How to get your packets from point a to point b

Routing with Solaris

Lance Spitzner

Last Modified: 9 March, 2000

This article is the second in a two part series. In the first article I discussed how to configure, modify and troubleshoot network interface cards. This second article will discuss routing issues for systems with two or more network interface cards. I will not be discussing gated nor any routing protocols, such as RIP or OSPF. This article will focus only on implementing static routing tables. Throughout this article I will be using the octet method for denoting subnet masks, as opposed to the more modern method of using a /. Example, I will be designating a class C network as 255.255.255.0, as opposed to /24. I decided to use the older notation as this is what Solaris uses. If you have any questions about IP addressing or subnetting, I highly recommend you first review IP Subnetting Tutorial.

Routing

Routing has always fascinated me, it is amazing how a system knows where to forward a packet. In this article, I will discuss how Solaris 2.6 knows just that, where to forward a packet. The first part of this article I will explain and setup some basic static routes. In the second part of the article I will cover some of Solaris 2.6's more advanced capabilities, specifically VLSM (Variable Length Subnet Masks).

Routing is the process of forwarding a packet from point A to point B. Solaris does this by building a routing table. When it forwards a packet, it first refers to the routing table to decided where to send the packet. The key to successful routing with Solaris is building a proper routing table. You start building your routing table with your first network interface device.

When you configure a network interface device, the kernel automatically builds a static routing table. For example, lets say you are on a system that has a single interface, elx0. You configure the device to have an IP address of 207.229.165.133, with a netmask of 255.255.255.0. To see the routing table the kernel has built, use the command netstat -nr.

```
Routing Table:
Destination Gateway Flags Ref Use Interface
----------------- -------------------- ----- ----- ------ ---------
207.229.165.0 207.229.165.133 U 1 20 elx0
127.0.0.1 127.0.0.1 UH 0 94 lo0
```

Here we see the system's routing table. The first column is *Destination*, which network does the packet want to go to. The second column is *Gateway*, the IP address the packet must go to get to the destination (the next hop). The third column is *Flags*, which denotes interface information, such as U for up, and G for Gateway. The fourth column is *Ref*, which denotes how many times that specific MAC address is referenced in the routing table. The fifth is *Use*, or how many packets have gone through the interface. The last column is *Interface*, it show the device the packet must go through if the destination is the local network.

In the example above we have two routes. The bottom one, 127.0.0.1, is the standard loopback route. All systems have this route, the kernel uses it to talk to itself. The second entry is a result of the elx0 interface. This entry says if you need to get to a node on the 207.229.165.0 network, go to 207.229.165.133, which is the system interface. This entry is called the local network entry and is added by default. The kernel assumed that because elx0 has an IP address of 207.229.165.133 and a netmask of 255.255.255.0, it must be connected to the local network 207.229.165.0. Thus, if you want to talk to any node on the 207.229.165.0 network, the kernel knows exactly where to send the packet.

Any time you add a new interface, the kernel adds a routing entry similar to the one above. It assumes that the new interface can talk to the local network. Your system can now talk to the local network, 207.229.165.0. But what about other networks, such as the Internet? If you were to attempt to talk to any node on any other network, such as 206.54.252.8, you would get the following error.

```
lisa #ping 206.54.252.8
ICMP Net Unreachable from gateway lisa (207.229.165.133)
for icmp from lisa (207.229.165.133)
```

The system has no idea how to reach this node. To fix this, we need to give the system a default route. When the kernel is given a destination it does not know, it sends it to the default route. The default route is usually the IP address of another router. This router takes the packet and does one of two things with it. If the destination is local to the router, it sends the packet to the local destination. If not, the router sends the packet upstream to another router. This process repeats itself until the packet has reached its destination.

By default, Solaris uses a routing protocol to dynamically determine the default route, RIP or Route Discovery. During the init process, the /etc/rc2.d/S69inet will attempt to find a router running route discovery (/usr/sbin/in.rdisc). If this fails after three attempts, the script then launches /usr/bin/in.routed, otherwise known as RIP. However, we will use neither method. Instead, we are going to manually set the default route. When the default route is manually set, neither routing protocol is initiated. The advantages to this are a simpler and more secure system to administer.

You manually define the default route with the file /etc/defaultrouter. This file consists of a single entry, the IP address of the default router. This file is read during the init process (specifically /etc/rc2.d/S69inet) and added to the routing table. For this system, we have identified the default router as 207.229.165.1. This is the IP address of the router that connects us Internet. If our system does not know a packet's destination, it sends the packet to the default router. With our default route, the new routing table would looks as follows

```
Routing Table:
Destination Gateway Flags Ref Use Interface
-------------------- -------------------- ----- ----- ------ ---------
207.229.165.0 207.229.165.133 U 2 20 elx0
default 207.229.165.1 UG 0 20
127.0.0.1 127.0.0.1 UH 0 30 lo0
```

Based on this table, the system now has two choices for forwarding a packet. If the destination is on the 207.229.165.0 network, the packet is sent to the local network. However, if the destination is any other network, then the packet is sent to the default router. Notice that the default router, 207.229.165.1, is on the local network. If the default route is not on the local network, the system cannot reach the default router.

