Lab in the IPv6 introductory IPv6 course

Version 6.0 / January 20006

Kurt Erik Lindqvist

1. Content

2. Introduction

For these exercises you are split into groups of two. For each group you have one Cisco router, one computer running FreeBSD and one running Windows XP, SP2. The objective with the exercises is that you should learn to configures and use IPv6. In the first exercises you will learn to configure IPv6 on your

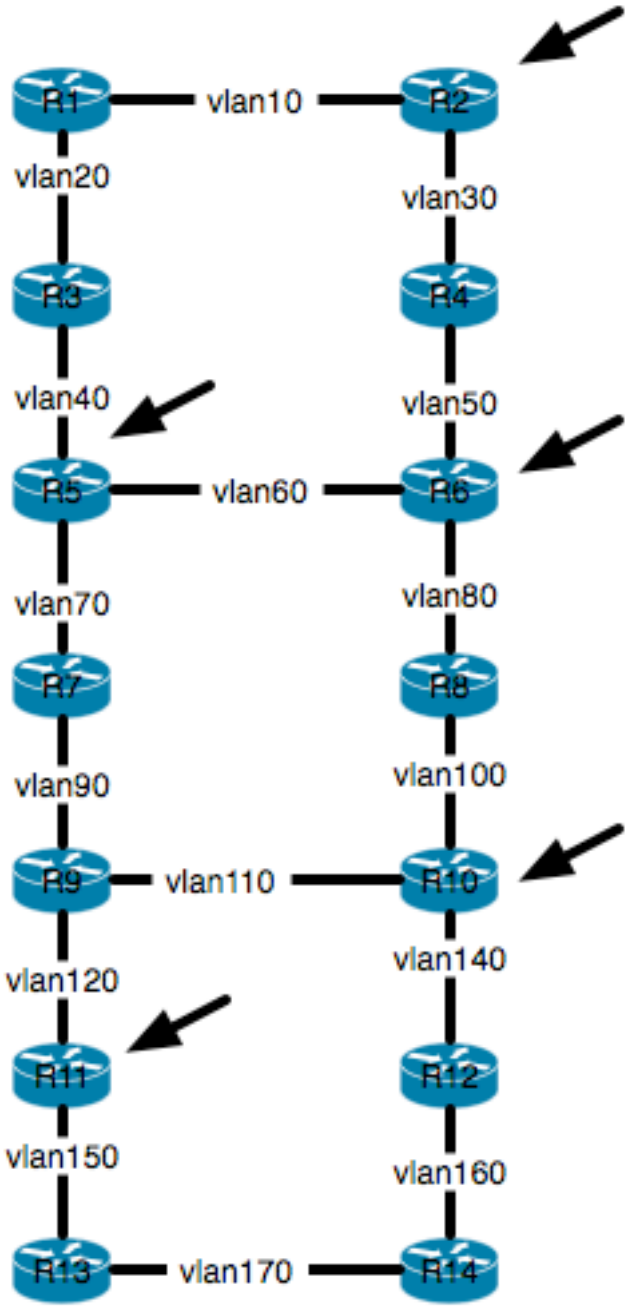
routers, and set-up your network for IPv6. In the second part of the exercises we will learn to set-up a Windows computer with IPv6. Then we will configure FreeBSD, as well as a number of services on the FreeBSD machine. We will also learn to look at the IPv6 packets on the wire.

3. Acknowledgement

The router and network exercises are modified versions of Cisco's ISP/IXP Networking Workshop Labs. The author would like to thank Philip Smith and Cisco for their generous permission to use these.

4. The lab network

The lab network is configured as a follows



Here we use a mix of Cisco and FreeBSD boxes running Quagga. The routers that are pointed to by arrows are Cisco routers running IOS. The other boxes are FreeBSD. We will need to install Quagga on them. The Quagga software can be found in the `/root` director as the file `quagga.tbz.` In order to install it we need to do the following steps

```
# pkg_add quagga.tbz
```

Then we need to edit the startup configuration

```
# vi /etc/rc.conf
```

We add the following lines

```
quagga_enable="YES"
quagga_daemons="zebra bgpd ospfd ospf6d"
router_enable="NO"
watchquagga_enable="YES"
```

Then we need to create the configuration files for quagga with

```
# cd /usr/local/etc/quagga
# touch bgpd.conf
# touch ospf6d.conf
# touch vtysh.conf
# touch zebra.conf
```

We then start quagga with

```
# /etc/rc.d/quagga start
```

You can enter the Quagga command shell using the `vtysh` tool as root (i.e, you must have logged in as root). Once in the vtysh tool it will look and behave as the Cisco command line interface (CLI). Where it doesn't we will note the differences. The chapter 5, the router labs, the FreeBSD boxes will the need of some of the routers.

5. Router and networking exercises

Cisco routers do not ship with IPv6 enabled by default. In order to continue our exercises we need to enable the IPV6 part of the IOS code. This is done with the following command

```
Router# config terminal
Router(config)# ipv6 unicast-routing
```

Save the configuration!

For Quagga the equivalent command is

```
Host# config terminal
Host(config)# ipv6 forwarding
```

Save the configuration!

5.1. IPv6 OSPF and iBGP

IPv6 Addressing Plans.

Addressing plans in IPv6 are somewhat different than what has been considered the norm for IPv4. The IPv4 system is based around the RIRs allocating address space to an LIR (an ISP who is a member of the RIR) based on the needs of the ISP; that allocation is intended to be sufficient for a year of operation without returning to the RIR. The ISP is expected to implement a similar process towards their customers – so assigning address space according to the needs of the customer.

The system changes a little for IPv6. While the RIRs still allocate address space to their membership according to their membership needs, the justification to receive an IPv6 allocation is somewhat lighter than it is for IPv4. If the ISP can demonstrate a plan to connect 200 customers to the Internet using IPv6, they will receive an allocation. However, a bigger advantage starts with the customer assignments made by the ISP – the ISP simply has to assign a /48 to each of their customers. This is the minimum assignment for any site/customer – within this /48 there are a possible 64k subnets, deemed sufficient for all but the largest networks around these days. Within this /48, the smallest unit which can be assigned is a /64 – so every LAN and point to point link receives a /64.

With this revised system, the address plan for IPv6 is hugely simplified. ISPs assign a single /48 for their network infrastructure, and the remainder of their /32 block is used for customer assignments. This workshop assumes this principle, as will be seen in the following steps.

IPv6 Addresses

We are going to assume that each group is an ISP. For the purpose of this supplement we are going to assume that the minimum allocation for this workshop is a /32 (the current IPv6 allocation made by the Regional Internet Registries). So each group receives a /32 worth of address space. The groups running Routers 3 and 10 receive a /31 worth address space – not because they are special, but because it helps with the exercises later on in the workshop. The following table lists the address allocations made.

```
R1  1001:10::/32
R2  1001:11::/32
R3  1001:12::/31
R4  1001:20::/32
R5  1001:21::/32
R6  1001:22::/32
R7  1001:23::/32
R8  1001:30::/32
R9  1001:31::/32
R10 1001:32::/31
R11 1001:40::/32
R12 1001:41::/32
R13 1001:42::/32
R14 1001:43::/32
```

Back to Back Serial Connections

Each team now needs to assign IPv6 addresses to the serial connections between the routers. Each /32 allows an ISP to assign /48 subnets to customers and their own infrastructure. It is recommended that each group takes one /48 address block and uses that for their infrastructure needs in the lab. Within each IPv6 /48 assignment there are a possible 65536 subnets (the smallest address unit per LAN or point to point link is a /64). Each router team should use the last /48 out of its /32 address block for its infrastructure. For example, Router 5 will use 1001:21:FFFF::48 for numbering its "infrastructure".

