Introduction

In our laboratory project, we use four Ubuntu 8.04 Linux Routers with the open-source package Quagga and two Cisco 2851 routers for Cisco-Linux integration skills for our students. Figure Lab-1 illustrates it.

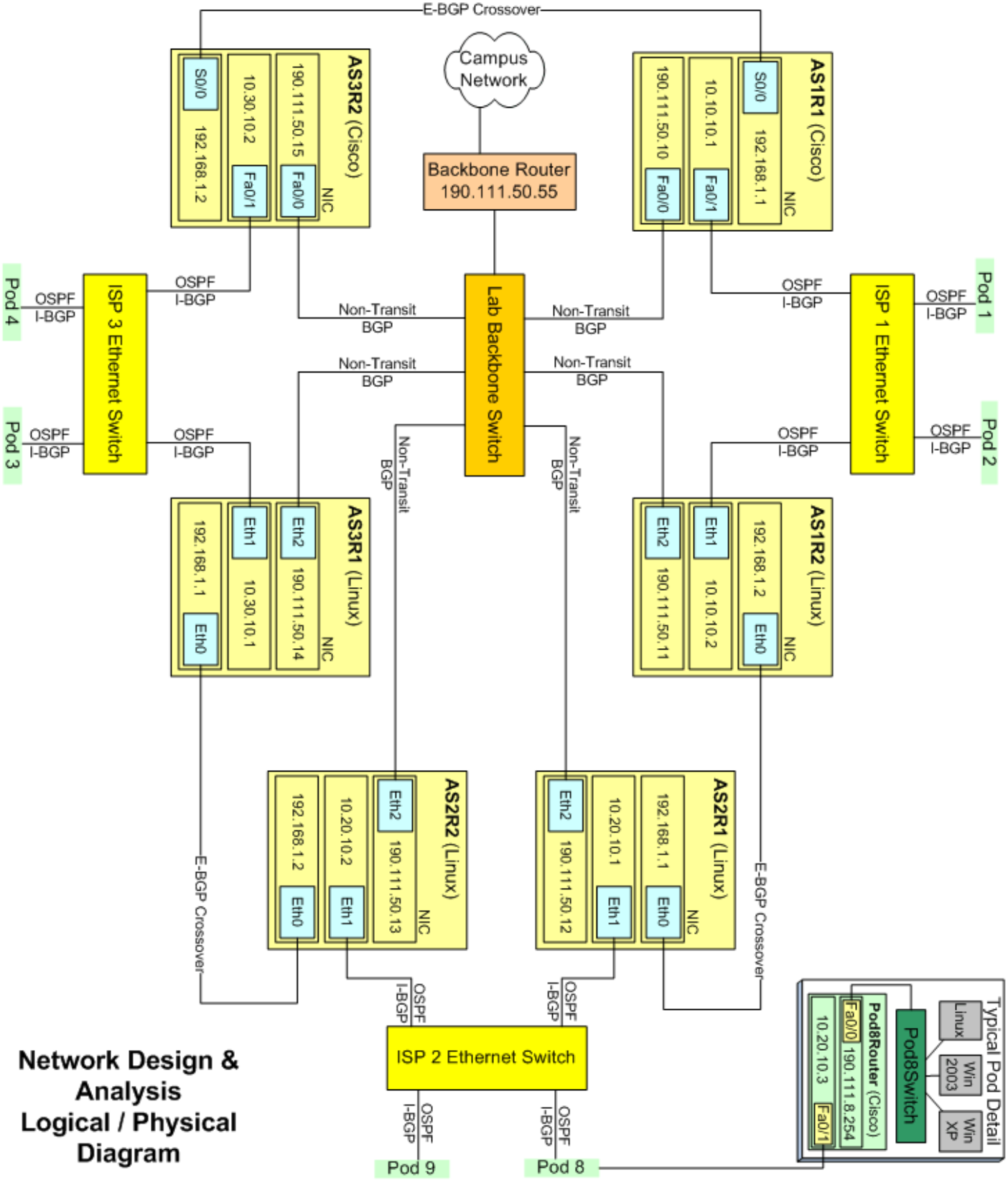

Figure Lab-1. The Physical/Logical Diagram for the Lab Project

You may well ask why only two Cisco routers and four Linux routers? The answer is simple: we would have to have purchased a T1 PCI NIC for one of the Linux routers, or have a third Ethernet on one of our Cisco routers to make the lab project a success. We chose two Cisco routers instead since each of our Cisco 2851 routers has two Ethernet and one T1 WIC interfaces. The T1 interfaces include an integrated CSU/DSU capability.

As you can see by the "E-BGP crossover" designation between the AS1 and AS3 Cisco routers, the interfaces are marked "S0/3/0". These are Serial interfaces with capabilities for fractional T1. We employ a **T1 crossover cable** for this purpose, and no doubt the other question you might have is "What exactly is a T1 Crossover cable?" You have heard of Ethernet crossover cables. Indeed, for back-to-back T1 interfaces a similar crossover is required. Our T1 interfaces come with a RJ45/48 physical form factor. An appendix to this book contains a description of the necessary steps to wire a T1 crossover.

In our laboratory arrangement at Radford University, we implement this lab in our "Network Design & Analysis" course, ITEC 451. We use the concept of a "pod" to represent subscribers to the ISPs. In our situation at Radford University, we have another course "Introduction to Computer Networking" that implements a basic small business network, with a Cisco router, switch and three operating systems as client and server computers: Linux, Windows 2003 Server and Windows XP. A pod is the collection of the router, switch and operating systems. This represents subscribers to the ISPs, and we reconfigure the pod router for OSPF and i-BGP.

Finally, for the laboratory setup it is necessary to have at least four Linux routers running Quagga 0.99.15 or better (www.quagga.net). We have performed this laboratory in the past with 6 Linux routers. It is not necessary to use Fedora Core 3. If you have other Linux distributions that are preferred, (Red Hat, Debian, Slackware, etc.) any of those should be successful. We have used Red Hat 9 and Fedora Core 3 in several past semesters.

To represent subscribers, it should be noted that we recommend at least 3 pods, one for each autonomous system. This brings the minimum number of pod routers to 3, if 6 Linux routers are used for the autonomous systems.

Many other arrangements are possible, but our notion of a triangular arrangement of autonomous systems has the benefit of teaching a complex interaction of each autonomous system with the other. Note that each Autonomous System (AS) has two routers. With proper configuration, one router can be shut down while the other takes over the routing. This arrangement is used by many service providers. Indeed several commercial ISPs use Quagga as the software for their systems. It is time to prepare you for the laboratory setup beginning with Linux.

Linux Router Setup

Determining NIC-to-Interface mappings

Before attempting to configure the network interfaces on the Linux routers it is helpful to determine which logical interface name corresponds to which physical port. The Linux kernel assigns the names of the interfaces (e.g. eth0) to the network cards in the order in which the cards are detected. That is, eth0 is assigned to the first Network Interface Card (NIC) that the kernel detects, eth1, to the second, etc. However, this order can vary depending on the hardware setup and the kernel version and can't be relied upon to remain constant. The surest way to cable the lab is to connect each NIC individually to the network and use the **ifconfig** command to see which interface detects a link. When an interface has a link, **ifconfig** shows the word "RUNNING" in the status flag line. Note that it might take up to 30 seconds for it to appear.

For this lab we will assume all Linux systems assign the interfaces in the order shown in Figure Lab-2. However, other arrangements are possible.

Since this lab requires Linux routers using Quagga, and Cisco routers, each Linux or Cisco router must have at least three interfaces. Most PCs come with an Integrated Network Interface Card. In many cases, the integrated NIC will be assigned to either the first or the last interface and the remaining PCI NICs will be assigned in the same order as they are positioned in the machine. The most secure method of detecting Ethernet assignments is to use the method described above, plugging an Ethernet cable from the switch into the NIC and observing the **ifconfig** command and its results.

Once the order of the interfaces has been established it is useful to view a logical view of the network we are constructing. This is shown in Figure Lab-4, below. A logical diagram has the advantage of abstracting the physical connections.