To define the netmasks of your network, you use the file /etc/netmasks. This file is read during the bootup process, defining the makeup of your networks. Here is an example of a /etc/netmasks file.

```
#
# The netmasks file associates Internet Protocol (IP) address
# masks with IP network numbers.
#
# network-number netmask
#
# The term network-number refers to a number obtained from the Internet Network
# Information Center. Currently this number is restricted to being a class
# A, B, or C network number. In the future we should be able to support
# arbitrary network numbers per the Classless Internet Domain Routing
# guidelines.
#
# Both the network-number and the netmasks are specified in
# "decimal dot" notation, e.g:
#
# 128.32.0.0 255.255.255.0
#
207.229.165.133 255.255.255.0
172.16.1.0 255.255.255.0
192.168.1.0 255.255.255.240
```

This file defines the network 207.229.165.0 as a standard class C network. However, the file also defines 172.16.1.0 as a class C network, even though it is normally a class B network. Last, we see the network 192.168.1.0 broken down even smaller as a 16 IP subnet. We discuss subnetting later in the article.

IP Forwarding

Up to this point we have been discussing singled homed system. Single homed systems have one of two choices, talk to the local network,

or to the default router. Things get more complicated when you add a second interface. Your system now becomes multi-homed, and potentially a gateway. A multi-homed host is any system with two or more interfaces, usually on different networks. A gateway is any multi-homed system that routes packets between different networks.

Lets take a look at what happens when we add a second interface. We add the interface elx1 to our system, with an IP address of 10.1.6.1, netmask 255.255.255.0.

```
Routing Table:
Destination Gateway Flags Ref Use Interface
-------------------- -------------------- ----- ----- ------ ---------
207.229.165.0 207.229.165.133 U 2 20 elx0
10.1.6.0 10.1.6.1 U 1 123 elx1
default 207.229.165.1 UG 0 20
127.0.0.1 127.0.0.1 UH 0 30 lo0
```

Looking at the routing table, you notice only one change, the addition of interface elx1. If a packet is destined for any node on the 10.1.6.0 network, the packet is forward out the elx1 interface.

However, there is another, and far more important change not seen here, IP forwarding has just been enabled on this machine. Basically, IP forwarding means the system will route packets between networks. Based on the table above, the gateway will forward a packet one of two ways. If it does not know the destination, it will forward the packet to the default router. If the destination is on one of the two local networks, then the packet will be forwarded to its destination.

IP forwarding is enabled during the init process, in /etc/rc2.d/S69inet. If the system detects more then 2 interfaces (including the loopback) ip forwarding will be enabled by default. Your system is now a gateway.

You can have a system with 2 or more interfaces and not forward packets if you want. This is done one of two ways. First, by touching the file /etc/notrouter. During the init process, /etc/rc2.d/S69inet will look for this file. If it finds it, it turns off ip forwarding by executing the following command.

```
ndd -set /dev/ip ip_forwarding 0
```

You can manually turn off ip forwarding any time by executing the same command.

Route Command

The route command allows you to manually change the route table. You can add, delete or change routes in real time. For example, lets say the IP address of the default router has changed, but you cannot afford to reboot the system. You have to change the IP address of the default route without rebooting. You do this with the route command. The syntax is simple,

```
lisa #route change default 207.229.165.5 1
```

This command changes the default router from 207.229.165.1 to a .5. The syntax is simple, type the network information as you want it to appear in the routing table. The last number at the end of the command is the metric, or how many hops to the next gateway. Any node, including a router, on the local network is a hop of 1. The route add command allows you to add additional routes to the routing table, just as route delete removes them. If you want to make a route command permanent, add the command to the bottom of /etc/rc2.d/S69inet. The init script will execute the route command and update the routing table.

VLSM

Starting with 2.6, Solaris supports VLSM (Variable Length Subnet Mask). VLSM means a network can be variably subnetted into smaller networks, each smaller network having a different subnet mask. What that means to you is that life just got a lot easier.

Under Solaris 2.5.1 or earlier, you could only define a single subnet for a network. For example, if you defined the network 10.1.6.0 with a

255.255.255.0 subnet mask as we have done, older versions of Solaris would assume that any network starting with 10 was a 255.255.255.0 subnet mask. You have now locked yourself in. You had to manually add an individual route for any 10.0.0.0 network that did not have this subnet. This could easily reach into the hundreds!

