Network in a box

Demo/Workshop/Learning Module

Crossbow: Network Virtualization & Resource Control
http://www.opensolaris.org/os/project/crossbow

Sunay.Tripathi@Sun.Com

Objective

Create a real network comprising of Hosts, Switches and Routers as a Virtual Network on a laptop. The Virtual Network is created using OpenSolaris project Crossbow Technology and the hosts etc are created using Solaris Zones (a light weight virtualization technology). All the steps necessary to create the virtual topology are explained.

The users can use this hands on demo/workshop and exercises in the end to become an expert in

- Configuring IPv4 and IPv6 networks
- Hands on experience with OpenSolaris
- Configure and manage a real Router
- IP Routing technologies including RIP, OSPF and BGP
- Debugging configuration and connectivity issues
- Network performance and bottleneck Analysis

The users of this module need not have access to a real network, router and switches. All they need is a laptop or desktop running OpenSolaris Project Crossbow snapshot 2/28/2008 or later which can be found at http://www.opensolaris.org/os/project/crossbow/snapshots.

Introduction

Crossbow (Network Virtualization and Resource Control) allows users to create a Virtual Network with fixed link speeds in a box. Multiple subnet connected via a Virtual Router is pretty easy to configure. This allows the network administrators to do a full network configuration, verify IP address, subnet masks and router ports and addresses. They can test connectivity and link speeds and when fully satisfied, they can instantiate the configuration on the real network.

Another big application is to debug problems by simulating a real network in a box. If network administrators are having issues with connectivity or performance, they can create a virtual network and debug their issues using snoop, kernel stats and dtrace. They don't need to use the expensive H/W
based network analyzers.

The network developers and researchers working with protocols (like high speed TCP) can use OpenSolaris to write their implementation and then try it out with other production implementations. They can debug and fine tune their protocol quite a bit before sending even a single packet on the real
network.

Note1: Users can use Solaris Zones, Xen or ldom guests to create the virtual hosts while Crossbow provides the virtual network building blocks.

Note2: The Solaris protocol code executed for a virtual network or Solaris acting a real router or host is common all the way to bottom of MAC layer. In case of virtual networks, the device driver code for a physical NIC is the only code that is not needed.

Try it Yourself

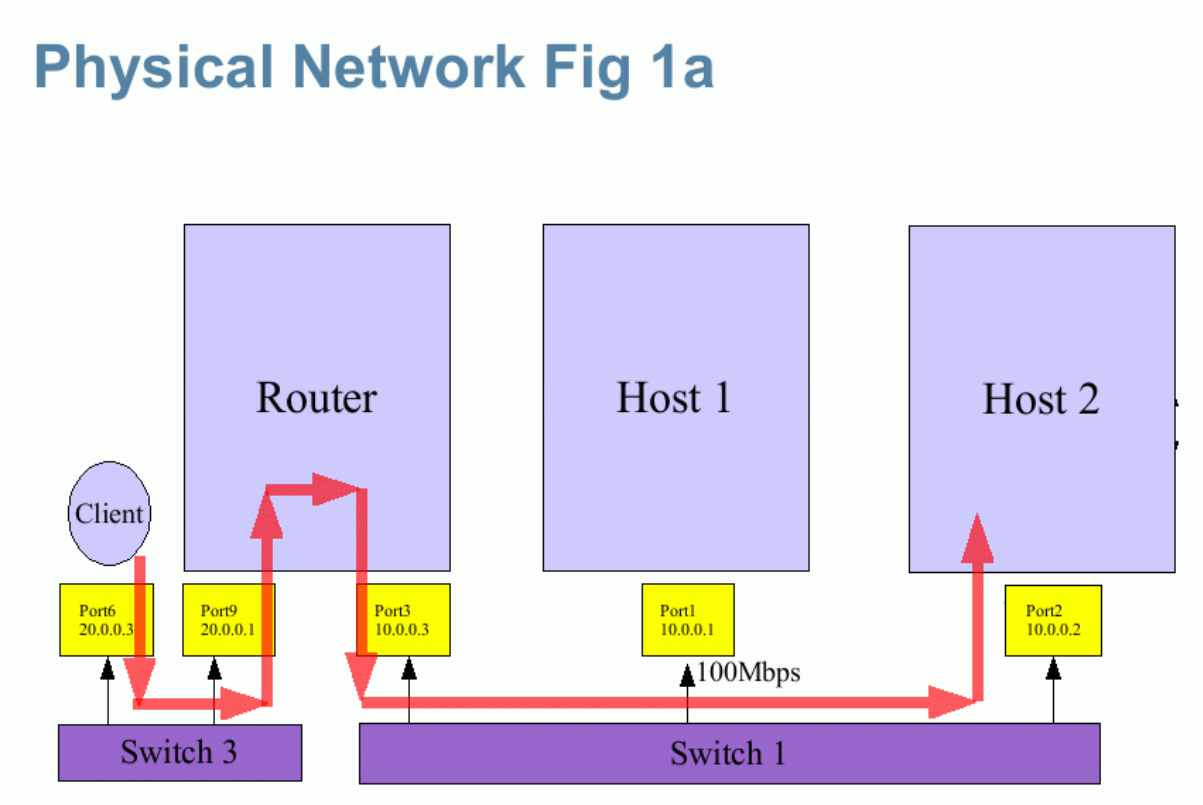
Lets do a simple exercise. As part of this exercise, you will learn