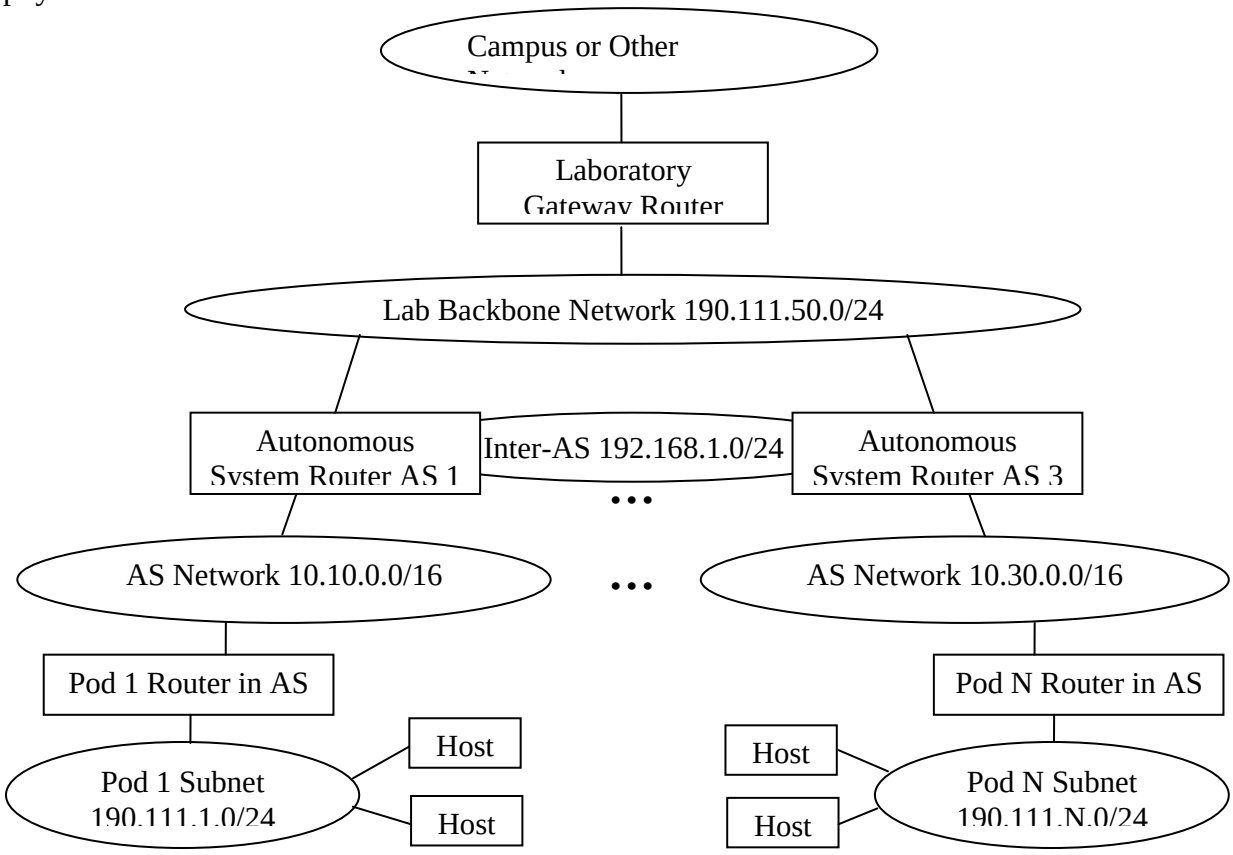

Figure Lab-4. Logical Diagram.

Configuring network settings

Once Ubuntu has booted up go to ***System->Administration->Network*** to configure the network settings.

The hostname and DNS options are under the General and DNS tabs of the Network settings window. The hostname of each Linux router should be set to the name given for that router on the lab diagram, such as AS1R2. The Primary DNS server should be set to the IP address of the lab server (**190.111.50.100**). You may have to press the "Unlock" button on the Network Settings window in order to make any changes. You will also have to uncheck the "Enable roaming mode" option when assigning an IP address to an interface.

Each network interface (eth0, eth1, eth2) must be configured with a statically assigned IP address and subnet mask. As shown in the diagram, eth0 is a point-to-point Ethernet crossover link to the neighboring AS router, eth1 is connected to the AS network, and eth2 connects to the lab backbone. Interface **eth0** should have an IP address of **192.168.1.<1 or 2>** and a subnet mask of **255.255.255.0**. Interface **eth1** is should have an IP address of **10.<AS# * 10>.10.<1 or 2>** and a subnet mask of **255.255.0.0**. Note that it is very important that the subnet mask be set correctly. **OSPF will not form neighbor relationships if subnet masks are mismatched**. Interface **eth2** should have an IP address of **190.111.50.<10-15>** , a subnet mask of **255.255.255.0, and a default gateway of 190.111.50.55**. Once networking is configured, it is best to restart the machine. You can verify connective by trying to connect to the internet.

Note that the 190.111.0.0 IP addresses, if you choose to use them, are class B address and Linux, Windows, and Cisco will all assume a 16-bit mask. You must override this with a 24-bit mask. They will also assume an 8-bit mask for the 10.0.0.0 addresses.

Enabling IP forwarding

By default, the Linux kernel does not route IP packets between network interfaces. This situation is similar to Cisco routers where the command **ip routing** must be used to allow the router to pass packets between interfaces. To manually enable forwarding, use the command:

```
sudo gedit /etc/sysctl.conf
Uncomment "net.ipv4.ip-forward=1
Remember to save sysct1.conf
Restart the computer
```

To display the current forwarding status (1=enabled, 0=disabled) use:

```
cat /proc/sys/net/ipv4/ip_forward
Should return 1
```

Installing and Configuring Quagga

Although most Linux distributions include a Quagga package, it is not usually installed unless a full or "everything" install was done. Even with a full install it is possible that the most recent Quagga package was not installed. On Fedora systems, the yum command can be used to install or upgrade the Quagga package. More detailed installation information can be found in the Appendix. Enter the following command to install or upgrade Quagga:

```
sudo apt-get install quagga
```

Quagga has a separate service for each routing protocol. Information from the routing protocol services is coordinated by the zebra server. Before these servers will run, an initial configuration file must be created. The configuration files for the Quagga services are stored in **/etc/quagga**. You will need to create the files

zebra.conf, **ospfd.conf**, and **bgpd.conf**. Each file should contain a minimal configuration to set the hostname and passwords. The hostname can be set with **hostname <hostname>** where <hostname> should be the server's name in the form <router name>-<service name>. The console and privileged mode passwords are set with **password <password>** and **enable password <password>** where <password> should be of the form as#as#. For example:

```
hostname AS2R1-ospfd
password as2as2
enable password as2as2
```

To prevent telnet sessions to the servers from disconnecting after a period of inactivity, the following lines may also be added to each configuration file:

```
line vty
 exec-timeout 0 0
```

After the files are created, the Quagga servers can be activated by editing the daemons config file. The config file should look line the lines below.

```
Use the command below to edit the daemons config file.
sudo gedit /etc/quagga/daemons
```

Edit the config file so that file looks like the lines below.

```
zebra=yes
bgpd=yes
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no

Save the daemons file
Restart the Linux router.
```

Once running, you can access each server with **telnet localhost <port>** where <port> is replaced with **zebra**, **ripd**, **ospfd**, or **bgpd**. Note that these names are actually aliases for the port numbers 2601, 2602, 2604, and 2605 respectively. A shorter form of this command is **telnet 0 <port>**.

Note well that there is an option to use IS-IS instead of OSPF in the material at the end of this Lab Guide. The reasoning for using IS-IS is that it is preferred by large ISPs for scalability. Unfortunately, it is necessary to compile Quagga 0.99.15 or greater (+isisd) for such purposes. See the appendix for instructions on compiling Quagga.

Configuring OSPF

OSPF will be used as the Interior Gateway Protocol (IGP) for each AS, that is, it will exchange routing information only between the routers inside of the AS. OSPF works by electing the routers with the highest priority on each subnet to be the Designated Router (DR) and Backup Designated Router (BDR). The DR is responsible for coordinating the exchange of routing information on the subnet. Each router on the subnet must form a neighbor relationship (called an adjacency) with the DR before routing information can be exchanged. Note that the DR will not form an adjacency with a router if their subnet masks do not match.

To start an OSPF process, use **router ospf**. By default, OSPF will select the highest active IP address on the router to use as the router ID. The router ID is used in all OSPF communications to identify the router. If the interface that corresponds to the router ID should go down or the router ID should change, information in the OSPF database can become unstable. To prevent the router ID from changing, manually set it using **router-id 10.<AS# * 10>.10.<1 or 2>**.

The only subnet on which OSPF should run is the AS network (the network between the pod and AS routers), which in this case is the 10.x.0.0/16 network. The command **network <network address>/<prefix length> area 0** is used to tell OSPF on which network interface to run. It also tells OSPF to advertise that network to its neighbors. Note that OSPF is capable of dividing a large network into multiple areas, but since only one OSPF area is needed, the network must be added to area 0 (the OSPF backbone area).

The AS routers will also use OSPF to propagate their default route (created above) back to the pod routers using the **default-information-originate** command. Since part of the goal of this lab is for the pod routers to display two default routes learned via OSPF, the metric on the propagated default routes will be manually configured to the same value. If this is not done, OSPF will calculate the metric based on the bandwidth of the interface, which may not be equal if the two AS routers do not have interfaces of the same speed. If the metrics are not equal, OSPF will use only the route with the lowest metric while the second route will be treated as a backup and will not be displayed in the routing table.

The following commands will start OSPF on AS3R1:

```
AS3R1-ospfd>enable
AS3R1-ospfd# configure terminal
AS3R1-ospfd (config-router)# router ospf 1
AS3R1-ospfd (config-router)# router-id 10.30.10.1
AS3R1-ospfd (config-router)# network 10.30.0.0/16 area 0
AS3R1-ospfd (config-router)# default-information originate metric 12
```

Configuring eBGP and iBGP

BGP will be used to exchange routing information between autonomous systems and to distribute that information to the routes within the AS. Although there is only one BGP routing protocol, BGP makes a distinction between exchanging information with a router that is external to the AS (eBGP) and one that is internal (iBGP). A router that is on the border between two autonomous systems will use eBGP to exchange routes with the neighboring AS. If an AS has multiple border routers connected to other autonomous systems, then those routers will communicate with each other using iBGP. Other internal routers can also run iBGP to learn routes to external networks (although on many networks, border routers will redistribute BGP routes into the IGP instead).