Note that the lab will not use EUI-64 interface addressing, but assign absolute addressing to each interface. The latter is easier to manage, easier to handle for managing point to point peers and neighbour relationships.

A sample configuration might look like:

```
Router2(config)# interface serial 0/0
Router2(config-if)# ipv6 address 1001:11:ffff:1::1/64
```

The above also work with Quagga. I.e you do not need to exit to FreeBSD to do address assignments.

Q: What network mask should be used on all IPv6 enabled interfaces?
A: The network mask should be /64. This is the subnet size used for all LANs, point to point links, and so on. While some providers would wish to subdivide

the /64 even further, this is counter to RFC3513 which specifies the IPv6 addressing architecture. So all point to point links use a /64, all LANs use a /64, etc.

Ethernet Connections

As for the previous step, assign IPv6 addresses to the Ethernet point to point connections. Note that for IPv4 we were prudent in our assignments. For IPv6 the defined LAN subnet size is a /64. No larger, no smaller.

Ping Test #1

Ping all physically connected subnets of the neighbouring routers. If the physically connected subnets are unreachable, consult with your neighbouring teams as to what might be wrong. Don't ignore the problem – it may not go away. Use the following commands to troubleshoot the connection:

show ipv6 neighbors : Shows the ipv6 neighbour cache
show ipv6 interface <interface> <number> : Interface status and configuration
show ipv6 interface : Summary of IP interface status and configuration

Assign IPv6 Addresses to Loopback Interfaces

While no router feature yet uses the IPv6 address of the Loopback interface, it is still useful to configure an IPv6 address here. It will be used for the iBGP peering later on in this lab. Note that OSPF and BGP router IDs are 32 bit integers and in IOS (And Quagga) these are derived from the IPv4 address assigned to the Loopback interface (this has potential issues on network devices with no IPv4 address configured).

Q. Why do you think this is a problem? Ask the lab instructors to discuss.

As the minimum subnet size possible for IPv6 is a /64, we will assign the last /64 out of our /48 infrastructure block to be used for loopbacks. Here are suggestions for the network blocks each router team should use for loopback addresses (these lab notes assume these assignments in all examples):

```
R1  1001:10:ffff:ffff::/64
R2  1001:11:ffff:ffff::/64
R3  1001:13:ffff:ffff::/64
R4  1001:20:ffff:ffff::/64
R5  1001:21:ffff:ffff::/64
R6  1001:22:ffff:ffff::/64
R7  1001:23:ffff:ffff::/64
R8  1001:30:ffff:ffff::/64
R9  1001:31:ffff:ffff::/64
R10 1001:33:ffff:ffff::/64
R11 1001:40:ffff:ffff::/64
R12 1001:41:ffff:ffff::/64
R13 1001:42:ffff:ffff::/64
R14 1001:43:ffff:ffff::/64
```

From these blocks, each group should then pick ONE address to be used on the router they are managing. For the purpose of this workshop, the first address in the block should be used (so the xxxxxx:ffff::1/128 host address). For example, Router Team 1 would assign the following address and mask to the loopback on Router 1:

```
Router1(config)#interface loopback 0
Router1(config-if)#ipv6 address 1001:10:ffff:ffff::1/128
```

HINT: It might be a very good idea to record the router configuration as it is just now as this entire workshop will return to this configuration several times through out the coming days. Do not say you were not warned!

OSPF within the same AS. Each router Team should enable OSPF for IPv6 on their router. For the labs, the OSPF process identifier should be 41 (see example). (The OSPF process identifier is just a number to uniquely identify this OSPF process on this router. It is not passed between routers.) IPv6 OSPF is implemented slightly differently from the IPv4 counterpart in IOS – the latter made use of network statements to both find adjacencies and inject prefixes into the OSPF Link State Database. For IPv6, setting up the basic OSPF process is an independent configuration activity, such as in the example below. Note that all interfaces should be marked as passive by default:

```
Router1(config)#ipv6 router ospf 41
Router1(config-rtr)#passive-interface default
```

These commands above will only work for Cisco IOS, and not yet for Quagga.

Once the OSPF process is running, the interfaces whose network link addresses are required in the OSPF Link State Database should be configured. This is done by going to the actual interface, and attaching it to the OSPF process, as this example shows:

```
Router1(config)#interface loopback 0
Router1(config-if)#ipv6 ospf 41 area 0
!
Router1(config)#interface serial 0/0
Router1(config-if)#ipv6 ospf 41 area 0
!
...etc...
```

For Quagga the equivalent commands are

```
Host1(config)#router ospf6
Host1(config-opsf6)# area 0.0.0.0 range 2001:670:87:1::/64
Host1(config-ospf6)# interface vr0 area 0.0.0.0
...etc...
```

Finally, those interfaces over which we expect to form OSPF adjacencies should be marked as active interfaces. For this, we return to the main OSPF router process and mark those interfaces with the no passive-interface subcommand:

```
Router1(config)#ipv6 router ospf 41
Router1(config-rtr)#no passive-interface serial 0/0
Router1(config-rtr)#no passive-interface ethernet 0/0
...etc...
```

Note that the loopback interface has no “network” attached to it, so there is no way it can be connected to another device in such a way as to form an OSPF adjacency.

OSPF Adjacencies

Enable logging of OSPF adjacency changes. This is so that a notification is generated every time the state of an OSPF neighbour changes, and is useful for debugging purposes:

```
Router2(config)#ipv6 router ospf 41
Router2(config-rtr)#log-adjacency-changes
```

The above is Cisco IOS specific.

(Optional). Enable OSPF name lookups on the routers

The following is IOS specific. Following from the previous step, now enable OSPF name lookups on the router.

```
Router2(config)#ipv6 ospf name-lookup
```

This command enables the display of the OSPF router-ids as domain names. So, rather than displaying the following output with name lookups disabled:

```
router2>sh ipv6 ospf neigh

Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
100.1.15.224      1   FULL/BDR        00:00:36    1               Ethernet0/0
100.2.15.224      1   FULL/  -        00:00:32    2               Serial0/0
100.4.63.224      1   FULL/DR         00:00:38    3               Ethernet0/1
```

the router will display the following:

```
router2#sh ipv6 ospf neigh

Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
router1.worksho   1   FULL/BDR        00:00:33    1               Ethernet0/0
router4.worksho   1   FULL/  -        00:00:39    2               Serial0/0
router14.worksh   1   FULL/DR         00:00:35    3               Ethernet0/1
```

which is much more informative.

Ping Test #2

Ping all loopback interfaces in the classroom. This will ensure the OSPF IGP is connected End-to-End. If there are problems, use the following commands to help determine the problem:

`show ipv6 route` : see if there is a route for the intended destination
`show ipv6 ospf` : see general OSPF information
`show ipv6 ospf interface` : Check if OSPF is enabled on all intended interfaces
`show ipv6 ospf neighbor` : see a list of OSPF neighbours that the router sees

Checkpoint #1: call lab assistant to verify the connectivity. Save the configuration as it is on the router – use a separate worksheet, or the workspace at the end of this Module. You will require this configuration several times throughout the workshop.