VLSM does not make this assumption, it gives you flexibility in setting up your routing tables. You can have as many different subnets as you want for a network. Lets take a look at an example. Our current routing table (see example from above) is configured for two local networks and a single default route for everything else (the Internet). However, this system is to be the gateway for a large corporation, the company's firewall. This means all inbound and outbound traffic must go through it.

The corporation is made up of an internal 10.0.0.0 network, which is subnetted into over 100 smaller networks. Each smaller, subnetted network has a different subnetmask. For example

10.15.146.0 255.255.254.0 (510 hosts)
10.128.112.0 255.255.248.0 (2046 hosts)
10.220.160.0 255.255.240.0 (4094 hosts)

Here you see the company's various, different networks. We have to create a routing table that routes all default traffic to the Internet, but at the same time routes anything on the 10.0.0.0 internally. Remember, our internal network is really over a hundred smaller 10.0.0.0 networks, all variably subnetted.

First, we have to identify the internal router. In our case, we will use 10.1.6.5. This is the IP address of the router on the internal network. Notice how this router is on the local network of interface elx1. Now, since we are using Solaris 2.6, which support VLSM, we need only 1 command to route all the variably subnetted 10.0.0.0 networks

```
lisa #route add net 10.0.0.0 10.1.6.5 1
```

With this single command, we have taken care of all routing issues, something possible only with VLSM. Lets take a look at the routing table and explain what I mean.

```
Routing Table:
Destination Gateway Flags Ref Use Interface
-------------------- -------------------- ----- ----- ------ ---------
207.229.165.0 207.229.165.133 U 2 20 elx0
10.1.6.0 10.1.6.1 U 2 123 elx1
10.0.0.0 10.1.6.5 U 0 0
default 207.229.165.1 UG 0 20
127.0.0.1 127.0.0.1 UH 0 30 lo0
```

The first line states that if your destination is on the 207.229.165.0 network, the network is local to the interface elx0. If your destination is on the 10.1.6.0 network, the network is local to the interface elx1. If your destination is on any 10.0.0.0 network EXCEPT to 10.1.6.0, the packet is forwarded to the gateway 10.1.6.5. Last, if the destination meets none of the above criteria (the Internet), the packet is forwarded to the default router, 207.229.165.1.

You may be confused as to how does the system know where to forward anything on 10.1.6.0. By looking at the routing table, you see two entries that would work, one for 10.1.6.0 and one for 10.0.0.0, both work for 10.1.6.0. The system always selects the most specific path first.

Now, if this was on a system that did not support VLSM, such as 2.5.1, things would be MUCH uglier. As stated earlier, the 10.0.0.0 is variably subnetted into smaller networks. Without VLSM, you would have to manually add a static route for each separate network with the route add command. If you do not, the kernel will assume that all the 10.0.0.0 networks are subnetted the same, causing all sorts of interesting routes. As you can see, VLSM is extremely powerful.

CIDR

I decided to discuss CIDR, as it is easy to confuse with VLSM and they are both closely related. Defined in 1993 by rfc 1519, Classless Inter-Domain Routing is used for routing aggregation, also known as supernetting. Simple stated, this means lumping several networks

into one. The purpose is to reduce routing tables, which are beginning to overload backbone routers. An example would be taking 256 class C networks and defining them as a single network, aggregating them together. You can define the networks 207.229.0.0 − 207.229.255.0 with the single routing entry of 207.229.0.0 subnet mask of 255.255.0.0.

CIDR aggregates several networks together for simpler routing, compared to VLSM which variably subnets a network into smaller networks. Confused? Don't feel bad, so is half the Internet. To learn more, I highly recommend you read 3Com's Whitepaper on IP addressing, VLSM, and CIDR.

Conclusion

The key to successful routing is your routing tables. By defining a proper routing table, your packets will get from point A to point B. VLSM is a standard that allows greater flexibility and in developing a proper routing table. Remember, a happy gateway is a happy network.

Downloads

Figuring out subnet masks and CIDR aggregation can be quite challenging. However, for the brain dead like me there exists an AWESOME little tool called IP Calculator from Net3 Group Inc. This freely distributed Windows tool is a great way to solve all your subnetting problems. I HIGHLY recommend it.

ip_calculator.zip 603 KB

For you Unix weenies, there is a Unix version of this tool, located at http://www.interloper.net/~dan/software

Author's bio

Lance Spitzner enjoys learning by blowing up his Unix systems at home. Before this, he was an Officer in the Rapid Deployment Force, where he blew up things of a different nature. You can reach him at lance@honeynet.org .

Whitepapers / Publications