The command to start a BGP process is **router bgp <AS #>** where <AS #> is the AS to which the router belongs. This number must be set correctly for BGP to determine which of its neighbors are internal to the AS and which are external. Similar to OSPF, the BGP routing process will use the highest active IP address as the router ID. To prevent possible router ID conflicts, use **bgp router-id 190.111.50.<10-15>**. Also use, **bgp network import-change** to detect downed interfaces (see the troubleshooting section for more information). To display changes in neighbor status, also include **bgp log-neighbor-changes**.

As with other routing protocols, BGP will only advertise routes to networks specified with the command **network <network address>/<prefix length>**. Only the AS network (10.x.0.0/16) should be advertised by BGP. Note that none of the routers should ever advertise the 192.168.1.0 network since that same network is

used between each AS and is not a uniquely identifiable destination. Also, the backbone network should not be advertised since its sole function is to provide Internet access to each AS.

Unlike OSPF and other IGPs, BGP does not automatically exchange routes with every other router on the subnet. Every BGP neighbor's IP address and AS number must be manually configured with **neighbor <neighbor address> remote-as <neighbor AS #>**. Each AS router should have a neighbor relationship to every other router within the AS and to the eBGP peer connected by the crossover cable. For each neighbor, also use **neighbor <neighbor address> next-hop-self** to replace the non-routable 192.168.1.(1/2) next hop address in advertisements (see the troubleshooting section).

The following configuration example was taken from the bgpd.conf file of AS3R1:

```
router bgp 3
 bgp router-id 190.111.50.14
 bgp log-neighbor-changes
 bgp network import-check
 network 10.30.0.0/16
 ! iBGP configuration
 neighbor 10.30.10.2 remote-as 3
 neighbor 10.30.10.2 next-hop-self
 neighbor 10.30.10.3 remote-as 3
 neighbor 10.30.10.3 next-hop-self
 neighbor 10.30.10.4 remote-as 3
 neighbor 10.30.10.4 next-hop-self
 ! eBGP configuration
 neighbor 192.168.1.2 remote-as 2
```

In addition to the BGP configuration needed to exchange routes between each AS, each AS router may also need to form an eBGP neighbor relationship with the lab backbone router to permit access to the Internet. There are several possible methods to provide routing information to the lab backbone router but a non-transit BGP AS is the preferred configuration. For further information about routing over the backbone or configuring alternative routing methods, see the Backbone Router Setup section. Otherwise, to establish an eBGP adjacency with the backbone router, issue the **neighbor 190.111.50.55 remote-as 50** command in the **router bgp <AS#>** configuration mode of each AS router.

Cisco Router Setup

Basic configuration

Every Cisco router should have a minimal configuration that sets its hostname, passwords, and IP addresses. The hostname should be the corresponding router name on the diagram. The privileged mode password can be set with **enable secret <password>** where password should be of the form as#as#.

Each Cisco AS router has two Ethernet interfaces and a Serial (T1) interface. Interface GigabitEthernet0/0 (ge0/0) connects to the lab backbone and interface GigabitEthernet 0/1 (ge0/1) connects to the AS network. The IP address can be set by entering interface config mode with **interface <interface name>** and using the **ip address <address> <subnet mask>** command. By default, all interfaces are in the down state and must be activated with **no shutdown**.

FastEthernet0/0 should have an IP address of **190.111.50.<10-15>** and netmask of **255.255.255.0**. Note the class B network address and the 24-bit subnet mask. FastEthernet0/1 should have **10.<AS# * 10>.10.<1 or 2>** and a mask of **255.255.0.0**, a class A network with a 16-bit mask.

The following example was taken from AS1R1:

```
hostname AS1R1
enable secret as#as#
! Interface to Lab Backbone
interface GigabitEthernet 0/0
 ip address 190.111.50.10 255.255.255.0
 no shutdown
! Interface to AS1 internal network
interface GigabitEthernet 0/1
 ip address 10.10.10.1 255.255.0.0
 no shutdown
```

Although not required, telnet access can be activated and idle timeouts removed by issuing the following commands in config mode:

```
line con 0
 password as#as#
 login
 exec-timeout 0 0
line vty 0 4
 password as#as#
 login
 exec-timeout 0 0
```

T1 crossover interface

In this lab, the crossover connection between the two Cisco routers (AS1R1 and AS3R2) is made using T1 CSU/DSU WAN Interface Cards (WICs). Although the T1 WIC uses a standard RJ-45/48 connector, an Ethernet crossover cable will not work. See the appendix at the back of the book for instructions on creating the necessary cable.

Once cabled, the T1 serial interfaces will require additional configuration. One of the routers must be configured to produce a clock signal. This can be done with the interface command **service module t1 clock source internal**. The other router can be made to receive the clock signal from the line by using **service module t1 clock source line**. In addition to the clock signal, the T1 signaling information should be set. Note that the signal configuration commands used in this lab will match the default settings on many routers and may not appear in the running-config after being entered. The following is the configuration used on AS1R1:

```
interface Serial 0/3/0
 ip address 192.168.1.1 255.255.255.0
 encapsulation ppp
 service-module t1 clock source internal
 service-module t1 linecode b8zs
 service-module t1 framing esf
 service-module t1 data-coding normal
 service-module t1 timeslots all
 no shutdown
```

Static default routes

Each AS router should have a static default route to the lab backbone router. This route will handle traffic sent to the Internet. A default route has the same function on a router that setting a default gateway has on a host. The following command will create the default route:

```
Router (config)# ip route 0.0.0.0 0.0.0.0 190.111.50.55
```

Configuring OSPF

As with the Linux routers, OSPF will handle routes originating from within the AS. The commands to configure OSPF on a Cisco router are almost identical to Quagga, with only a few exceptions.

Cisco routers support multiple instances of OSPF and distinguish between each instance using a process ID. The process ID is only used locally and does not need to be unique across routers. It is very common to use 1 as the process ID on every router.

The network command on Cisco routers use what is known as a wildcard mask in place of the more familiar subnet mask or prefix length. The wildcard mask can be thought of as an inverse subnet mask. For example, if the subnet mask is 255.255.0.0 then the wildcard mask would be 0.0.255.255. The following example will enable OSPF on an AS1 router:

```
Router (config)# router ospf 1
Router (config-router)# router-id 10.10.10.1
Router (config-router)# log-adjacency-changes
Router (config-router)# network 10.10.0.0 0.0.255.255 area 0
Router (config-router)# default-information originate metric 12
```

In addition to the above commands, the pod routers will also need to have their OSPF priority set to 0. This will prevent the pod routers from becoming Designated Routers. Since the DR is responsible for coordinating routing updates between all OSPF routers on the subnet, this role should be handled by the AS routers since they are more centrally located. Note that the priority must be set on the interface attached to the network between the AS routers and pod routers (10.x.0.0/16). For example:

```
Router (config)# interface GigabitEthernet 0/1
Router (config-if)# ip ospf priority 0
```

Configuring iBGP and eBGP

The BGP configuration used on a Cisco router is very much like that of the Quagga router. One important difference is the **no synchronization** command. This command tells BGP that the networks it will advertise to its neighbors do not need to be in the IGP (OSPF in this case). On some of the latest versions of the IOS, this command may be the default, but for many Cisco routers it will not be.

Normally, BGP would not advertise a route that is not present in OSPF. Because iBGP neighbors do not have to be on the same subnet, it is possible to have two iBGP neighbors that are several hops away from each other. That is, there could be non-BGP routers between any two iBGP neighbors. If a packet needed to go from one iBGP neighbor to the other to be sent to a network that is outside of the AS, the intermediate non-BGP routers would have to know how to route a packet destined for another AS. The only way the non-BGP routers could know about networks in other ASes is if BGP routes were redistributed into OSPF.

To ensure that there is definitely a path between iBGP neighbors, Cisco BGP normally will not advertise a route it learned from an iBGP neighbor until it sees that route in the routing table from a non-BGP routing

protocol. However, since this lab configuration runs BGP on every router within the AS, every router will learn external routes via iBGP and there is no need to redistribute that same information into OSPF. Since iBGP routes will not be redistributed into OSPF, synchronization must be disabled or no routes learned via iBGP will be advertised to eBGP neighbors.

Cisco BGP also requires the **no auto-summary** command to prevent subnets from being automatically summarized into a single classful network route. Also, the BGP network statement uses the **mask <subnet mask>** keyword instead of a prefix-length.

The following configuration example was taken from AS1R1:

```
router bgp 1
 no synchronization
 bgp router-id 190.111.50.10
 bgp log-neighbor-changes
 network 10.10.0.0 mask 255.255.0.0
 ! iBGP configuration
 neighbor 10.10.10.2 remote-as 1
 neighbor 10.10.10.2 next-hop-self
 neighbor 10.10.10.3 remote-as 1
 neighbor 10.10.10.3 next-hop-self
 neighbor 10.10.10.4 remote-as 1
 neighbor 10.10.10.4 next-hop-self
 ! eBGP configuration
 neighbor 192.168.1.2 remote-as 3
 !
 no auto-summary
```

As with the Linux routers, the Cisco AS routers must also be configured to provide routing information to the backbone router to permit access to the Internet. If using the preferred non-transit BGP AS configuration, type the **neighbor 190.111.50.55 remote-as 50** command in the **router bgp <AS #>** configuration mode. For alternative backbone routing configurations see the Backbone Router Setup section.

Pod Router Configuration Example

The pod router configuration is similar to the Cisco AS routers with a few exceptions. The GigabitEthernet0/0 interface connects to the pod network and should have an IP address of **190.111.<pod #>.254** and mask of **255.255.255.0** and the GigabitEthernet 0/1 connects to the AS network with an address of **10.<AS# * 10>.10.<3 or 4>** and a mask of **255.255.0.0**.