Configuring iBGP Neighbours (Part 1)

All Routers are still in Autonomous System (AS) 10 for this first lab. The extension for BGP which allows it to support multiple protocols is known as an address family. IPv4 unicast is one of the many address families supported – it's the one we are most familiar with. IPv6 is another address family supported by the multiprotocol BGP, and we must configure the peerings to belong to the IPv6 address family. Before we actually configure each IPv6 iBGP peer, we need to disable the router's assumption that all BGP peers are IPv4 unicast. To do this, we use the command as in the example below:

```
Router4(config)#router bgp 10
Router4(config-router)#no bgp default ipv4-unicast
```

Configuring iBGP neighbours (Part 2)

Now we can configure the IPv6 iBGP neighbours, as in the example for Router 4 below. Use the `show bgp ipv6 summary` to check the peering. The BGP peering will be established using the loopback interface's IP address.

```
Router4(config)#router bgp 10
Router4 (config-router)#address-family ipv6
Router4 (config-router-af)#neighbor 1001:10:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:10:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:10:ffff:ffff::1 description iBGP
with Router1
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:11:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:11:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:11:ffff:ffff::1 description iBGP
with Router2
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:13:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:13:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:13:ffff:ffff::1 description iBGP
with Router3
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:21:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:21:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:21:ffff:ffff::1 description iBGP
with Router5
```

```
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:22:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:22:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:22:ffff:ffff::1 description iBGP
with Router6
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:23:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:23:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:23:ffff:ffff::1 description iBGP
with Router7
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:30:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:30:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:30:ffff:ffff::1 description iBGP
with Router8
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:31:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:31:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:31:ffff:ffff::1 description iBGP
with Router9
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:33:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:33:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:33:ffff:ffff::1 description iBGP
with Router10
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:40:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:40:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:40:ffff:ffff::1 description iBGP
with Router11
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:41:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:41:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:41:ffff:ffff::1 description iBGP
with Router12
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:42:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:42:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:42:ffff:ffff::1 description iBGP
with Router13
Router4 (config-router-af)#
Router4 (config-router-af)#neighbor 1001:43:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:43:ffff:ffff::1 update-source
loopback 0
```

```
Router4 (config-router-af)#neighbor 1001:43:ffff:ffff::1 description iBGP
with Router14
```

For Quagga, you will need to configure the remote-as as part of the Global BGP configuration. Such as

```
Router4(config)#router bgp 10
Router4 (config-router-af)#neighbor 1001:10:ffff:ffff::1 remote-as 10
Router4 (config-router-af)#neighbor 1001:10:ffff:ffff::1 update-source
loopback 0
Router4 (config-router-af)#neighbor 1001:10:ffff:ffff::1 description iBGP
with Router1
```

Q. Why is update-source loopback 0 necessary on iBGP?

`Use show bgp ipv6 summary` to check the status of the IPv6 iBGP neighbour connections. If the iBGP session is not up and/or no updates are being sent, work with the group for that neighbour connection to troubleshoot the problem.

Sanity Check

Remember to use the following commands to ensure you are getting the information you are suppose to be getting:

show ipv6 ospf : see general IPv6 OSPF information
show ipv6 ospf interface : see a list of IPv6 OSPF interfaces that the router sees
show ipv6 ospf neighbor : see a list of IPv6 OSPF neighbours that the router sees
show ipv6 ospf database : see IPv6 OSPF link state database that the router has learned
show bgp ipv6 summary : see a list of BGP IPv6 peers that the router sees
show bgp ipv6 : see a list of BGP IPv6 paths that the router sees
show ipv6 route : see all the IPv6 routes that the router has installed

Q. Are there routes seen via show bgp ipv6? If not, why not? Are there any routes tagged "B" when you do a show ipv6 route?

Add Networks via BGP

Each Router Team will use BGP to advertise the address block assigned to them earlier in the Module. For example, Router Team 1 would add:

```
Router1 (config)#router bgp 10
Router1 (config-router)#address-family ipv6
Router1 (config-router-af)#network 1001:10::/32
```

Use show bgp ipv6 on neighbour's router to see if you are advertising your network via BGP.

Q. Does the network show up via BGP? If not, why?

Enter a static route for the CIDR block. For example, Router 1 would use:

```
Router1 (config)#ipv6 route 1001:10::/32 Null0
```

Q. Does the network show up via a neighbour's BGP? Use the command `show bgp ipv6 neighbor <neighbour's IP address> advertised-routes` to see what you are exporting to the other router. Physically go to one of your neighbour's routers and check their BGP Table. Explain what you see.

Q. Does the network appear in the router's forwarding table? Use the command show ip route to check the local forwarding table. If not, why not?

16. Add the following commands to BGP:

```
Router1 (config)#router bgp 10
Router1 (config-router)# address-family ipv6
Router1 (config-router)# no synchronization
```

For Quagga :

```
Router1 (config)#router bgp 10
Router1 (config-router)# no synchronization
```

Q. Does the network appear in the router's forwarding table? Use the command `show ip route` to check the local forwarding table. What does the no synchronisation command do in BGP? How does it effect the router's forwarding table?

Note: As from IOS 12.3, synchronization is disabled by default and do not appear in the default BGP configuration. These two features have not been

required in Service Provider networks since the classless routing system was introduced to the Internet in 1994.

Checkpoint #2 : call the lab assistant to verify the connectivity.

Traceroute to all routers.

Once you can ping all the routers, try tracing routes to all the routers using trace x.x.x.x command. For example, Router Team 1 would type:

```
Router1# trace 1001:41:ffff:ffff::1
```

to trace a route to Router R12. If the trace times out each hop due to unreachable destinations, it is possible to interrupt the traceroute using the Cisco break sequence CTRL-^.

Q. Why do some trace paths show multiple IP addresses per hop?

A. If there are more than one equal cost paths, OSPF will "load share" traffic between those paths.

```
Router1>trace 1001:41:ffff:ffff::1

Type escape sequence to abort.
Tracing the route to 1001:41:ffff:ffff::1

  1 1001:10:ffff::2   4 msec
    1001:10:ffff:2::2 0 msec
    1001:10:ffff::2   0 msec
  2 1001:42:ffff::2   4 msec
    1001:11:ffff:2::2 4 msec
    1001:42:ffff::2   0 msec
  3 1001:43:ffff::2   4 msec *  4 msec
Router1>
```

Other Features in OSPF and BGP

Review the documentation or use command line help by typing ? to see other show commands and other OSPF and BGP configuration features.

