Linux networking guide: Part 2

ROUTES AND GATEWAYS

In the second part of our simple guide to configuring Linux networks from the command line, Bruce Richardson shows us how to configure a Linux box as a router

Introduction

As was discussed in the first article in this series, the minimum requirements for a computer to function as part of a network are:

- A physical connection to the network, such as a Network Interface Card (NIC).
- An address on the network. It's no use being able to talk to other computers if they can't find a return address to talk back to.
- A way of determining how to reach any given address. That is to say, given an arbitrary address to connect to, can we find it on this network, can we reach it through this network or must we find some other route?

Items one and two were covered in the last article. This article looks in detail at routes, routing tables and how to manipulate them.

Before we begin

To implement the procedures outlined here you should be able to do the following:

- Install NICs in a Linux box (covered in the previous article).
- Perform basic configuration of network interfaces on your distribution of choice (also covered in the previous article).
- Compile a kernel. This is not the terrible prospect many recently-converted Linux users seem to dread. There are friendly menuing systems (text or GUI) to help you and if you haven't tried it yet you really should learn.

Networking concepts

Configuring a Linux box as a router is a more complex task than simply connecting it to a network. It is necessary to explain some elements of IP networking on Linux before giving any practical examples.

What is a subnet?

A subnet is a subdivision of a network: you might say that a subnet is to a network what a network is to an Internet.

As the previous article explained, each IP address contains a network number, uniquely identifying the network on which the host may be found. A network may be further subdivided into subnets by allocating further bits to specify subnet numbers. A network with address 194.206.0.0/8 could allocate the third most significant byte to subnet numbers: this would allow for 256 subnets (each with up to 255 hosts) with addresses 194.206.0.0/24, 194.206.1.0./24 and so on.

There may be many reasons why a network may be split into subnets, linked by switches or routers, but the greatest benefit is that dividing your network in this way makes the implementation of routing and firewalling much easier.

Routes, gateways and the routing table

When the Linux kernel is presented with an IP packet to deliver, it needs to know the route by which the packet should be sent if it is to reach its destination. To do this it consults the routing table. Each entry in the routing table lists a destination and the way to reach it. The kernel works through the entries in sequence until it finds a match. If there is no match, an error is returned. The Routing Table boxout shows a sample routing table as displayed by the route command.

The machine in question has two routes in its table. The first route is to a subnet with address 192.168.10.0 and netmask 255.255.255.0, which is the local network to which the computer is attached. The entry tells us that any IP address on that subnet can be reached through the eth0 network interface (see the Iface column at the far right).

The second route is a default route: the special address 0.0.0.0 with netmask 0.0.0.0 matches any address. This entry catches any IP address that isn't on the local network and forwards it to the host at 192.168.10.1 (as specified by the entry in the Gateway column) via network interface eth0. The machine at 192.168.10.1 is expected to know how to send the packet on at least the next step in its journey and is thus acting as a gateway to other subnets and networks.

Routing table – as shown by the *route* command

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
0.0.0.0	192.168.10.1	0.0.0.0	UG	0	0	0	eth0

The route command

The *route* command (usual location */sbin/route*) is used to manipulate the routing table. If we assume that the Linux box began with no routes at all, we can create the routing table shown in the Routing Table boxout with the following commands:

```
route add -net 192.168.10.0 netmask ➥
255.255.255.0 eth0
route add default gw 192.168.10.1
```

The second command doesn't specify an interface because the route command can work it out from the route added in the first (that is, the kernel already knows how to get to any 192.168.10.xxx address). You can specify the interface if you wish, however.

You must be root to manipulate the routing table but any user may examine it (though they may have to type the full path to the command) thus:

```
route -n
```

Which should give the same output shown in the sidebar. Note: the –n parameter isn't essential but stops the route command from attempting to resolve IP addresses into fully qualified names (which takes longer and hides the IP address information).

How the distributions do it

On a simple network you will not normally need to use the route command. For one thing, standard subnet routes like the first entry in our example are now automatically added by the kernel (any 2.2.x or later version) when the network interface is configured. As for default routes, on a Debian box a gateway entry is added to the configuration details for the associated interface (see the Debian Network Config File boxout) and the route is created and destroyed when that interface is brought up or down. On a Red Hat box the same thing can be achieved by adding an entry to */etc/sysconfig/static-routes* as shown in the Red Hat interface config script boxout.

What is a router?

All networked computers use routes and any host may be linked to more than one network but a router is a piece of equipment that connects different subnets or networks together and acts as a gateway between them, enabling hosts to on different subnets to communicate. Routers can be proprietary hardware or, as in this article, properly configured Linux boxes.

Static versus dynamic routes

Static routes do not change unless actively reconfigured. All the examples in this article use static routes. Most hardware routers are, in contrast, able to build routing tables dynamically, using established protocols to map the networks to which they are connected and responding automatically to network changes. Linux boxes can emulate this behaviour but there is not space to cover that here.

Policy routing and the IPROUTE suite

Ordinarily, routing decisions are made solely on the basis of an IP packet's destination. Policy routing allows us to set rules that route IP packets based on other criteria, such as the source address.