Only the AS routers should have a static default route. The pod routers will learn their default route dynamically through OSPF and should not have a static default. If **show ip route** on a pod router has an entry that begins with "S* 0.0.0.0/0" then that route should be removed with the command **no ip route 0.0.0.0 0.0.0.0**.

Both the OSPF and BGP routing processes should have two network statements, one for the pod network and one for the AS network. Also notice that the BGP process has a neighbor statement for each other router on the AS network but does not have an eBGP neighbor. Note that the GigabitEthernet 0/1 interface must have its OSPF priority set to 0 using **ip ospf priority 0** to ensure that the pod routers will not become the OSPF DR for the subnet.

This is the configuration for AS1 pod1 router**:**

```
hostname pod1router
!
enable secret as#as#
!
interface GigabitEthernet0/0
 ip address 190.111.1.254 255.255.255.0
 no shutdown
!
interface GigabitEthernet0/1
 ip address 10.10.10.3 255.255.0.0
 ip ospf priority 0
 no shutdown
!
router ospf 1
 log-adjacency-changes
 network 10.10.0.0  0.0.255.255 area 0
 network 190.111.1.0  0.0.0.255 area 0
!
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 network 10.10.0.0 mask 255.255.0.0
 network 190.111.1.0 mask 255.255.255.0
 neighbor 10.10.10.1 remote-as 1
 neighbor 10.10.10.2 remote-as 1
 neighbor 10.10.10.4 remote-as 1
 no auto-summary
!
line con 0
 password as#as#
 login
line aux 0
line vty 0 4
 password as#as#
 login
!
```

Testing

OSPF Neighbors

Before OSPF will exchange routing information, it must establish neighbor relationships between the routers in the AS, that is, the two pod routers and the two AS routers. The list of OSPF neighbors detected on the local network can be viewed with the command **show ip ospf neighbor**. This will show each neighboring router, the state of the relationship (FULL, 2WAY) and the role of the router (DR, BDR, DROTHER). Each router should show three neighbors. If the OSPF priority was set to 0 on both pod routers, then the two AS routers should have been assigned the roles of DR and BDR. The two AS routers should show all three neighbors as being in the FULL state. On the pod routers, only the DR and BDR should be in the FULL state, the other pod router should have a state of 2WAY. Routers on the same Ethernet segment will only peer with the DR and BDR, not with each other. The 2-WAY state shows that the pod routers were able to establish communication but are not exchanging information since that will be handled by the DR and BDR.

The following output was taken from the pod3 router in AS3:

```
pod3router#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.30.10.1        1   FULL/BDR        00:00:39    10.30.10.1    GigabitEthernet0/1
190.111.4.254     0   2WAY/DROTHER    00:00:31    10.30.10.5    GigabitEthernet0/1
192.168.1.2       1   FULL/DR         00:00:38    10.30.10.2    GigabitEthernet0/1
```

Note that the first field is the neighbor ID, not the neighbor's IP address. OSPF selects the highest IP address on the router as its ID. The actual IP addresses of the neighbors should all be from the subnet between the AS and pod routers (e.g. 10.x.0.0). Also notice that the priority of each neighbor is reported.

If no neighbors are listed but there are other routers running OSPF on the segment then the most likely cause is a mismatched netmask on the interface. For OSPF to form neighbor relationships, all OSPF routers on the segment must be in the same IP subnet, have the same netmask, and be using the same OSPF timer values.

BGP Neighbors

Like OSPF, BGP must also establish neighbor relationships to exchange routing information. Every router should have three iBGP neighbors (neighbors with the same AS number). Each AS router should also have an eBGP neighbor in the directly adjacent AS. The command **show ip bgp summary** will give a brief list of the BGP neighbors, their AS numbers, and the state of the connections. It is important to note that when a neighbor relationship is in the Established state, the summary will show the number of prefixes (networks) received from the neighbor, but when BGP is attempting to establish a connection, the neighbor will be in the "Active" state. If a neighbor remains in the Active state (or any other state) then this is a sign of a problem, most likely related to physical connectivity or routing to the neighbor's network. The output from the routers should resemble the following:

```
AS3R2#show ip bgp summary
BGP router identifier 190.111.50.15, local AS number 3

...

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.30.10.1      4     3    9866    9869       26    0    0 6d20h           4
10.30.10.3      4     3    9860    9869       26    0    0 6d20h           2
10.30.10.5      4     3    9860    9869       26    0    0 6d20h           2
192.168.1.1     4     1    9872    9866       26    0    0 6d20h           6
```

If more detailed information is needed about the neighbors then the command **show ip bgp neighbors** can be used. Other helpful commands are **show ip bgp neighbors <neighbor address> advertised-routes** and **show ip bgp neighbors <neighbor address> routes** which will show the routes being advertised to a neighbor and the routes learned from a neighbor respectively.

Routing Tables

If the routing protocols have been configured on all routers and all neighbor relationships are up, then each router should have at least a route to each pod (six total), a route to each of the three AS networks, and either a static default route or two OSPF default routes. In addition, each AS router will have directly connected routes to the lab backbone and to the point-to-point crossover (192.168.1.0) networks, neither of which should be advertised by a routing protocol. Each router will learn about the networks within its AS either from OSPF or by being directly connected. The routes to the pods and networks in other ASes will be learned from BGP.

To display the routing table on a Cisco router use **show ip route**. On the Linux routers, there are in fact two routing tables maintained. The Linux kernel maintains the primary routing table, which is the table that is consulted to perform the actual routing of packets. The Linux routing table can be displayed by using the command **route –n** at a terminal prompt. The second routing table is maintained by Quagga's zebra server. Zebra's routing table is built from the routing information gathered by ospfd and bgpd. Zebra then selects the best routes and passes those to the Linux kernel's routing table. The routes that are passed to the kernel are flagged with a *. To see the Zebra routing table, telnet to port 2601 on the local host, login, and use the **show ip route** command. The following is the routing table of the pod1 router in AS1. Other pod routers should be nearly identical to this:

```
Router# show ip route

...

Gateway of last resort is 10.10.10.1 to network 0.0.0.0
     190.111.0.0/24 is subnetted, 6 subnets
C       190.111.1.0 is directly connected, FastEthernet0/0
B       190.111.3.0 [200/0] via 10.10.10.1, 01:24:06
O       190.111.2.0 [110/2] via 10.10.10.4, 01:30:14, FastEthernet0/1
B       190.111.4.0 [200/0] via 10.10.10.1, 01:24:06
B       190.111.9.0 [200/0] via 10.10.10.2, 00:23:45
B       190.111.8.0 [200/0] via 10.10.10.2, 00:23:45
     10.0.0.0/16 is subnetted, 3 subnets
C       10.10.0.0 is directly connected, FastEthernet0/1
B       10.30.0.0 [200/0] via 10.10.10.1, 01:24:07
B       10.20.0.0 [200/0] via 10.10.10.2, 00:23:46
O*E2 0.0.0.0/0 [110/12] via 10.10.10.1, 01:30:15, FastEthernet0/1
               [110/12] via 10.10.10.2, 01:30:15, FastEthernet0/1
```

Notice the two default routes in the above routing table. This is the effect of the **default-information originate metric 12** command on the AS routers. With both AS-1 routers up, there are two default routes. With one down, there will be only one.

Ping

If the routing tables are correct on all of the routers, it should be possible for each pod to ping every address within the lab network (except 192.168.1.0 addresses). From a pod router, you should be able to ping the pod network interface of each other pod router (190.111.x.0), the AS network interface of each pod and AS router (10.x.0.0), and the backbone interface of each AS router (190.111.50.0). From an AS router you should be able to ping every host within the AS, but pings into to the directly adjacent AS will likely fail (see the Troubleshooting section for a full explanation).

If routing is configured on the lab backbone router, you should also be able to access the Internet from a host on the pod networks. Pings should also work from the pod hosts. If you are able to ping from the pod router but not the host, verify that the pod host has its default gateway set to the pod router's IP address.

Note that on Cisco routers, the very first ping packet sent to a host will fail. Presumably, this is a result of the delay cause by the initial ARP request to resolve the host's MAC address. You should always wait until two ping packets have failed before terminating the ping.

Shutting down each AS router

The final test of the success of this lab is to simulate a failure of each AS router. If everything is configured correctly, all of the routers should adjust their routing tables so that any affected routes will be replaced by alternate routes that go through the remaining AS routers. This process may take several minutes as the routers will first wait for their neighbor relationships to the downed router to expire before recalculating their routing tables.

The routers that will be affected most by the shutdown are the two neighboring pod routers within the AS and the routers of the adjacent AS, which previously was connected through the downed AS router. The pod routers will replace nearly half of their routes. Almost every route will then go through the remaining AS router. They will also lose the OSPF default route learned from the downed router. The routers in the adjacent AS will use alternate routes that go through the third AS to reach the pods and network of the now disconnected AS.

Once all of these changes have happened and the network has stabilized, there again should be a route to every network, the pod routers should be able to ping any destination within the lab, and if the backbone router is configured, the pods should be able to access the Internet. When the downed router is restored, another recalculation should occur and the routing tables should return to their previous state.