Review Questions

1. What IOS show command(s) will display the router's IPv6 forwarding table?

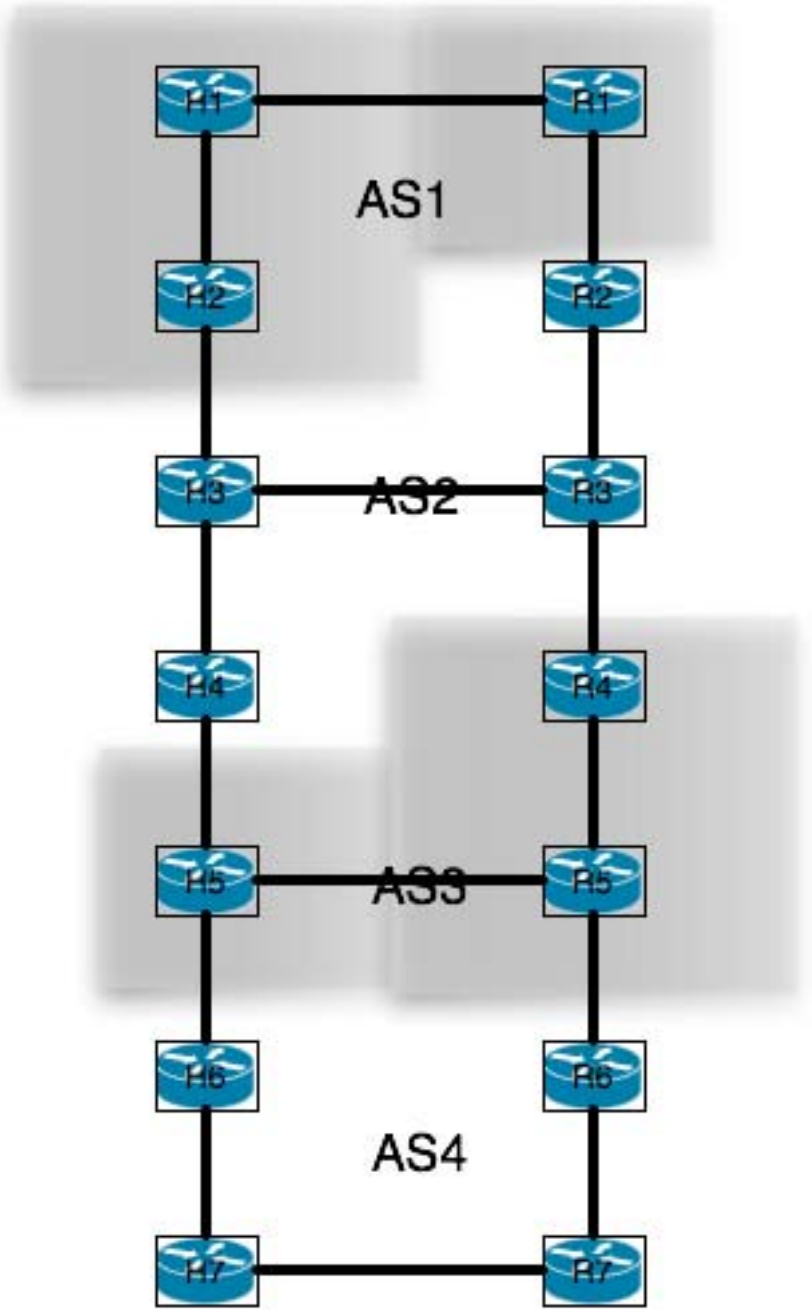
2. What IOS show command(s) will display the router's IPv6 OSPF database?

3. What IOS show command(s) will display the router's IPv6 BGP route table?

5.2. iBGP and Basic eBGP

The purpose of this chapter is to introduce you to external BGP (eBGP). This is the relationship between different autonomous systems in an "Internet". The classroom is split into four distinct networks, and the teams belonging to each network work together as a typical ISP. Each AS has two links to its neighbouring ASes, and this feature will used throughout a significant portion of this workshop.

The connectivity shown in the diagrams represents links between AS's. It is assumed that all the routers within an AS are physically connected to each other as per the figure below.



First connect the routers as shown above. All routers within an AS must be physically connected and reachable. The relationship between the ASes is as drawn in Figure 2 and gives a view which can be related to the “real world”.

The address assignments and addresses used for links between routers should be left the same as those chosen for chapter 5.1.

Re-configure IPv6 OSPF.

On each router, remove IPv6 OSPF processes if they are still present from chapter 5.1 by using the following command:

```
Router1 (config)# no ipv6 router 41
```

For the hosts running Quagga you need to do

```
Router1 (config)# no router ospf6
```

This will clear the IPv6 OSPF configuration for the current module.

Each group will also need to remove the IPv6 OSPF instances from the interfaces which were running OSPF in chapter 1. (This is if the instructor didn't already request this to be done for the main part of chaper 1.)

Configure IPv6 OSPF on the routers within each AS. In each AS configure IPv6 OSPF routing. This means that networks for the links to each member in the AS must be entered in OSPF via the network statement. All the routers in an AS will be in the same OSPF area 0 and use the same OSPF process ID. For example, Router 1, with two interfaces connecting to other routers in their AS, would have the following:

```
ipv6 router ospf 1
 passive-interface default
 no passive-interface ethernet 0/0
 no passive-interface serial 0/0
 log-adjacency-changes
!
interface ethernet 0/0
 ipv6 ospf 1 area 0
interface ethernet 0/1
 ipv6 ospf 1 area 0
interface serial 0/0
 ipv6 ospf 1 area 0
interface loopback0
 ipv6 ospf 1 area 0
!
```

And for Quagga :

```
!
router ospf6
 interface vr0 area 0.0.0.0
 interface vr1 area 0.0.0.0
 interface vr2 area 0.0.0.0
!
```

Notes:
• Passive-interface default makes sure that OSPF does not attempt to set up adjacencies on any interfaces apart from those specified by the no passive-interface commands.
• The number following "ipv6 router ospf" is a process ID and is used inside the router only (so it can be any number). But for this lab we recommend the OSPF process ID be the same as the AS number (which is the convention used by a number of ISPs). Quagga does not have the concept of process number.

Ping Test

Check the IPv6 routes known via OSPF. Make sure you can see all the networks within your AS. Ping all loopback interfaces within your AS. Use the "`show ipv6 ospf neighbor`" and "`show ipv6 route`" commands. If you cannot see the other routers in your AS, you will not be able
to bring up BGP in the next steps.

Save the configuration!!!

Checkpoint #1 : call the lab assistant to verify the connectivity.

Configure iBGP peering between routers within an AS.

Use the loopback address for the iBGP peerings. Also, configure the network command to add the address block assigned to each Router Team for advertisement in BGP.

```
router bgp 1
address-family ipv6
```

```
 no synchronization
 network 1001:10::/32
 neighbor 1001:11:ffff:ffff::1 remote-as 1
 neighbor 1001:11:ffff:ffff::1 update-source loopback 0
 neighbor 1001:11:ffff:ffff::1 description iBGP Link to R2
 neighbor 1001:13:ffff:ffff::1 remote-as 1
 neighbor 1001:13:ffff:ffff::1 update-source loopback 0
 neighbor 1001:13:ffff:ffff::1 description iBGP Link to R3
!
```

And for Quagga :

```
router bgp 1
 no synchronization
 network 1001:10::/32
 neighbor 1001:11:ffff:ffff::1 remote-as 1
 neighbor 1001:11:ffff:ffff::1 update-source loopback 0
 neighbor 1001:11:ffff:ffff::1 description iBGP Link to R2
 neighbor 1001:13:ffff:ffff::1 remote-as 1
 neighbor 1001:13:ffff:ffff::1 update-source loopback 0
 neighbor 1001:13:ffff:ffff::1 description iBGP Link to R3
```

And for both :

```
ipv6 route 1001:10::/32 Null0
```

Q: Do you need the BGP command no synchronization? Why?

A: An ISP network is a transit network, meaning it accepts packets from one peering AS, carries it across the backbone, then hands it over to the next AS toward the destination. To ensure that routers internal to the AS are able to forward transit packets (from the ingress border router to the egress border router), all BGP border routers will wait for a network prefix to arrive in the IGP (as they all participate in the same IGP) before advertising them to external BGP peers. This is referred to as synchronization. In other words, internal routers must be aware of those prefixes learned via IGP that border routers learn via iBGP.

As you can see, this applies to an environment where BGP routes are redistributed into the IGP. A typical ISP usually doesn't do that as the Internet routing table is somewhat large. Instead, it runs a fully meshed iBGP (or uses route-reflectors) among all routers in the backbone. Therefore synchronisation should be turned off in this kind of environment.

Test internal BGP connectivity.

Use the BGP Show commands to ensure you are receiving everyone's routes from within your AS.

Configure passwords on the iBGP sessions.

Passwords should now be configured on the iBGP sessions. Review the presentation why this is necessary. Agree amongst all your team members in your AS what the password should be on the iBGP session, and then apply it to all the iBGP peerings on your router. For example, on Router2's peering with Router3, with "cisco" used as the
password:

```
router bgp 1
address-family ipv6
 neighbor 1001:13:ffff:ffff::1 password cisco
```

IOS currently resets the iBGP session between you and your neighbouring router whenever an MD5 password is added. So when passwords are added to BGP sessions on live operational networks, this work should be done during a maintenance period when customers know to expect disruptions to service. In the workshop lab, it doesn't matter so much. (Future IOS releases will avoid having this rather serious service disruption.)