To do this on Linux we use the iproute suite and a properly configured kernel. A kernel configured for policy routing can maintain multiple user-defined routing tables. The IP tool from the iproute suite allows us to manipulate those tables and to create rules to specify which IP packets use which table.

The IP tool is a networking Swiss Army Knife. It can be used to replace most of the common Linux networking commands (*ifconfig*, *route* and so on) and adds a whole range of new capabilities. In this article it is used to perform functions beyond the reach of the traditional tools.

Iproute packages are available in rpm and deb format for all the main distributions and so installation is a simple one-line (or single-click) operation. There is not space here to describe the suite in any detail: the examples in the section called Building A Gateway Box should give a good introduction to the use of the IP tool.

Debian network config file

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)
# The loopback interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
         address 192.168.10.10
         netmask 255.255.255.0
         gateway 192.168.10.1
```

Figure 2: How we want our new set-up to appear

Building a gateway box

The rest of this article shows how to turn a Linux box into a gateway linking several subnets on a network.

The old setup

The network used to be organised as shown in Figure 1. There was one subnet, 192.168.10.0/24, holding all the servers and workstations. There were two routes to the Internet, one through a hardware ADSL router/firewall on 192.168.10.1 and one through a leased line whose router/firewall was on 192.168.10.2. Most user workstations were set up to use the ADSL router as a gateway while those servers which required Internet access and certain developer workstations used the leased line.

There were two main problems with this network configuration:

- 1. There is no easy way to control which hosts go out which gateway. The ADSL connection has been unreliable, sometimes through problems with the line, other times because problems with routing at the ISP. But if any machines need to switch from one gateway to the other they must either be visited individually (in the case of those which are statically configured) or wait for the change to percolate through DHCP (for those which are dynamically configured). The DHCP lease on this network is three days, meaning that a change in the configured gateway takes up to 36 hours to percolate throughout.
- 2. It is insecure. Firstly, there is only one subnet and so access must be allowed from the Internet to the internal network to reach the mail and Web servers. Secondly, having two points of egress/entry to the network makes it twice as vulnerable, with two sets of firewall rules to get right but a failure in just one causing a breach.

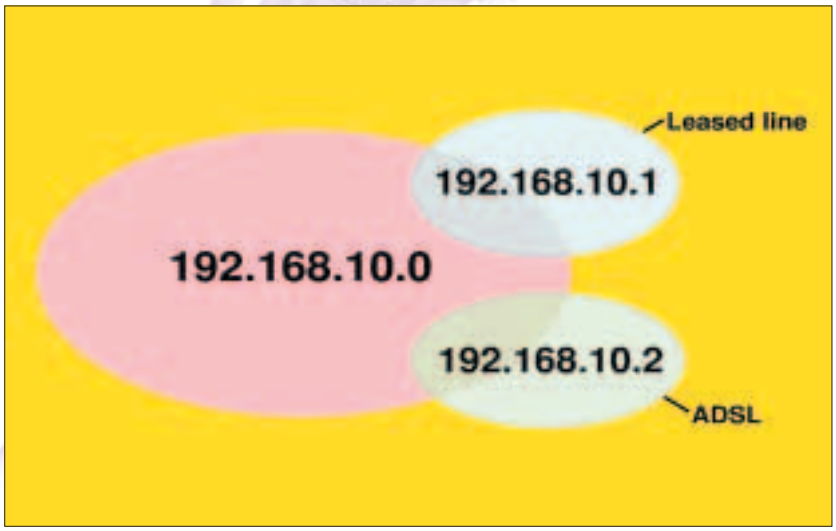

Figure 1: How our old set-up was organised

The new setup

The proposed new configuration is shown in Figure 2. There will be two extra subnets, 192.168.11.0/24, 192.168.12.0/24, the first leading to the leased line and the second to the ADSL line. The two hardware router/firewalls have been allocated IP addresses on the new subnets (192.168.11.2 and 192.168.12.2). The mail and Web servers have been moved to the leased line subnet.

The new Linux box will act as a router between all three subnets and as a firewall shielding the internal subnet (beyond the scope of this article). Its configuration will include the following key points:

- It will become the sole gateway for the 192.168.10.0/24 subnet, using policy routing rules to decide which hosts are routed out through ADSL or leased line.
- Its interface to the internal subnet will be assigned both 192.168.10.1 and 192.168.10.2

as addresses, removing the need to reconfigure any hosts on the internal subnet.
- We will add a couple of scripts to switch the default route between the ADSL and leased line, so that users can quickly be routed through the leased line if there are problems on the ADSL.

This configuration offers several advantages over the old one:

- It's simpler to administer, since all routing decisions are made at the gateway box.
- Users can be switched instantly from routing through the ADSL line to routing through the leased line.
- It's much more secure, since the Internet-facing servers have been moved into a Demilitarized Zone (DMZ) between the Internet and the internal subnet. Now access from the Internet can be restricted to the DMZ, with no access allowed from the Internet to the internal subnet.

The implementation

Prepare a Linux box with three network cards in it (a Pentium II or equivalent with 64Mb RAM and fast Ethernet cards will be more than sufficient). You will need a 2.2.x or later kernel and the iproute suite.

Make sure the kernel has been compiled to be an advanced router (CONFIG_IP_ADVANCED_ROUTER=y) , with the policy routing (CONFIG_IP_MULTIPLE_TABLES= y), Netlink socket (CONFIG_NETLINK=y) and routing messages (CONFIG_RTNETLINK=y) options. It would also be desirable have it optimised for routing (CONFIG_IP_ ROUTER=y). Additional options are required for firewalling but that is beyond the scope of this article.

Configure the network interfaces so that eth0, eth1 and eth2 have IP addresses 192.168.10.1, 192.168.11.1 and 192.168.12.1 respectively. The Debian Gateway Interfaces boxout shows how this would be done on a Debian box.

Simpler steps

First we should switch on IP forwarding, so that the box will forward packets that come in on one interface back out on the appropriate interface:

Debian gateway interfaces

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)
# The loopback interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
         address 192.168.10.1
         netmask 255.255.255.0
auto eth1
iface eth1 inet static
         address 192.168.11.1
         netmask 255.255.255.0
auto eth2
iface eth2 inet static
         address 192.168.12.1
         netmask 255.255.255.0
```