Shutting down the Linux router

Instead of shutting down the entire Linux router it is easiest to use the command **service network stop** to shutdown just the network interfaces and **service network start** to bring them back up. Note that the stop command also turns off IP forwarding and causes any manually added routes to be cleared. The start command will re-enable IP forwarding if it was set in /etc/sysctl.conf and will reconfigure any routes set within the network configuration applet. Another option is to use the commands **ifdown** and **ifup** on each interface (eth0, eth1, eth2).

Lab Backbone Router Setup

Basic Configuration

The lab backbone router will provide the connection between the campus network and the lab network. It will not provide a path between AS routers within the lab network or in any way influence the routing decisions of the other routers. The exact configuration will vary depending on how you connect to your campus network and the local policies of your institution. The initial configuration will be similar to that of the other Cisco routers.

One interface on the router should be attached to the lab backbone network. The address of this interface should be **190.111.50.55** with a network mask of **255.255.255.0**. If a different address is used, then the static default routes created on the AS routers should point to that address. In addition to the IP address, the lab backbone interface should also be configured to suppress ICMP redirect messages. These messages are sent by the router when it has a more efficient route to a destination. However, since the backbone router is not supposed to influence routing on the lab network, ICMP redirects should be disabled on the interface by issuing the **no ip redirects** command in interface config mode. For example:

```
interface FastEthernet 0/0
 ip address 190.111.50.55 255.255.255.0
 no ip redirects
```

The backbone router is assumed to have one interface connected to the campus network. The ip address on the campus interface can be set statically, or the **ip address dhcp** command can be used to receive an address from a DHCP server. The router will also need a default route to your campus router. This can be created statically with **ip route 0.0.0.0 0.0.0.0 <campus router IP>** or, on recent versions of the Cisco IOS, it can be learned via DHCP with **ip route 0.0.0.0 0.0.0.0 DHCP**.

If your backbone router must run a routing protocol on the campus side then you must make certain that none of the routes from the Network Design lab are injected into the campus network. If you are running a routing process other than RIP, make sure that process does not have a network statement for the lab backbone network or any other options that might result in routing information being redistributed into the process.

If you are required to run RIP on the campus network interface and you wish to use RIP for the lab network as well, then you must take special care that lab routes do not get injected into the campus network. The easiest way to prevent this is to use the **passive-interface <campus interface>** command in "router rip" config mode. This will allow RIP to learn routes from the campus network but it will not advertise any routes to it. If RIP must advertise routes then you will have to use the distribute-list command and an associated access list to create a route filter to remove the lab networks from the outgoing RIP advertisements.

Access Lists and Packet Filtering

Access lists serve a dual purpose on Cisco routers. When applied to a network interface they will filter packets, but they are also used by many other commands to identify networks or routes for special handling. Standard access lists are numbered from 1-99 and only match on the source address. Extended lists are in the 100-199 range and can match the source and destination address, and optionally TCP/UDP ports. Similar to OSPF network statements, access-lists use a wildcard mask instead of a netmask. The keyword, "any" can be used in place of an address/wildcard-mask pair to match any address.

Most importantly, access-lists have an implicit "deny any" statement at the end of the list. That is, if an address does not match any permit statement in the access list, then the router will automatically deny that address even though there is no deny statement explicitly given in the list.

At most institutions, there is a computer usage policy that often prohibits certain types of traffic on the campus network. On many campuses, computer labs will have their traffic filtered by the router serving as the default gateway for the lab. In the case of this lab configuration, it may be possible for the hosts on the lab network to bypass the normal filters put in place by the institution. Although it is not required for any part of this lab configuration, you may wish to setup an access list that will filter all traffic leaving the lab network and only permit what is deemed acceptable by your institution. The following example blocks all outbound network traffic except for a few basic services:

```
access-list 103 permit tcp any any eq www
access-list 103 permit tcp any any eq 443
access-list 103 permit tcp any any eq domain
access-list 103 permit udp any any eq domain
access-list 103 permit tcp any any eq telnet
access-list 103 permit tcp any any eq 22
access-list 103 permit tcp any any eq ftp
access-list 103 permit tcp any any eq ftp-data
access-list 103 permit tcp any any eq 1521
access-list 103 permit tcp any any eq 143
access-list 103 permit tcp any any eq smtp
access-list 103 permit tcp any any established
access-list 103 permit icmp any any
```

```
!
interface FastEthernet0/1
 ip access-group 103 out
```

Note that the implicit deny at the end of this list will block every other type of packet, including non-IP packets used by OSPF, IGRP, and EIGRP, as well as UDP 520 (RIP) and TCP 179 (BGP). If you need to receive routing information from OSPF, EIGRP, or BGP, you must permit outbound packets for those protocols or else they will not be able to establish connections to their neighbors.

Network Address Translation

If you use the same IP addressing scheme demonstrated in this lab then you will have to use some form of Network Address Translation (NAT) to allow hosts to access the Internet. NAT will rewrite the IP addresses of packets originating from inside the lab network before they are routed to the campus network so that those packets will be sourced from a valid, routable IP address. One particular form of NAT, called Port Address Translation or overloading, will translate multiple internal addresses to a single global address. This will allow every host on the lab network to access the Internet through a single IP address on the campus network and is similar to Internet Connection Sharing in Windows or IP Masquerading in Linux.

To create the translation, you will have to make an access-list that defines the addresses from inside the lab network that the router is expected to translate. You will also have to designate which interface is the inside network (the lab backbone) and which is the outside (the campus network). The following example assumes that fa0/0 is the lab network and fa0/1 is the campus network and will create a translation to map all of the lab subnets to the IP address used by the campus interface:

```
interface FastEthernet 0/0
 ip nat inside
!
interface FastEthernet 0/1
 ip nat outside
!
access-list 5 permit 190.111.0.0 0.0.255.255
access-list 5 permit 10.0.0.0 0.0.0.255
access-list 5 permit 192.168.1.0 0.0.0.255
!
ip nat inside source list 5 interface FastEthernet 0/1 overload
```

If you have a DNS server (or any other server) on the lab network that will need to be accessible from the Internet then you will have to create a separate static NAT mapping that will give each server its own routable IP address. This is necessary because overloading works by recording the source port of a translated packet and then when a response is received on that source port the router can determine which inside host should receive the packet. A server will receive unsolicited packets to various ports and the router has no way of telling to which inside host it should forward the packet since they are all sharing the same IP address. To permanently map an IP address from the campus network to a server on the lab network, use:

```
ip nat inside source static <server's inside address> <server's campus address>
```

Routing between the lab network and the campus Internet connection

The primary purpose of the lab backbone router is to provide each AS with access to the Internet. Every router should have a default route pointing to the lab backbone router. For the lab router to forward packets back into the lab network, it must have a route back to every subnet (10.*.0.0 and 192.111.*.0). At the same time, the lab backbone router should not provide routes between ASes or in any way influence the routing tables of the lab network. For this reason, running a dynamic routing protocol on the backbone network

poses a problem. If there were a routing protocol running between the AS routers and the lab backbone router, each AS router would quickly learn that it could reach every network in the other ASes through the backbone router and would have no need to use BGP or the crossover links between ASes.

At first glance, it might seem that the solution is simply to create static routes to each subnet with a next hop of the AS routers that are connected to that subnet, but this too poses a problem. One of the tests for this lab is to shutdown each AS router and check to see if the other routers adjust to use the second AS router. If the lab backbone router had a static route to a subnet that went through the downed AS router, it would have no way of knowing that an AS router is down and would continue attempting to route packets through it. The only way the lab router could detect the failure of an AS router is if it had a point-to-point connection to it. That would require either a lab router with six Ethernet interfaces or a layer 3 switch. Fortunately, there are solutions that will work with a standard switch and router setup but they require additional configuration. The following sections provide three methods for handling routing between the backbone and the ASes. The eBGP configuration is the preferred method.

Routing configuration using a non-transit eBGP backbone AS

The most effective method to distribute routes to the lab backbone router is with eBGP. With this method, the lab backbone router is setup as its own AS and each AS router is setup to form an eBGP neighbor relationship to the lab router. Unlike a typical IGP, BGP will not broadcast routing information to every other router on the backbone network, it will only advertise to its neighbors. Since the AS routers will not be configured as neighbors, they will not learn routes from each other.

By default, a single router configured with BGP will operate as a transit AS. That is, it will forward eBGP routes learned from one AS to neighbors in other ASes, thus providing a path between those ASes. If the lab router were left as a transit AS, each eBGP neighbor would learn that it could reach the other networks by going through the backbone AS. To prevent this, the backbone router must be configured as a non-transit AS, that is, it must not forward routes learn from one eBGP neighbor to other eBGP neighbors.

The backbone router must use the distribute-list command within the BGP process to filter outgoing BGP route advertisements. The distribute-list will compare each route that will be advertised by BGP to a given access-list, those routes that match a deny entry in the access list will not be advertised to neighbors. The access list can be a standard list with a single entry that denies any network.

In addition to the required BGP configuration, it is helpful to set the BGP hello and hold timers to lower values. By default, BGP will only send a hello packet to neighbors every 60 seconds and will wait up to 3 minutes without receiving a hello packet before it will presume a neighbor is down. When an AS router is shutdown and hosts within that AS attempt to access the Internet, the backbone router will continue sending return packets through the path with the downed AS router for 3 minutes before finally removing the route. To reduce the waiting time for BGP to recalculate the topology, the hello time can be set to 10 seconds and the hold time to 30 seconds, which is slightly faster than the OSPF equivalents.