While Quagga hae rudimentary support for MD5 passwords, that is not in use here.

Watch the router logs – with the BGP session neighbour changes being logged, any mismatch in the password should be easy to spot.

Checkpoint #2: Call the lab assistant and demonstrate the password as set on the iBGP session. Once confirmed by the lab assistant, move on to the next steps.

Configure eBGP peering

Use Figure 1 to determine the links between the AS's. Addressing for eBGP links between 2 AS's will use the point-to-point interface addresses, NOT the loopback addresses (review the BGP presentation if you don't understand why). So, for Router1's peering with Router13, the configuration might look like:

```
router bgp 1
 neighbor 1001:10:ffff:2::2 remote-as 4
 neighbor 1001:10:ffff:2::2 description eBGP to Router13
```

Use the BGP Show commands to ensure you are sending and receiving the BGP advertisements from your eBGP neighbours.

Q. Why can't the loopback interfaces be used for the eBGP peerings?

A. The IP address of a router's loopback interface is not known to external BGP peers, so the external peers will have no way of knowing how to contact each other to establish the peering.

Q. Which BGP show command allows you to see the state of the BGP connection to your peer?

A. Try `show bgp ipv6 neighbor x.x.x.x` – this will give detailed information about the state of the peer. There are subcommands of this one, giving more information about the peering.

Q. Which BGP Show command will allow you to see exactly which networks you are advertising and receiving from your eBGP peers?

A. Try `show bgp ipv6 neighbor x.x.x.x route` – this will show which routes you are receiving from your peer. Likewise, replacing route with advertised-routes will list the networks which are being announced to your peer. (Note that in general ISP operational practice, there are caveats here – if you apply route-maps and some BGP policies, these will not be processed by the advertised-routes command. Use the advertised-routes subcommand with due caution.)

Configure passwords on the eBGP session

Passwords should now be configured on the eBGP sessions between your and your neighbouring ASes. Agree between you and your neighbouring AS what the password should be on the eBGP session, and then apply it to the eBGP peering. For example, on Router2's peering with Router4, with "cisco" used as the password:

```
router bgp 1
```

```
neighbor 1001:11:ffff:1::2 password cisco
```

As previously for the iBGP session, watch the logs for password mismatches, or missing passwords. As with the iBGP sessions previously, you will find that the router will reset the eBGP session as soon as the password is applied.

Note: Wherever a BGP (either iBGP or eBGP) session is configured from now on in the workshop, all Router Teams MUST use passwords on these BGP sessions.

Checkpoint #3: Call the lab assistant and demonstrate the password as set on the eBGP session. Once confirmed by the lab assistant, move on to the next steps.

Aggregate each AS's CIDR Blocks

Each router team was allocated either a /32 address block or a /31 address block in the first Module. However, each AS has three or four routers in it, so we need to take the address space from each router team in the AS and aggregate it before we make any announcement to any external AS. It is expected by all Internet operators that any address space an ISP is using is aggregated as much as possible before it is announced to the rest of the Internet. Subdividing the address space inside an AS is perfectly acceptable and obviously very common – but leaking this subdivided address space out to the Internet at large is considered antisocial and unfriendly by many ISPs. In this case the address blocks belonging to each AS can be aggregated into a larger /18 address block.

For example, AS1 has three routers in it. Router 1 was allocated 1001:10::/32, Router 2 was allocated 1001:11::/32 and Router 3 was allocated 1001:12::/31. These three address blocks can be aggregated into the 1001:10::/30 network. And this /30 is what should be announced to eBGP neighbours.

Q. How do you automatically aggregate via BGP smaller address blocks from within your
network to a larger address block outside your network? Hint: Review the BGP documentation.

A. Configure:

```
router bgp 1
 address-family ipv6
 aggregate-address 1001:10::/32
```

Type ? after the command to see what options this command has. Is this command supported on Quagga?

Examine the origin of the network prefixes

What is the origin type for the aggregated prefixes?
Write your answer here:

Check the network paths

Do traceroutes to hosts nominated on the network by the lab instructor.

Checkpoint #4: Call the lab assistant to verify the connectivity. Use commands such as "`show ip route sum`", "`show bgp ipv6 sum`", "`show bgp ipv6`", "`show ipv6 route`", and "`show bgp ipv6 neigh x.x.x.x route | advertise`". There should be 13 specific prefixes and 4 aggregate prefixes (one for each ISP).

Review Questions

1. How many origin types exist in BGP?

2. List the origin types. Hint: Review the BGP presentations.

3. How are they used?

4. Why are passwords necessary on both iBGP and eBGP sessions? What do they protect against?

5. Why is aggregation important for the Internet?

5.3. IPv6 route filtering and advanced BGP

The main part of this Module provided an introduction to some more advanced features of BGP, including an introduction to the types of routing policy which are available for BGP. The topology will be the same as in 5.2.

In steps 2) we will configure each AS to become a non-transit AS from the IPv6 perspective, i.e. the AS won't allow a connecting AS's traffic to traverse it to reach another connecting AS. Recall we already completed this exercise for IPv4 earlier. This will mean disconnectivity in the classroom – for example, AS3 will no longer be able to see any networks from AS1, etc. This is a deliberate policy to demonstrate the effectiveness of the BGP filtering being employed. We will achieve this by configuring an outgoing route filter to allow only local prefix(es) be sent to eBGP peers. At the same time, we also make sure that our peers only send us their own prefixes. We guarantee this by configuring an incoming filter. In general, it is good practice to configure filters in both directions to protect against misconfiguration at both ends of the peering.

Important: each step must be carried out and completed by the whole workshop before the next step can be started. Do not immediately start the next step without receiving the go-ahead from the workshop instructors. If you do, it is likely the routing will break, and other router teams may be unable to understand the results of the configuration they are trying to implement.

Important: retain the configuration used for the main part of chapter 5.2, but remove the aggregate address line from the IPv6 address family of the BGP configuration.

Implement BGP policies

Before doing any configurations in this module it is important to note how to go about implementing BGP policies. Entering prefix lists, as-path filters or route-maps can be done at the CLI, but they only apply to BGP updates received after the policy configuration has been entered at the router. This is because BGP sends incremental updates describing changes in routes announced or withdrawn. To apply the policy to the entire BGP routing table received from or sent to the peer, the BGP session needs to be "reset". In older versions of IOS, this literally meant tearing down the BGP session, and then restoring it. However, as can be imagined, this causes severe stability issues on the service provider network, and 'RFC2918: Route Refresh Capability' has been added to most modern BGP implementations to allow graceful updates to BGP sessions when policy changes are required.

To implement policy changes in any of the following worked examples, use the following router commands, for example to implement new policy inbound and outbound on the peering between Router 1 and Router 13:

```
Router1# clear bgp ipv6 1001:10:ffff:2::2 out
Router1# clear bgp ipv6 1001:10:ffff:2::2 in
```

Note: Do not forget the in and out subcommands in the above clear commands – omitting them will implement a hard reset of the BGP session. Review the BGP presentation if you don't understand why you do not want to ever do a hard reset on a BGP session.

Configure prefix filter based on network address: This step configures route prefix filtering based on network address. This is done using prefix-lists, and is one method of controlling networks which are exchanged in BGP peerings. The aim here is to configure eBGP peerings so that only networks from neighbouring ASes are exchanged.