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

The routes to each subnet should have been configured along with their respective interfaces but now we add a default route out through the ADSL line:

```
route add default gw 192.168.12.2 eth2
```

At this point the routing table should look like the Gateway Routing Table boxout.

Now we turn from the route tool to the IP tool, adding an extra address to the eth0 interface so that the gateway box can impersonate both of the routers:

```
ip addr add 192.168.10.2/24 brd + dev eth0
```

Policy routing implementation

At this point we do have a functioning router. If the cables are connected it will route from the internal subnet to the Internet through the ADSL line (via eth2 and to the mail and Web servers via eth1). Now for the complicated bit where we try to add an alternative set of routes for certain hosts.

First we add a custom routing table, leasedline, to the rt_tables file (on a Debian system this is in */etc/iproute2/*):

Red Hat interface config script

```
# /etc/sysconfig/static-routes                    # matching line
# Each line  should have the format:              # is passed to route as:
#                                                 #
#    device args                                  #    route add -args device
#
# When an interface is brought up, each           eth0 default  gw 192.168.10.1
```

Gateway routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.11.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1
192.168.12.0	0.0.0.0	255.255.255.0	U	0	0	0	eth2
0.0.0.0	192.168.12.2	0.0.0.0	UG	0	0	0	eth2

Route switching scripts

```
#!/bin/sh
# /usr/local/bin/goleased – switches default
route to leased line
route del default gw 192.168.12.2
route add default gw 192.168.11.2
ip route flush cache
#!/bin/sh
# /usr/local/bin/goadsl - switches default
route to ADSL line
route del default gw 192.168.11.2
route add default gw 192.168.12.2
ip route flush cache
```

```
/etc/iproute2/rt_tables
# syntax: priority name
#
255     local
254     main
253     default
0       unspec
#
# table added for leased line
#
200     leasedline
```

The new table has no routes, so we need to add some:

```
ip route add 192.168.10.0/24 dev eth0 table➥
leasedline
ip route add 192.168.11.0/24 dev eth1 table➥
leasedline
ip route add 192.168.12.0/24 dev eth2 table➥
leasedline
ip route add default via 192.168.11.2 dev➥
eth1 table leasedline
```

Now we add a rule for each ip address of a host that will use the new table:

```
ip rule add from 192.168.10.14 table➥
leasedline
ip rule add from 192.168.10.17 table➥
leasedline
ip rule add from 192.168.10.32 table➥
leasedline
```

We can check to see if the routes and rules have been properly configured:

```
ip route show table leasedline
ip rule show
```

Finally, we flush the cached list of routes so that the new routing tables are used:

```
ip route flush cache
```

If everything worked, traffic to the Internet from the three hosts listed above will go through the leased line, no matter what the state of the main routing table.

Final detail

Now all we need do is add scripts that can be used to switch the default route from the ADSL to the leased line and vice versa. A simple example is provided in the Route Switching Scripts boxout. Note: these scripts do not affect the leased line routing table.

Making it permanent

If you have all this working, it may occur to you that most of the configuration will vanish if you restart the machine or bring network interfaces down. A crude way to give it permanence would be to place the sequence of commands listed above into a script, to be run during start-up. A more robust solution would be to arrange it so that routes and rules are added or deleted with their associated interfaces. Each distribution has its own way of achieving this and I leave it to you to research as an educative project.

Summary

This article has shown you how routing works on Linux, from the simple set-up of a typical workstation to the complex configuration of a router. Most small networks don't need anything so complex but the ambitious scope of this article hopefully provides you with examples that you can adapt to your own needs.

The next article will explain DNS, how to configure workstations to use DNS properly and how to configure a DNS server using BIND or some of the smaller Open Source DNS daemons.

Info

iproute site
http://defiant.coinet.com/iproute2/
Advanced ruting HOWTO
http://www.linuxdoc.org/HOWTO/Adv-Routing-HOWTO.html
Linux network administrators guide
http://www.tldp.org/LDP/nag2/index.html