Although not necessary, it is also helpful to set BGP's maximum-paths to 2. Normally, BGP will only place one route to a destination in the routing table even when there are multiple equal-cost paths. Since each AS router will provide routes to the networks within their AS, there will be two equal-cost routes to every network address in the lab. Although BGP is aware of the second route and would use it if the first route were to go down, it will not add the second route to the table or balance the load over them. Note that when maximum-paths is set to 2 and one of the AS routers has been shutdown, pings to the Internet from behind the downed AS router will alternate between timed out and succeeded. The lab router will load balance every other ping to the downed AS router until the hold timer expires and BGP removes the downed route.

In addition to the normal BGP configuration on each AS, you will need to issue the **neighbor 190.111.50.55 remote-as 50** command in BGP router config mode. On the lab backbone router, the following configuration will create an access list for the distribute-list filter and run BGP as AS 50:

```
access-list 50 deny any
router bgp 50
 maximum-paths 2
 timers bgp 10 30
 distribute-list 50 out
 no auto-summary
 no synchronization
 bgp log-neighbor-changes
 neighbor 190.111.50.10 remote-as 1
 neighbor 190.111.50.11 remote-as 1
 neighbor 190.111.50.12 remote-as 2
 neighbor 190.111.50.13 remote-as 2
 neighbor 190.111.50.14 remote-as 3
 neighbor 190.111.50.15 remote-as 3
```

Although static default routes are used between the AS routers and the backbone in this lab configuration, it is possible to distribute a default route to the AS routers via BGP by adding **access-list 50 permit 0.0.0.0 0.0.0.0** and including **network 0.0.0.0** under router bgp 50.

Routing configuration using RIP with suppressed advertisements

If it is not possible to run BGP, another method is to run RIP between the lab backbone router and the AS routers but to suppress all incoming RIP routes on the AS routers. This would allow the backbone router to learn all of the interior networks in the lab and dynamically adjust to downed AS routers while preventing the AS routers from learning routes through the backbone network. Although this method will produce the best paths, it requires a bit of additional configuration on the AS routers. It should be used only if running BGP on the lab backbone router is not an option.

Each AS router must run RIP and use the distribute-list command within RIP to filter out any incoming RIP routes. The distribute-list will compare each route learned by RIP to a given access-list, those routes that match a deny entry in the access list will be dropped by the RIP routing process. The access list can be a standard list with a single entry that denies any network.

Because RIP will only run on the lab backbone (see Figure Lab-4) network, the OSPF and Connected routes of each AS router will have to be redistributed into the RIP process. This is necessary because RIP uses the network statement not only to learn on which network interfaces to run but also to know which directly connected networks to advertise. Since there will not be a network statement for any directly connected network other than the backbone, the other connected networks must be manually redistributed into RIP. Furthermore, since the pod routers are running OSPF as their IGP and not RIP, the RIP process will have no way of learning routes to the pod networks. So you must manually redistribute the OSPF learned routes into RIP as well.

On Cisco routers, a metric must be specified to redistribute OSPF into RIP. To ensure consistent metrics, a redistribute metric should also be used on the Quagga routers. Also, Cisco routers will advertise classful summary routes by default, that is, they will advertise 10.0.0.0/8 and 190.111.0.0/16 instead of the individual subnets. Because of this, the **no auto-summary** command must be used in RIP or the backbone router will

receive the same summary routes from both Cisco routers. Quagga does not auto-summarize routes. The following commands will configure RIP on the Cisco AS routers:

```
access-list 10 deny any
router rip
 version 2
 redistribute connected metric 12
 redistribute ospf 1 metric 12
 distribute-list 10 in
 network 190.111.0.0
 no auto-summary
```

To configure RIP on the Quagga Linux AS routers you must first create the **/etc/quagga/ripd.conf** file and start the service as described above for the other Quagga servers. Once running, you may telnet to port 2602 and issue the following commands in config mode or enter them into the ripd.conf file:

```
access-list 10 deny any
router rip
 version 2
 redistribute connected metric 12
 redistribute ospf metric 12
 distribute-list 10 in
 network 190.111.50.0/24
```

The lab backbone router will require only a minimal RIP configuration to receive the routes. Note that RIP is relatively slow to update the routing table and may it take several minutes to reflect that an AS router is down. Until then, the lab router will continue to use the downed route. Also be aware that the backbone router will receive multiple routes to the 192.168.1.0 network. Because no AS router should ever send a packet to the backbone sourced from 192.168.1.0, there should be no issue with these routes being in the table. If they should cause a problem for other reasons, the routes can be filtered using a distribute list on the backbone router. Issue the following commands in config mode to setup RIP on the backbone router:

```
router rip
 version 2
 network 190.111.0.0
```

Routing configuration using static routes and proxy ARP

The simplest but least efficient solution is to use static routes that have a next hop of the backbone network interface and enable proxy ARP on the AS routers. This will cause the lab backbone router to treat every subnet as though it were directly connected to it via the lab backbone switch. Each time the lab router attempts to send a packet addressed to a host on one of the interior subnets, it will broadcast an ARP request for that host's address on the backbone network. Each AS router configured for proxy ARP will pick up this request and check its routing table to see if the host is reachable through one of its routes. If the host is reachable, the AS router will respond with its own MAC address and transparently forward packets to the real host. Cisco routers are configured for proxy ARP by default. For Linux routers, proxy ARP must be enabled on the interface that is attached to the lab backbone network. For example, if the lab backbone is on eth1 then the following command will enable proxy ARP:

```
echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
```

To make this change permanent, set the variable **net.ipv4.conf.eth1.proxy_arp** equal to 1 in **/etc/sysctl.conf**.

For the proxy ARP method to work, the lab backbone router must send ARP requests frequently enough that, should a router that is acting as a proxy for the host go down, a different AS router will have a chance to respond with its MAC address and continue forwarding packets to the host. By default, Cisco routers cache ARP responses for four hours. This must be set to a much lower time, such as 60 seconds.

The disadvantage to this solution is that the path the lab backbone router will use to reach a host in an interior subnet is based entirely on which AS router responds first to the ARP request, not on which has the shortest path. When all of the routing protocols are fully operational, each AS router will have a route to every subnet used in the lab, but some of these routes will have been learned from BGP neighbors and may pass through another AS router before reaching the final destination. Since the backbone router is connected to all of the AS routers, it should never have a need to use a route that passes through two AS routers. However, each AS router will respond to the ARP request based on the fact that it has a route to the destination and not whether it has the best route. The first AS router to respond to the ARP request is the one the lab backbone router will use to forward packets to the destination. So when doing traceroutes to the lab router or hosts on the Internet, you may see packets taking a path that is not the most efficient.

The following configuration example is for the lab backbone router and assumes that the lab network is attached to FastEthernet 0/0. Also note that it is not necessary to create an individual route to each subnet, two summary routes can be used to match all of the interior subnets:

```
ip route 190.111.0.0 255.255.0.0 FastEthernet 0/0
ip route 10.0.0.0 255.0.0.0 FastEthernet 0/0
interface FastEthernet 0/0
 arp timeout 60
 no ip redirects
```

Troubleshooting

Pings and traceroutes from AS routers fail to reach certain destinations

When attempting to ping from the AS routers, it is important to realize from which interface the ping will be sourced. When the AS router attempts to ping a host on a network that is reached via its 192.168.1.0 interface (which can be determined from the routing table), it will use its 192.168.1.(1/2) address as the source IP for the ping. Because the 192.168.1.0 network is never advertised by any routing protocol, the only machines that will be able to respond to that ping is the directly adjacent 192.168.1.(1/2) router and any machines whose default route go through that router. Put simply, each AS router will be unable to ping most of the machines in the AS attached by the BGP crossover unless you manually specify the source IP of the ping/trace. This will only happen when pinging from an AS router, that is, a router directly attached to a 192.168.1.0 network. To specify the source IP address of ping and traceroute on a Cisco router, type **ping** or **traceroute** without any arguments and enter the source address when prompted. On Linux, use the commands **ping –I <src address> <destination>** and **traceroute –s <src address> <destination>**.

eBGP routes with a next-hop of 192.168.1.(1/2) aren't added to the routing table

For a route to be considered valid by BGP, it must have a next-hop address that can be reached using the existing routing table. Unlike most routing protocols, when iBGP forwards a route it learned from an eBGP neighbor, it leaves the next-hop attribute as the IP address of the eBGP router that originated the route. In this lab setup, the eBGP routes will always originate from a 192.168.1.(1/2) address. When the iBGP neighbors receive these routes they will reject them because they do not have a route to the 192.168.1.0

network. To fix this, each AS router (those with an eBGP crossover) needs to set the next-hop address to its self when it advertises eBGP routes to its iBGP neighbors inside the AS. This can be done by setting the "next-hop-self" flag on each iBGP neighbor (every neighbor except the 192.168.1.(1/2) neighbor). For example:

```
Router (config)# router bgp 1
Router (config-router)# neighbor 10.10.10.2 next-hop-self
Router (config-router)# neighbor 10.10.10.3 next-hop-self
Router (config-router)# neighbor 10.10.10.4 next-hop-self
```

Linux interfaces remain up even when disconnected

Unlike Cisco routers, Linux network interfaces remain in the up state even when the cable is unplugged. As a result, routes that depend on that interface remain in the routing table even when there is no longer a network attached to it. Furthermore, BGP and OSPF will continue advertising routes that go through that interface to all the other routers. So instead of routing around the disconnected router, packets continue to be sent to the interface as normal, where they are then dropped. If you wish to test the response of a single interface being disconnected, use the **ifdown <interface>** command to shut down the interface and **ifup <interface>** to bring it back up.