Note in the example below that the IPv6 prefix-list has been given the name "v6-out-peer". Recall from the main part of this module that the IPv4 prefix-list was called "out-peer". The difference in names is for operator convenience and confusion reduction only. The router will only apply an IPv6 prefix-list to an IPv6 address family – so it knows. But when an operator is confronted with two lists of the same name, but referring to different address families, it is quite possible in the heat of the moment for a mistake to be made. So follow the simple rule: use simple, descriptive and meaningful names for prefix-lists, whether they are IPv4 or IPv6.

Example: Router R13 (peering with R1)

```
!
ipv6 prefix-list v6-out-peer permit 1001:40::/30 le 32
ipv6 prefix-list v6-out-peer deny ::/0 le 128
!
ip prefix-list v6-in-peer permit 1001:10::/30 le 32
ip prefix-list v6-in-peer deny ::/0 le 128
!
router bgp 4
address-family ipv6
 no synchronization
 network 1001:42::/32
 neighbor 1001:10:ffff:2::1 remote-as 1
 neighbor 1001:10:ffff:2::1 description eBGP peering with R1
 neighbor 1001:10:ffff:2::1 prefix-list v6-out-peer out
 neighbor 1001:10:ffff:2::1 prefix-list v6-in-peer in
!
```

Example: Router R1 (peering with R13)

```
!
```

```
ip prefix-list v6-out-peer permit 1001:10::/30 le 32
ip prefix-list v6-out-peer deny ::/0 le 128
!
ip prefix-list v6-in-peer permit 1001:40::/30 le 32
ip prefix-list v6-in-peer deny ::/0 le 128
!
router bgp 1
address-family ipv6
 no synchronization
 network 1001:10::/32
 neighbor 1001:10:ffff:2::2 remote-as 4
 neighbor 1001:10:ffff:2::2 description Peering with Router13
 neighbor 1001:10:ffff:2::2 prefix-list v6-out-peer out
 neighbor 1001:10:ffff:2::2 prefix-list v6-in-peer in
!
```

And for quagga :

```
ip prefix-list v6-out-peer permit 1001:10::/30 le 32
ip prefix-list v6-out-peer deny ::/0 le 128
!
ip prefix-list v6-in-peer permit 1001:40::/30 le 32
ip prefix-list v6-in-peer deny ::/0 le 128
!
router bgp 1
 no synchronization
 network 1001:10::/32
 neighbor 1001:10:ffff:2::2 remote-as 4
 neighbor 1001:10:ffff:2::2 description Peering with Router13
address-family ipv6
 neighbor 1001:10:ffff:2::2 prefix-list v6-out-peer out
 neighbor 1001:10:ffff:2::2 prefix-list v6-in-peer in
!
```

Note: an IOS prefix-list always has an implicit deny any as the last statement even though it is not listed in the configuration. Some ISPs add the implicit deny any as they consider it good practice and a security precaution.

Note: these prefix-lists are only applied to peerings with other ASes. These are called external peerings (using eBGP). There is usually no need to apply such filters for iBGP peerings.

Checkpoint #1: call the lab assistant to verify the connectivity. Each router team should check peerings to see the effect of this step. Use the "`show bgp ipv6 neigh x.x.x.x advertise|route`" commands. Once complete and the lab instructors give you the go-ahead, remove the prefix-list configuration, and carry on with step 3.

Remove configuration from previous example. This step will demonstrate how to remove the configuration entered in the previous example. This is essential before we move onto the next step.

Example: Router R1

```
Router1#conf t
Router1(config)#router bgp 1
Router1(config-router)#address-family ipv6
!
! First remove prefix list from BGP peering with R13
!
Router1(config-router-af)#no neighbor 1001:10:ffff:2::2 prefix-list v6-out-peer out
Router1(config-router-af)#no neighbor 1001:10:ffff:2::2 prefix-list v6-in-peer in
!
! Now remove the prefix-lists themselves
!
Router1(config)#no ipv6 prefix-list v6-out-peer
Router1(config)#no ipv6 prefix-list v6-in-peer
!
! That's the configuration nice and tidy, the way it should be.
!
Router1(config)#end
!
! Now clear the bgp peering so that the old policy is removed
!
Router1#clear bgp ipv6 1001:10:ffff:2::2 out
Router1#clear bgp ipv6 1001:10:ffff:2::2 in
Router1#
```

Setting BGP Communities

As we saw in the main part of this module, ISPs attach communities to a network when the network is injected into the BGP routing table.

Each router team should assign a community to the network block they have been allocated in Module 1. Review the BGP documentation to find out how to do this. Each router should set a community of format [AS number]:[Router number] exactly as in the previous step.

Example for Router R1:

```
ip bgp-community new-format
!
```

```
route-map community-tag permit 10
  set community 1:1
 !
 router bgp 1
 address-family ipv6
  no synchronization
  network 1001:10::/32 route-map community-tag
  neighbor 1001:10:ffff:2::2 remote-as 4
  neighbor 1001:10:ffff:2::2 send-community
 !
 ipv6 route 1001:10::/32 null0
```

Check that the network appears with its community in the BGP routing table.

Q: Why do your external, but not your internal, peers see the community set on the network?
A: See earlier. All peerings require the BGP send-community subcommand for the community attribute to be sent between BGP peers.

Communities on internal BGP peerings

Following on from the previous step, now set up the internal peerings so that the community attribute for your network is sent to local peers.

Hint: to do this, simply add in the configuration line neighbor x.x.x.x send-community for all iBGP peerings. Don't forget to refresh the BGP peering sessions so that the configuration change can be implemented.

Checkpoint #5: call the lab assistant and demonstrate how the community has been set for your network using the "show ip bgp" commands. Also, demonstrate that you can see the communities set by your internal and external BGP peers.

Configure incoming prefix filter based on community attribute

The aim here is to only accept networks which are received from the neighbouring external BGP peer. (This is similar to what was being attempted in Steps 2 and 4 with prefix and AS path filtering.) For example, R13 should only accept the network originated by R1, and should use the knowledge of the community R1 has attached to the network to achieve this.

Example on Router R13:

```
route-map infilter permit 10
match community <community-list-no>
!
ip community-list <community-list-no> permit 1:1
!
router bgp 4
 address-family ipv6
 neighbor 1001:10:ffff:2::2 route-map infilter in
```

The <community-list-no> choice is up to each router team – it is not announced in any BGP peering or used in any other way apart from identifying the community-list (compare with the access-list number).

Set local-preference attribute on received eBGP routes

In this example, local preference will be set on the routes matching the community filter in Step 10. Retain the route-map used in Step 10 – an additional configuration line will be added to it. You also want to allow the other networks heard through the filters with the local-preference set to the default value.

Q. Why?
A. Without the second permit directive the route-map implements a default deny and no other prefixes will be passed through.

Example:

```
route-map infilter permit 10
 set local-preference 120
!
route-map infilter permit 20
```

Note that after a new policy is set, BGP session needs to be refreshed so that the new policy can then be applied. The router does not automatically keep all the updates it ever received from the peer so this is necessary. This is done by using "`clear ip bgp <peer address> in`" exec command.

Checkpoint #6: call the lab assistant and demonstrate how the routes originated by your eBGP peers now have local preference set to 120. Also show how other routes have the default local preference of 100.

Converting the CLI from IPv4 unicast centric format to address family format

BGP was first extended to support multicast as DVMRP was no longer sufficient to provide the necessary transport of routing information for multicast. At the same time, BGP was being extended to support the IPv6 routing information. And the IOS CLI grew "organically" to support these different extensions. It became very clear that this organic growth was not sustainable, hence in early 2002 it was decided to unify the CLI and adopt the address family approach for all "versions" of BGP.