- How to configure a virtual network having two subnets and connected via a Virtual Router using Crossbow and Zones
- How to set the various link speeds to simulate multiple speed network
- Do some performance runs to verify connectivity

What you need

A laptop or machine running Crossbow snapshot from Feb 28, 2008 or later
http://www.opensolaris.org/os/project/crossbow/snapshots/

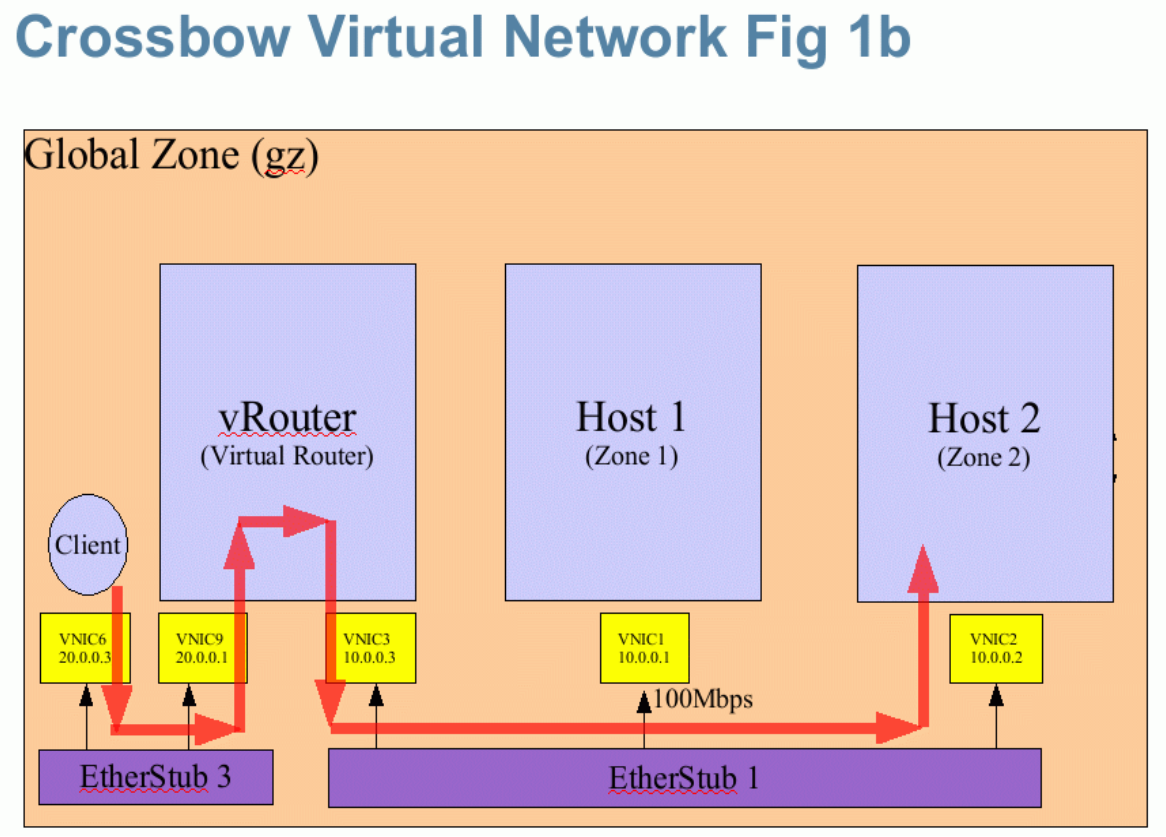
Virtual Network Example



Lets take a physical network. The example in Fig 1a is representing the real network showing how my desktop connects to the Lab servers. The desktop is on 20.0.0.0/24 network while the server machines (host1 and host2) are on 10.0.0.0/24 network. In addition, host1 has got a 10/100 Mbps NIC limiting its connectivity to 100Mbps.

We will represent the network shown in Fig 1a on my Crossbow enabled laptop as a Virtual Network. We use Zones to act as host1, host2 and the Router while the global zone (gz) acts as the client (as a user exercise, create another client zone and assign VNIC6 to it to act as a client).

Note 3: The Crossbow MAC layer itself does the switching between the VNICs. The Etherstub is craeated as a dummy device to connect the various virtual NICs. User can imagine etherstub as a Virtual Switch to help visualize the virtual network as a replacement for a physical network where