Quagga continues to advertise BGP routes to downed interfaces

By default, Quagga's bgpd does not check that it can actually reach the networks it advertises (as given by the network statements in the configuration). So if an interface is shutdown, bgpd will continue to advertise the network on that interface even though the Linux machine has removed the directly connected network from the routing table. Quagga bgpd can be made to verify the existence of advertised networks by issuing the following commands:

```
Router (config)# router bgp 1
Router (config-router)# bgp network check-import
```

Quagga doesn't forward eBGP routes to peers with a 192.168.1.(1/2) router ID

By default, BGP (and OSPF) use the high IP address on the machine as the router's ID. On the AS routers this will usually result in an ID of 192.168.1.(1/2). It appears that Quagga 97.3 will not forward eBGP routes to a router with an ID that is the same as the IP address of the next-hop of the eBGP route. To eliminate this problem, each AS router should have a manually set router-id. We would suggest using the router's lab backbone interface address as the BGP router ID. For example:

```
Router (config)# router bgp 1
Router (config-router)# bgp router-id 190.111.50.10
```

Cisco routers refuse to accept BGP configuration commands

The "IP Base" IOS image that comes pre-installed on most Cisco routers does not include support for BGP. To support BGP commands, the IOS must be upgraded to an "Advanced" or "Enterprise" level image. A new image can be copied to the router using the command **copy tftp flash** and entering the image name and TFTP server address when prompted. Note that the router must have at least one interface configured and be capable of reaching the network on which the TFTP server is located. See the appendix for instructions on configuring a Linux TFTP server and upgrading the IOS.

To be able to learn BGP routes directly, every router in the AS must be running BGP. If this is not feasible then it is possible to run BGP only on the AS routers and have them redistribute BGP routes into OSPF. This should permit the pod routers to learn BGP routes without the need to run BGP. Note that the AS routers will still need to have an iBGP neighbor relationship between them. To redistribute BGP into OSPF on a Cisco router issue the commands:

```
Router (config)# router ospf 1
Router (config-router)# redistribute bgp <AS #> subnets
```

On a Quagga router use:

```
Router (config)# router ospf
Router (config-router)# redistribute bgp
```

Pod routers do not regain their default route after an AS router is restarted

This appears to only be an issue when both of the AS routers are using Quagga. OSPF normally uses the highest IP address on the machine as its router ID. If every interface is not up when the OSPF process is started then it will use the highest address currently available. If all the interfaces are shutdown then the OSPF process will effectively be reset. When the interfaces are brought back up OSPF may pick a different router ID than what it was using before the restart since more interfaces are up. At the same time, the other Quagga AS router keeps the default route associated with the old ID in its database, it merely flags the route as being down as it waits for the router with the old ID to return. Since that router ID will not return, the original default route remains down. OSPF will see the default route from the new ID as a third default route, but for reasons unknown it will not add it to the routing table immediately. Eventually, the old default route will expire from the database and the new one will be added. Also, restarting ospfd will trigger an update. To verify that this is the cause of the problem, use the **show ip ospf database** command on the pod router. If there are more than two 0.0.0.0 Link IDs then this is likely what is happening.

To avoid this problem, it is best to set the router ID for each OSPF process to the IP address of the 10.x.0.0/16 interface. For example:

```
Router (config)# router ospf
Router (config-router)# router-id 10.10.10.2
```

On production networks it is normally recommended that a loopback interface be created on each OSPF (and BGP) router with the IP address that you want to be the OSPF router ID (typically from a private address block). OSPF will automatically use the IP address of the first loopback interface it finds as its ID. Since the loopback will always be up, the ID will remain stable.

Cisco routers will not boot a newly installed IOS image

If there is sufficient room on the flash device when you copy a new IOS image to the router, it will permit you to have both the original image and the new image. However, the router will continue to boot the first image it finds on the flash device. The command, **show flash** will show the list of images and their order. If you wish to boot to the new image while still keeping the original you will have to manually specify which image the router should boot using the command **boot system flash <image name>**. For example:

```
Router (config)# boot system flash c2691-i-mz.123-18.bin
```

Because the boot system command is a configuration command, it must be present in the startup-config to work. As a result, doing an erase startup-config will cause the router to return to booting the original image.

To make the router permanently default to booting the new image, both images must be deleted from flash and recopied from TFTP in the desired order.

Debug commands and logging

In addition to the show commands, both Cisco and Quagga support an extensive range of debug commands. On the Cisco routers, all debugging output is logged to the console screen by default, no configuration is required. To see the output of debug commands on the Quagga routers you must enable logging to a file with the command **log file /var/log/quagga/<service name>.log,** replacing <service name> with bgpd, ospfd, etc. This command can be in the service's config file or issued in config mode. Be aware that the quagga user must have write permission to the specified file path. If Quagga was installed from the Fedora RPM package, /var/log/quagga should already exist and be owned by the Quagga user, if not, the directory must be created and its owner changed. Alternatively, the log file can be placed under /tmp. Once enabled, all debugging output will be recorded to the specified log file.

On the Cisco routers, there are numerous debug commands available. To see a list of the IP related debugging options issue the **debug ip ?** command in privileged mode exec mode. Some of the more useful debug commands are **debug ip ospf events**, **debug ip bgp events**, and **debug isis adj-packets**. A listing of every packet received by the router (not including forwarded packets) can be seen by using the **debug ip packet** command. Once a debugging command has been issued, the router will continue outputting debugging information to the screen until a "no" version of the debug command is issued. To turn off all debugging use **undebug all** or **no debug all**.

The debug commands are similar on the Quagga routers with the exception that they do not include the "ip" keyword and each Quagga service only contains the debug commands related to that service. The Quagga equivalents to the above commands are **debug ospf event**, **debug bgp events**, **debug isis adj-packets**, **debug isis events**. Note that the "debug ospf event" command does not have a trailing "s."

The Why's and the Wherefore's

Why would we use a lab project in which three AS have a triangular arrangement with cross-connected logical links between the routers in each AS, and effective physical links between the AS? The answer is that this practice is performed by most ISPs, web hosting services and other enterprises for reliability.

An example is that of Superb web hosting services (www.superb.net). In the Figure Lab-3 below, there is clearly a cross-connecting relationship between core routers and Catalyst switches.

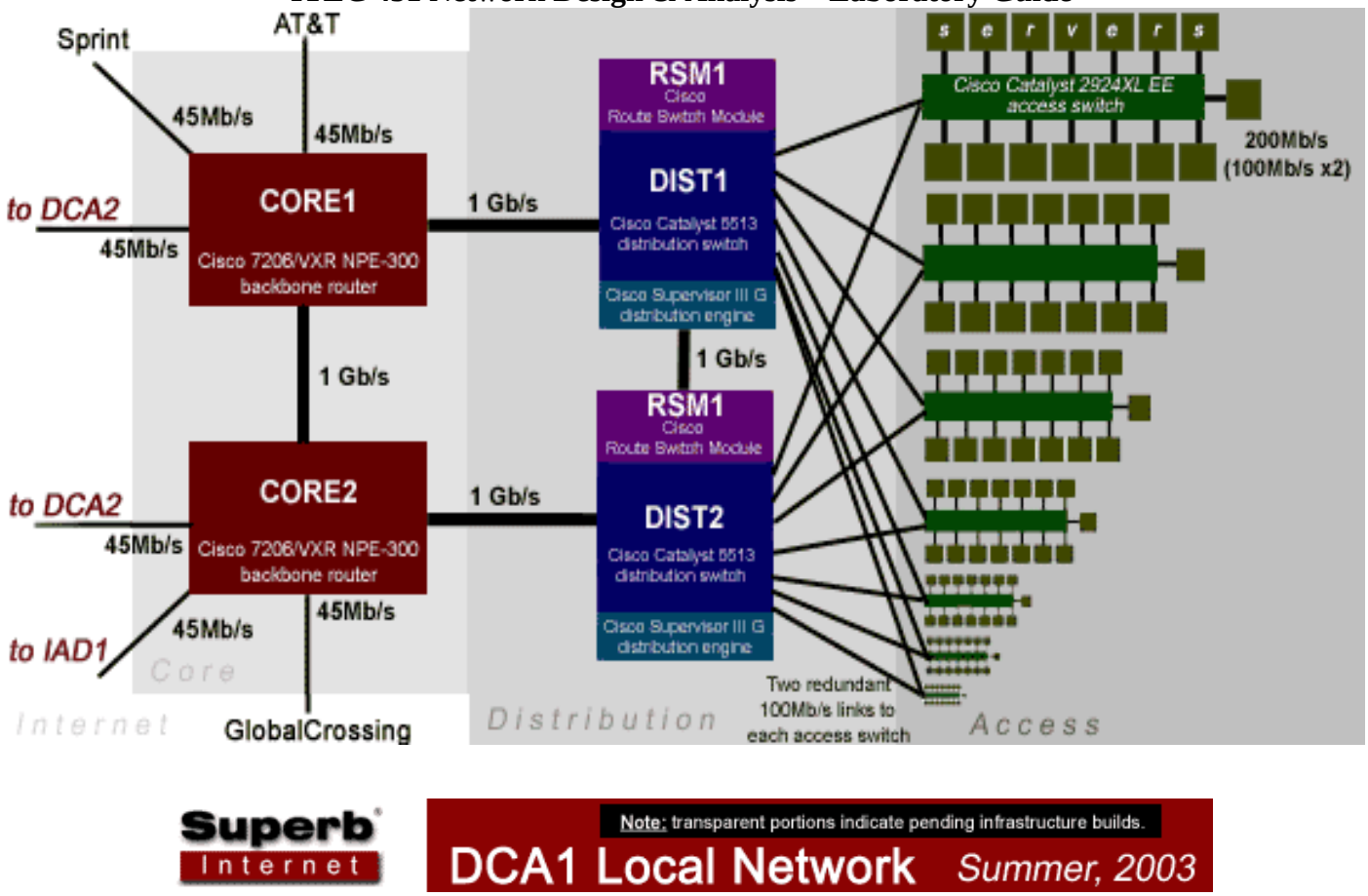

Figure Lab-5. Washington DC Superb Internet, Summer 2003 © Superb Internet.