The default BGP CLI in use on each router at the moment is the "old format". Each router team is now going to upgrade the CLI to the "new format". Before doing so, save the configuration in to NVRAM. And then do a "show run | b bgp" – which shows the running configurations first instance of the word "bgp". What do you see? Write your answer here:

Converting the BGP CLI

The following exercise is Cisco IOS specific. In this exercise we will now convert the BGP CLI. The command sequence to do this is as follows:

```
Router2(config)#router bgp 1
Router2(config-router)#bgp upgrade-cli
```

The "upgrade-cli" option to BGP upgrades the command line interface from the old format (which will soon become obsolete) to the new format which supports multiple address family formats in an easy to read style. When the router asks if "you are sure", the answer is "yes".

```
 You are about to upgrade to the AFI syntax of bgp commands

 Are you sure ? [yes]: yes
Router2(config-router)#
```

Comparing old and new CLI formats for BGP

Now save the configuration and inspect what has happened to your existing IPv4

BGP configuration. Compare what you see now with what you wrote down above. You will see that it has been "divided" into two pieces. The first part describes the generic BGP neighbour configuration which would apply to all address families, with the remaining sections having configuration which is specific to each address family. This is an excerpt from what the team operating Router 1 might expect to see:

```
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 100.1.31.224 remote-as 1
 neighbor 100.1.63.224 remote-as 1
 neighbor 1001:11:ffff:ffff::1 remote-as 1
 neighbor 1001:11:ffff:ffff::1 remote-as 1
 ...etc...
 no auto-summary
 !
 address-family ipv4
 neighbor 100.1.31.224 activate
 neighbor 100.1.63.224 activate
...etc...
 no auto-summary
 no synchronization
 exit-address-family
!
address-family ipv6
 neighbor 1001:11:ffff:ffff::1 activate
 neighbor 1001:13:ffff:ffff::1 activate
..etc...
 no synchronization
 exit-address-family
```

The "activate" option in the address family specifies that this neighbour should be activated for this particular address family. Without the "activate" option, the neighbour is not activated for this address family.

Configure the peer-group feature for iBGP peers

BGP peer-groups help reduce the router processor load in sending updates to peers which have the same policy. This step configures BGP peer-groups for the iBGP peers in each AS. Replace the individual configuration for each iBGP peer with a peer-group configuration, as given in the example below.

Example for Router R9:

```
router bgp 3
address-family ipv6
 neighbor ibgp-peers peer-group
 neighbor ibgp-peers description Peer-Group used for all iBGP
peers
 neighbor ibgp-peers remote-as 3
 neighbor ibgp-peers update-source loopback 0
 neighbor ibgp-peers send-community
 neighbor 1001:30:ffff:ffff::1 peer-group ibgp-peers
 neighbor 1001:30:ffff:ffff::1 peer-group ibgp-peers
```

Note: the old pre-peer-group configuration must be removed when converting from non-peer-group configuration to using peer-groups.

It is strongly recommended that the peer-group subcommand be the "default" way of configuring all BGP peers. As was stated before, most iBGP peers have the same configuration, so it is of great benefit to the router, the operations staff, and network engineering staff to simplify the configurations wherever possible. Besides, a configuration which makes extensive use of peer-groups is usually much easier to read than one which has distinct configuration per peer, especially
in networks with large numbers of peers.

Note: Wherever BGP is configured in future in the workshop, it is expected that peer-groups will be used as part of the BGP configuration (and most definitely for any iBGP configuration).

Review Questions:

1. What is the reason for converting the BGP CLI from old format to new format?

2. What happens to the configuration once the CLI is converted?

3. Are there any differences between BGP's behaviour for IPv6 and IPv4 so far? If so, what?

5.4. IPv6 multihoming strategies lab

This chapter demonstrates how an AS can use Local Preference to control

outbound routing paths, and use AS path prepend and MEDs (multi-exit discriminators or metrics) to determine inbound routing paths. All three are very powerful tools for ISPs to control how their externalpeering links are used. Refer to the BGP documentation for more information about the BGP path selection process and the default values for, and priorities of, the “local_pref” and “metric” attributes.

Before starting this module, retain the topology and configurations as used in chapter 5.2. This requires the removal of all the filtering and community configurations examined in chapter 5.3.

Recommendation: Remember, if any configuration on a router is not in use, it should be removed. Surplus configuration usually gives rise to delayed error detection and debugging of configurations in cases of routing problems or other network failures.

Tidy up from Chapter 5.3

If the previous chapter completed was chapter 5.3, the router configuration needs to be tidied up substantially before this module is attempted. The following steps show exactly what is required.

Example: Router R1

```
Router1#conf t
Router1(config)#router bgp 1
!
! First remove BGP damping
!
Router1(config-router)#no bgp dampening
Router1(config-router)#address-family ipv6
!
! Now remove BGP neighbour route-map statement
!
Router1(config-router-af)#no neighbor 1001:10:ffff:2::2 route-map v6-infilter in
!
! Now remove community-tag from network statement
!
Router1(config-router-af)#no network 1001:10::/32 route-map v6-community-tag
Router1(config-router-af)#network 1001:10::/32
!
! Now remove route-maps
!
Router1(config)#no route-map v6-community-tag
Router1(config)#no route-map v6-infilter
```

```
!
! Now remove community list
!
Router1(config)#no ip community-list 10
!
! That's the configuration nice and tidy, the way it should be.
!
Router1(config)#end
!
! Now clear the bgp peering so that the old policy is removed
!
Router1#clear ip bgp 1001:10:ffff:2::2 in
Router1#
```

Advertising your AS's aggregate route

Make sure that your AS is sending an aggregate prefix
for all routes in the AS along with the more specific routes. Use the aggregate-address BGP
configuration command to generate the aggregate.

Example:

AS 1 will advertise 1001:10::/30 and its subnets to AS 2 and AS 4.
AS 4 will advertise 1001:40::/30 and its subnets to AS 1 and AS 3.
Etc...

Check routing tables and perform ping & traceroute tests to confirm connectivity. Test routing connectivity by disabling links to external peers and repeat the traceroute, this should show you the alternate path computed by BGP when the primary fails.

HINT: The configuration on your router should now be the same as it was at the end of Module 3 but without the community settings, route-maps and community-lists. Soft-configuration shouldn't be necessary, but if any router team wishes to retain it for troubleshooting purposes, they should do so.

Checkpoint #1: Call your lab instructor and display the following:

i] Output of a "`show ipv6 route`" and "`show bgp ipv6`"
ii] Outputs of the '`ping`' and '`trace`' to various IPv6 destinations within the network
iii] Outputs of the 'ping' and 'trace' after the primary link fails.

Aim of the Chapter

The aim of the module is to demonstrate how to achieve a particular traffic flow using three different methods. These three methods involve modification of outbound policy, and two ways of modifying inbound policy. The reader should review the BGP presentation given prior to this module as a reminder on how to influence path choice using BGP policies.

The diagram following (Figure 2) displays the traffic flows which are desired between particular routers and ASes. Six traffic flows are being implemented. The arrows in the figure show the flows which will be configured. Each arrow originates from a border router in an AS, and terminates on one of the routers in another AS. This signifies the traffic flows desired for the links between the two systems. Each of the following steps has a description on how to implement the traffic flow represented by each arrow. If at anytime there is any doubt as to the configuration required, consult the Cisco CD Documentation, or ask the lab instructors.

Implement the following Outbound policies

Discuss among teams in the same AS and negotiate with your neighbouring AS on how to implement each of the following policies on your routers. It is important that backup paths should still function. Based on the agreed method, configure the following primary paths:

AS 1:
- All traffic TO 1001:41::/32 (R12) must exit AS 1 via Router R1 only.

AS 2:
- All traffic TO 1001:30::/32 (R8) must exit AS 2 via Router R7 only.