each physical switch is replaced by a virtual switch (implemented by a Crossbow etherstub).



Create the Virtual Network

Lets start by creating the 2 etherstubs using the dladm command

```
gz# dladm create-etherstub 1
gz# dladm create-etherstub 3
gz# dladm show-etherstub
LINK
etherstub1
etherstub3
```

Create the necessary Virtual NICs. VNIC1 has a limited speed of 100Mbs while others have no limit

```
gz# dladm create-vnic -d etherstub1 1
gz# dladm create-vnic -d etherstub1 2
gz# dladm create-vnic -d etherstub1 3
```

```
gz# dladm create-vnic -d etherstub3 6
gz# dladm create-vnic -d etherstub3 9
gz# dladm show-vnic
LINK        OVER              SPEED  MACADDRESS
MACADDRTYPE
vnic1       etherstub1      - Mbps  2:8:20:8d:de:b1    random
vnic2       etherstub1      - Mbps  2:8:20:4a:b0:f1    random
vnic3       etherstub1      - Mbps  2:8:20:46:14:52    random
vnic6       etherstub3      - Mbps  2:8:20:bf:13:2f    random
vnic9       etherstub3      - Mbps  2:8:20:ed:1:45     random
```

Create the hosts and assign them the VNICs. Also create the Virtual Router and assign it VNIC3 and VNIC9 over etherstub1 and etherstub3 respectively. Both the Virtual Router and Hosts are created using Zones in this example but you can easily use Xen or logical domains.