Note the division of the network into Core, Distribution and Access layers. This is a configuration for a "virtual hosting" company that provides virtual and dedicated hosting for web sites, email and other services. A similar arrangement is often used by Enterprises and ISPs where [CCNA-Thomson, Ch4, p. 154] shows a Core, Distribution and Access three-layer Network Model in figure 4.25.

Using IS-IS in place of OSPF

As an alternative to using OSPF, IS-IS can be used as the IGP for each AS. IS-IS is very similar to OSPF in both configuration and operation. In IS-IS, a router is called an Intermediate System (IS). Like OSPF, IS-IS elects a Designated Intermediate System (Designated Router in OSPF) which forms an adjacency with each neighboring router (Intermediate System) on the segment. Each IS-IS router sends a single Link State Protocol-data-unit (LSP) to advertise all the networks that are attached to it. This is very similar to OSPF's Link State Advertisements but more efficient since only one LSP is sent per router. IS-IS also supports multiple areas using a two level hierarchy. Unlike OSPF, IS-IS was not originally designed for IP and uses the OSI network layer protocol CLNS as the basis for communication between routers.

To communicate using OSI CLNS, each IS-IS router must be given a Network Entity Title (NET), which serves as the router's OSI address. The NET address is an 8 to 20 byte hexadecimal number separated at key places by dots. The OSI standard permits the length and included parts of the NET to vary but in general it has a format similar to **<Area ID>.<System ID>.<NSEL>**. The exact values of each part are also left up to individual preference but with some conditions.

As the name implies, the area ID identifies the area to which the router belongs. All routers within the same area must have the same area ID. The area ID consists of two hex values separated by a dot, a two digit address class and a four digit area number. The address class identifies the type of address. Although any number will work, the OSI standards designate type 49 as the private address class. The area number can be any number but is easiest if it corresponds to the AS number. We use 49.0001, 49.0002, and 49.0003 as the area IDs for AS1, AS2, and AS3 respectively.

The system ID is 6 byte hex value that uniquely identifies the router within the area. It is formatted as three numbers of four digits each separated by dots, very much like a MAC address. On many production networks, MAC addresses are used as the system ID. Another popular convention is to simply start at 0000.0000.0001 and assign values sequentially. To better relate the NET address to the router's AS and IP address, we use the AS number as the first four digits and the last number in the router's 10.x address for the last number in the system ID. For example, AS2's 10.20.10.4 would have a system ID of 2222.0000.0004.

The final number in the NET, the NSAP Selector (NSEL), is similar in nature to a TCP/UDP port number. The NSEL has no function in routing and must always be set to 00 to signify that the connection is to the router itself and not an upper layer protocol.

Note that since there is no IS-IS connection between areas, it is possible to reuse the same NET addresses within each AS. However, it is best if each router has a globally unique NET since this serves as the router's address in OSI protocols. It is also possible to use the same area number for every router.

As of the time of this writing, IS-IS support in Quagga is in the alpha development stage. None of the binary distributions of Quagga include the isisd server, nor is it enabled in the default build of the source. To install isisd, Quagga must be built with IS-IS support enabled. The primary difference between the standard build and the IS-IS build is the inclusion of the --enable-isisd flag in the configure script options. Specific instructions for building Quagga can be found in the appendix. Once installed, isisd will require the same initial configuration file used for the other Quagga servers. The server can be started with **service isisd start** and configured through **telnet localhost <port>**, where <port> is either isisd or 2608.

To start the IS-IS routing process and enter IS-IS router config mode, issue the command **router isis <area tag>** in global config mode. Use the name of your AS (AS1, AS2, AS3) as the area tag. The area tag is merely a label for the local IS-IS process and is used when associating interfaces the process.

To assign a NET address to the router, issue the command **net 49.000<x>.<xxxx>.0000.000<y>.00** in router-config mod, where <x> is your AS number and <y> is the last number in your router's 10.<x>.10.<y> address. It is important to be aware that because the lengths and parts of the NET address are variable, the router will not complain if a part is left out or is shorter than it should be. However, it is likely that any unintentional variations in the NET address will prevent the router from communicating with its neighbors. To be notified when a link to a neighbor has been established, enter **log-adjacency-changes** in router-config mode.

When doing multi-area IS-IS, routing information within an area is normally exchanged at Level-1 (L1) and routing information between areas is exchanged at Level-2 (L2). Routers that forward packets between areas operate at both levels and are designated as L1L2 routers. Both Cisco and Quagga IS-IS routers operate as L1L2 routers by default. When all routers are in a single area, it is best to set the router to operate at L2 only by entering **is-type level-2-only** in router-config mode. The following configuration was taken from AS1R1:

```
AS1R1 (config)#
router isis AS1
```

```
net 49.0001.1111.0000.0001.00
is-type level-2-only
log-adjacency-changes
```

Unlike other routing protocols, IS-IS does not have a network statement. To include an IP network in the IS-IS process, the **ip router isis <area tag>** command must be used on the interface associated with that network. On the AS routers, only the 10.x.0.0 interface should be included in IS-IS. On the pod routers, both Ethernet interfaces should be in IS-IS. To ensure that only an AS router is elected as the DIS, the pod routers should also have the IS-IS priority set to 0 on the AS network interface. The following examples are from the AS1R2 (Quagga) and Pod1 (Cisco) routers:

```
AS1R2-isisd (config)#
interface eth1
   ip router isis AS1

Pod1router (config)#
interface fa0/0
   ip router isis AS1
interface fa0/1
   ip router isis AS1
   isis priority 0
```

To verify that IS-IS is able to communicate with neighboring routers use **show isis neighbors**. To view the routing information database use **show isis database detail**. If the IS-IS process is making connections to its neighbors but does not appear to be updating its database or the routing table, use **clear isis *** on the Cisco routers to reinitialize IS-IS. The use of the clear isis command is often necessary to remove stale entries in the IS-IS database and trigger updates to the routing table. Note that it is sometimes necessary to simultaneously issue the clear command on all of the Cisco routers and do a **service isisd restart** on the Linux routers to completely reinitialize the IS-IS database.

Normally, the only remaining step in the IS-IS configuration process is to issue the default-information originate command in router isis config mode. However, the alpha version of isisd that is available at the time of this writing lacks support for propagating a default route back to the AS network. Fortunately, iBGP can be used as a substitute source for the default route. Note that if later versions of isisd include the default-information originate command, be aware that IS-IS only advertises the default route at Level-2. The IS-IS protocol uses the Attached (ATT) bit in advertisements to find the default router at Level-1, but only when using a multi-area configuration.

BGP can propagate default routes from the AS routers by using the network 0.0.0.0 command but that would advertise the default route over BGP to the entire lab network. To prevent that, the default route advertisement should be restricted to the immediate neighbors on the AS network (10.x.0.0). BGP has a per neighbor default-information option to limit the propagation of default routes. Only the two neighboring pod routers on the AS network need to receive the default route. On each AS router, issue the command **neighbor <pod router> default-information** for both of the neighboring pod routers. The following example is from AS1R2:

```
AS1R2-bgpd (config)#
router bgp 1
   neighbor 10.10.10.3 default-information
   neighbor 10.10.10.4 default-information
```

Once two default routes are being propagated to the AS network using iBGP, the pod routers must be configured to load balance between the two paths. By default, BGP will only place one route to a destination in the routing table. To allow both default routes to appear in the table, issue the command **maximum-paths ibgp 2** in BGP router config mode. Since the default routes will be learned via BGP, the only IS-IS route in the pod router's table will be that of the neighboring pod network.

As with the normal OSPF setup, the topology can be tested by shutting down each AS router. The easiest way to simulate a shutdown on the Linux routers is with **network service stop** or by using **ifdown** on each Ethernet interface. However, the current developmental version of Quagga isisd will die without warning when either method is used to shutdown the network interfaces. These methods can still be used but isisd will have to be restarted with **service isisd start**.

To access the Internet from the pod networks, the backbone router must have a route back to each pod. One of the options presented to enable the lab router to learn these routes is to use RIP with suppressed routes. With the RIP method, each AS router redistributes routes from the IGP into RIP. This method will not work with the currently available versions of Quagga because the ripd server does not support redistribution of IS-IS routes into RIP. Either the non-transit BGP or proxy ARP method must be used.

References:

Superb Internet web site: http://www.superb.net/. Click on Network to see newest details of DCA1

[CCNA-Thomson] Caudle, Kelly and Kelly Cannon, "CCNA Guide to Cisco Networking," 3rd Ed., Thomson, 2004.