adjust the mtu and package queue length like this:

<pre> ip link set <span class="highlight">eth1</span> mtu 1500 ip link set <span class="highlight">eth1</span> txqueuelen 1000 </pre>

If the interface you are configuring is down, you can adjust the interface name and the arp flag associated with the device:

<pre> ip link set <span class="highlight">eth1</span> name eth10 ip link set <span class="highlight">eth1</span> arp on </pre>

To adjust the addresses associated with the interfaces, we again use the `ip addr` subcommand.

We can add an address to a device by typing:

<pre> ip addr add <span class="highlight">ip_address/net_prefix</span> brd + dev <span class="highlight">interface</span> </pre>

The `brd +` portion of the command automatically sets the broadcast address. Multiple addresses can be added to each interface without a problem.

We can get rid of addresses with the inverse operation. To delete a specific address associated with an interface, you can use it like this:

<pre> ip addr del <span class="highlight">ip_address/net_prefix</span> dev <span class="highlight">interface</span> </pre>

Optionally, you can omit the address, and the first listed address associated with that interface will be deleted.

You can also adjust the routing of the server, using the `ip route [add | change | replace | delete ]` syntax, but we won't be covering this here, because most people will will not be adjusting this on a regular basis.

Additional Capabilities of IPRoute2

IPRoute2 has some additional capabilities that we will not be able to discuss in-depth in this guide. Instead, we will talk about what these are and what situations you may find them useful.

The idea of IP routing rules is difficult to talk about because it is very situation dependent. Basically, you can decide on how to route traffic based on a number of fields, including target address, source address, routing protocol, packet size, etc.

We access this functionality by using the `ip rule` subcommand. The basic querying follows the general pattern of the other subcommands:

```
ip rule show
```

```
0:      from all lookup local
32766:  from all lookup main
```

appropriate actions. You should not do this without knowing what you are doing however. Look at the man pages and search for `ip rule` for more information.

```
man ip         # search for "ip rule"
```

Another thing that we'll discuss briefly is the handling of arp information through these tools. The subcommand that deals with this information is called `ip neigh`.

```
ip neigh
```

```
107.170.58.1 dev eth0 lladdr 00:00:5e:00:01:68 DELAY
```

By default, this should at least list your gateway. Arp is a protocol used to gather information about physical devices accessible through the local network.

Basically, an arp request is broadcast over the local network whenever an IP address needs to be reached. The matching IP address responds and then the local computer knows where to send information to that IP address. This information is cached on the local system for some time (typically about 15 minutes) to avoid having to query during follow up communication.

Conclusion

You should now have a fairly good idea of how to use the tools included in the iproute2 suite. While many guides and tutorials still refer to the old utilities, partly because knowledgeable system admins often grew up using the older tools, the commands discussed in this guide will be taking over in the coming years.

It is important to familiarize yourself with these commands now before you find yourself troubleshooting issues on a system that has switched to these commands (Arch Linux already fully converted in 2011). In general, they are much more consistent, and you can count on certain conventions being available in all of the commands. The more you use these commands, the more they will become second nature.

<div class="author">By Justin Ellingwood</div>