Create a base Zone which we can clone.

```
gz# zonecfg -z vnmbase
vnmbase: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:vnmbase> create
zonecfg:vnmbase> set zonepath=/vnm/vnmbase
zonecfg:vnmbase> set ip-type=exclusive
zonecfg:vnmbase> add inherit-pkg-dir
zonecfg:vnmbase:inherit-pkg-dir> set dir=/opt
zonecfg:vnmbase:inherit-pkg-dir> set dir=/etc/crypto
zonecfg:vnmbase:inherit-pkg-dir> end
zonecfg:vnmbase> verify
zonecfg:vnmbase> commit
zonecfg:vnmbase> exit
```

This part takes 15-20 minutes

```
gz# zoneadm -z vnmbase install
```

Now lets create the 2 hosts and the Virtual Router as follow

```
gz# zonecfg -z host1
host1: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:vnmbase> create
zonecfg:vnmbase> set zonepath=/vnm/host1
```

```
zonecfg:vnmbase> set ip-type=exclusive
zonecfg:vnmbase> add inherit-pkg-dir
zonecfg:vnmbase:inherit-pkg-dir> set dir=/opt
zonecfg:vnmbase:inherit-pkg-dir> set dir=/etc/crypto
zonecfg:vnmbase:inherit-pkg-dir> end
zonecfg:vnmbase> add net
zonecfg:vnmbase:net> add physical=vnic1
zonecfg:vnmbase:net> end
zonecfg:vnmbase> verify
zonecfg:vnmbase> commit
zonecfg:vnmbase> exit

gz# zoneadm -z host1 clone vnmbase
gz# zoneadm -z host1 boot

gz# zlogin -C host1
```

Connect to the console and go through the sysid config. For this example, we assign 10.0.0.1/24 as IP address for vnic1. You can specify this during sysidcfg. For default route, specify 10.0.0.3 as the default route. You can say 'none' for naming service, IPv6, kerberos etc for the purpose of this example.

Similarly create host2 and configure it with vnic2 i.e.

```
gz# zonecfg -z host2
host2: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:vnmbase> create
zonecfg:vnmbase> set zonepath=/vnm/host2
zonecfg:vnmbase> set ip-type=exclusive
zonecfg:vnmbase> add inherit-pkg-dir
zonecfg:vnmbase:inherit-pkg-dir> set dir=/opt
zonecfg:vnmbase:inherit-pkg-dir> set dir=/etc/crypto
zonecfg:vnmbase:inherit-pkg-dir> end
zonecfg:vnmbase> add net
zonecfg:vnmbase:net> add physical=vnic2
zonecfg:vnmbase:net> end
zonecfg:vnmbase> verify
zonecfg:vnmbase> commit
zonecfg:vnmbase> exit

gz# zoneadm -z host2 clone vnmbase
gz# zoneadm -z host2 boot
```

```
gz# zlogin -C host2
```

Connect to the console and go through the sysid config. For this example, we assign 10.0.0.2/24 as IP address for vnic2. You can specify this during sysidcfg. For default route, specify 10.0.0.3 as the default route. You can say 'none' for naming service, IPv6, kerberos etc for the purpose of this example.

Lets now create the Virtual Router as

```
gz# zonecfg -z vRouter
vRouter: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:vnmbase> create
zonecfg:vnmbase> set zonepath=/vnm/vRouter
zonecfg:vnmbase> set ip-type=exclusive
zonecfg:vnmbase> add inherit-pkg-dir
zonecfg:vnmbase:inherit-pkg-dir> set dir=/opt
zonecfg:vnmbase:inherit-pkg-dir> set dir=/etc/crypto
zonecfg:vnmbase:inherit-pkg-dir> end
zonecfg:vnmbase> add net
zonecfg:vnmbase:net> add physical=vnic3
zonecfg:vnmbase:net> end
zonecfg:vnmbase> add net
zonecfg:vnmbase:net> add physical=vnic9
zonecfg:vnmbase:net> end
zonecfg:vnmbase> verify
zonecfg:vnmbase> commit
zonecfg:vnmbase> exit

gz# zoneadm -z vRouter clone vnmbase
gz# zoneadm -z vRouter boot

gz# zlogin -C vRouter
```

Connect to the console and go through the sysid config. For this example, we assign 10.0.0.3/24 as IP address for vnic3 and 20.0.0.1/24 as the IP address for vnic9. You can specify this during sysidcfg. For default route, specify 'none' as the default route. You can say 'none' for naming service, IPv6,
kerberos etc for the purpose of this example. Lets enable forwarding on the Virtual Router to connect the 10.x.x.x and 20.x.x.x networks.

```
vRouter# svcadm enable network/ipv4-forwarding:default
```

Note 5: The above is done inside virtual router. Make sure you are in the window where you did the zlogin -C vRouter above

Now lets bringup VNIC6 and configure it including setting up routes in the global zone. You can easily create another host called host3 as the client on 20.x.x.x network by creating a host3 zone and assigning it 20.0.0.1/24 IP address

Lets configure the VNIC6. Open a xterm in the global zone

```
gz# ifconfig vnic6 plumb 20.0.0.3/24 up
gz# route add 10.0.0.0 20.0.0.1
gz# ping 10.0.0.1
10.0.0.1 is alive
gz# ping 10.0.0.2
10.0.0.2 is alive
```

Similarly, login into host1 and/or host2 and verify connectivity

```
host1# ping 20.0.0.3
20.0.0.3 is alive
host1# ping 10.0.0.2
10.0.0.2 is alive
```

Set up Link Speed

What we configured above are unlimited B/W links. We can configure a link speed on all the links. For this example, lets configure the link speed of 100Mbps on VNIC1

```
gz# dladm set-linkprop -p maxbw=100 vnic1
```

We could have configured the link speed (or B/W limit) while we were creating the vnic itself by adding the "`-p maxbw=100`" option to create-vnic command.

Test the performance

Start '*netserver*' (or tool of your choice) in host1 and host2. You wil have to install the tools in the relevant places

```
host1# /opt/tools/netserver &
```

```
host2# /opt/tools/netserver &

gz# /opt/tools/netperf -H 10.0.0.2
TCP STREAM TEST to 10.0.0.2 : histogram
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 49152  49152  49152    10.00    2089.87

gz# /opt/tools/netperf -H 10.0.0.1
TCP STREAM TEST to 10.0.0.1 : histogram
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 49152  49152  49152    10.00     98.78
```

Note6: Since 10.0.0.2 is assigned to VNIC2 which has no limit, we get the max speed possible. 10.0.0.1 is configured over VNIC1 which is assigned to host1 and we just set the link speed to 100Mbps and thats why we get only 98.78Mbps.

Cleanup

```
gz# zoneadm -z host1 halt
gz# zoneadm -z host1 uninstall

delete the zone

gz# zonecfg -z host1
zonecfg:host1> delete
Are you sure you want to delete zone host1 (y/[n])? y
zonecfg:host1> exit
```

In this way, delete host2 and vRouter zones. Make sure you don't delete vnmbase since re creating it takes time.

```
gz# ifconfig vnic6 unplumb
```

After you have deleted the zone, you can delete vnics and etherstubs as follows

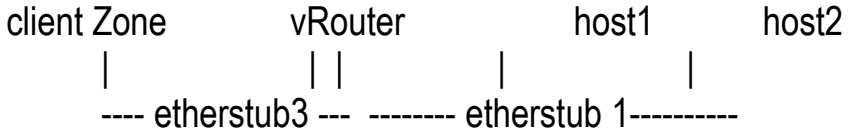
```
# dladm delete-vnic 1           /* Delete VNIC */
# dladm delete-vnic 2
# dladm delete-vnic 3
# dladm delete-vnic 6
# dladm delete-vnic 9

# dladm delete-etherstub 3      /* Delete etherstub */
# dladm delete-etherstub 1
```

Make sure that VNICs are unplumbed (ifconfig vnic6 unplumb) and not assigned to a zone (delete the zone first) before you can delete them. You need to delete all the vnics on the etherstub before you can delete the etherstub.

User Exercises

Now that you are familiar with the concepts and technology, you are ready to do some experiments of your own. Cleanup the machine as mentioned above. The exercises below will help you master IP routing, configuring networks, and debugging for performance bottlenecks.

- 1. Recreate the Virtual Networkwork as show in Fig 1b but this time create an additional zone called client and assigned vnic6 to that client zone.

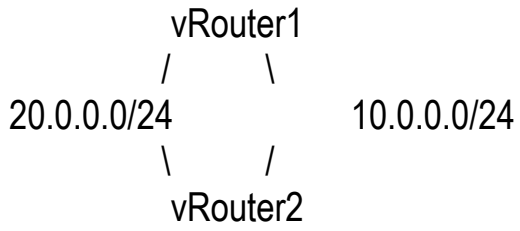
  Run all your connectivity tests from zloging into the client. Now change all IPv4 addresses to be IPv6 addresses and verify that client and hosts still have connectivity

- 2. Leave the Virtual Network as in 1, but configure OSPF in vRouter instead of RIP by default. Verify that you can still get the connectivity. Note the steps needed to configure OSPF

- 3. Configure 20.0.0.0 and 10.0.0.0 networks as two separate autonomous networks, assign them unique ASN numbers and configure unique BGP domains. Verify that connectivity still works. Note the steps needed to configure BGP domains.

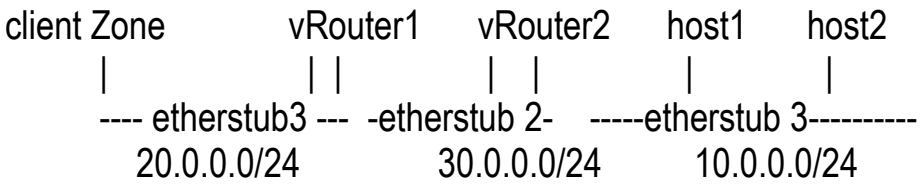
- 4. Cleanup everything and recreate the virtual network in 1 above but instead of statically assigning the IP addresses to hosts and clients, configure NAT on the vRouter to give out address on subnet 10.0.0.0/24 on vnic3 and address on 20.0.0.0/24 for vnic9. While creating the hosts and clients, configure them to get their IP address through DHCP.

- 5. Cleanup everything and recreate the virtual network in 1 above. Add additional router vRouter2 which has a vnic each on the 2 etherstubs.



  This provides a redundant path from client to the hosts. Experiment with running different routing protocols and assign different weight to each path and see what path you take from client to host (use traceroute to detect). Now configure the routing protocol on two vRouters to be OSPF and play with link speeds and see how the path changes. Note the configuration and observations.

- 6. Cleanup. Lets now introduce another Virtual Router between two subnets i.e.



  Now set the link (VNIC) between vRouter1 and etherstub2 to be 75 Mbps. Use snmp from client to retrive the stats from the vRouter1 and check where the packets are getting dropped when you run netperf from client to host2.

  Remove the limit set earlier and instead set the link speed of 75 Mbps on link between etherstub2 and vRouter2. Again use snmp to get the stats out on vRouter1. Do you see similar results as vRouter1? If not, can you explain why?

Conclusion & More resources

Use the real example and configure the virtual network to get familiar with the techniques used. At this point, have a look at your network and try to create a virtual network.

Get more details on the OpenSolaris Crossbow page
http://www.opensolaris.org/os/project/crossbow

You can find high level presentations, architectural documents, man pages etc at
http://www.opensolaris.org/os/project/crossbow/Docs

Join the crossbow-discuss@opensolaris.org mailing list at
http://www.opensolaris.org/os/project/crossbow/discussions

Send in your questions or your configuration samples and we will put it in the use cases examples.

A similar Virtual Network example using global zone as a NAT can be found on Nicolas's blog at http://blogs.sun.com/droux

Kais has a a example of dynamic bandwidth paritioning at
http://blogs.sun.com/kais

Venu talks about some of the cool crossbow features at
http://blogs.sun.com/iyer which allows virtualizing services with Crossbow technology using flowadm.

A copy of this document can be found on my blog
http://blogs.sun.com/sunay