AS 3:
- All traffic TO 1001:21::/32 (R5) must exit AS 3 via Router R8 only.
- All traffic TO 1001:42::/32 (R13) must exit AS 3 via Router R10 only.

AS 4:
- All traffic TO 1001:12::/31 (R3) must exit AS 4 via Router R14 only.

Note that we are only trying to define outgoing traffic flow. The return path has no policies implemented, and the router's normal path decision process applies.

Hints:

- Remember in step 1 above that the router was configured to announce the aggregate as well as the specific networks in each AS? Therefore, mindful of this fact, use "local preference" internally to influence the exit path out of your AS. Set the preferred router to a higher than default local preference, and the less preferred routers to a lower than default local preference.
- Using a route filter is not a good way of achieving the above policies. You still need to allow alternate path rerouting if failure happens. Using local preference permits this.
- Draw your AS topology and address block assignment on a blank piece of paper. Discuss within your router team, and within your AS, which team should be applying which configuration at the AS boundaries. It is important to decide upon strategy before typing configuration into the router.

The example configurations below should be used as guidance for the router configuration for each router team. Note that one router in each AS will have to set high local preference, the remaining routers in the AS will set low local preference. The motivation for doing this is redundancy of configuration. If, for example, Router 7 lost its local-preference configuration due to some operator error, the low local preference set on the other three routers will ensure that the traffic policies required will still be implemented. It's quite common for many ISPs to have more than one configuration to implement a particular policy – a primary configuration, and an "inverse" secondary configuration on other routers which could be impacted.

Example configurations for the AS 2 scenario above (using LOCAL_PREF).
Router 7:

```
ipv6 prefix-list R8-prefix permit 1001:30::/32
!
route-map set-local-pref-high permit 10
 match ipv6 address prefix-list R8-prefix
 set local-preference 200
!
route-map set-local-pref-high permit 20
!
router bgp 2
address-family ipv6
 neighbor 1001:31:ffff:1::1 remote-as 3
 neighbor 1001:31:ffff:1::1 route-map set-local-pref-high in
!
```

Router 4,5,6:

```
 ipv6 prefix-list R8-prefix permit 1001:30::/32
 !
route-map set-local-pref-low permit 10
 match ipv6 address prefix-list R8-prefix
 set local-preference 50
!
route-map set-local-pref-low permit 20
!
router bgp 2
address-family ipv6
 neighbor x.x.x.x remote-as ASN
 neighbor x.x.x.x route-map set-local-pref-low in
!
```

Checkpoint #2: Call your lab instructor and display the following:

1. Each router in an AS will be asked to do a 'trace' to selected destinations, the trace must show packets exiting the AS as specified in the exercise given above.

2. Explain your configuration used to achieve the desired result to the instructor. Display the output of "`show ip bgp`", and "`show ip bgp x.x.x.x`" for the networks with local preference set to 200. Show the output of a trace according to instructions.

3. Wait until the lab instructor gives the goahead to move onto the next step.

STOP AND WAIT HERE

Remove configuration from the previous step

Before moving on to the next step it is important that the configuration from the previous step is removed. This involves removing the route-maps, prefix-lists and the per-neighbour configuration to set local preference. All router teams should do this, and then do a soft reset of their eBGP peerings.

Implement the following Inbound policies using MEDs.

This step introduces one of two methods of influencing inbound policies. Here MEDs are used, while the next step will introduce the concept of AS path prepends. As for the previous step, read the instructions careful, and discuss in your team, and in your AS, how you are going to implement the following.

The example in this step achieves exactly the same traffic flow between neighbouring ASes as in the previous step for the networks in question – remember that local preference is used by an AS to influence outbound traffic paths, whereas MEDs are used to influence inbound traffic paths. Refer to Figure 2 for a picture of traffic flow...

AS 1:
- All traffic TO 1001:12::/31 (R3) from anywhere in AS 4 must enter AS 1 via the R14 – R2 link. (Hint: this means that R1 must announce 1001:12::/31 to AS 4 with a higher metric than the equivalent announcement from R2.)

AS 2:
- All traffic TO 1001:21::/32 (R5) from anywhere in AS 3 must enter AS 2 via the R8 – R6 link.
(Hint: this means that R7 must announce 1001:21::/32 to AS 3 with a higher metric than the equivalent announcement from R6.)

AS 3:
- All traffic TO 1001:30::/32 (R8) from anywhere in AS 2 must enter AS 3 via the R7 – R9 link. (Hint: this means that R8 must announce 1001:30::/32 to AS 2 with a higher metric than the equivalent announcement from R9.)

AS 4:
- All traffic TO 1001:41::/32 (R12) from anywhere in AS 1 must enter AS 4 via the R1 – R13 link. (Hint: this means that R14 must announce 1001:41::/32 to AS 1 with a higher metric than the equivalent announcement from R13.)

- All traffic TO 1001:42::/32 (R13) from anywhere in AS 3 must enter AS 4 via the R10 – R12 link. (Hint: this means that R11 must announce 1001:42::/32 to AS 3 with a higher metric than the equivalent announcement from R12.)

Example configurations for the AS 2 scenario above (using MED).
Router 6:

```
ipv6 prefix-list R5-prefix permit 1001:21::/32
!
route-map set-med-low permit 10
 match ipv6 address prefix-list R5-prefix
```

```
 set metric 10
!
route-map set-med-low permit 20
!
router bgp 2
address-family ipv6
 neighbor 1001:22:ffff:1::2 remote-as 3
 neighbor 1001:22:ffff:1::2 route-map set-med-low out
!
```

Router 7:

```
ipv6 prefix-list R5-prefix permit 1001:21::/32
!
route-map set-med-high permit 10
 match ipv6 address prefix-list R5-prefix
 set metric 50
!
route-map set-med-high permit 20
!
router bgp 2
 neighbor 1001:31:ffff:1::1 remote-as 3
 neighbor 1001:31:ffff:1::1 route-map set-med-high
out
!
```

Note: the teams operating Routers 3, 4, 5 and 10 have only to remove the configuration which was added in the previous step. They don't require to configure MEDs as above because they do not have peerings which require alterations to the inbound policy for their AS.

Checkpoint #3: Call your lab instructor and display the following:

- Each router in an AS will be asked to do a '`traceroute`' to selected destinations, the trace must show packets exiting the AS as specified in the exercise given above.

- Explain your configuration used to achieve the desired result to the instructor. Display the output of "`show ip bgp`", and "`show ip bgp x.x.x.x`" for the networks with MED set to 50. Show the output of a trace according to instructions.

STOP AND WAIT HERE

Remove the configuration used for the previous step

Before moving on to the next step it is important that the configuration from the previous step is removed. This involves removing the route-maps, prefix-lists and the per-neighbour configuration to set MEDs. All router teams should do this, and then do a soft reset of their eBGP peerings.

Implement the following inbound policies using the AS path prepend method

This step introduces the second of two methods of influencing inbound policies. As for the previous step, read the instructions carefully, and discuss within your team, and within your AS, how you are going to implement the following.

The example in this step achieves exactly the same traffic flow between neighbouring ASes as in the previous step for the networks in question. Refer to Figure 2 for a picture of traffic flow...

AS 1:
- All traffic TO 1001:12::/31 (R3) from anywhere in the lab topology must enter AS 1 via the R14 – R2 link. (Hint: this means that R1 and R3 must announce 1001:12::/31 with a longer AS path than the other networks in AS 1. R2 needs to announce 1001:12::/31 with a longer AS path in its peering with R4.)

AS 2:
- All traffic TO 1001:21::/32 (R5) from anywhere in the lab topology must enter AS 2 via the R8 – R6 link. (Hint: this means that R4, R5 and R7 must announce 1001:21::/32 with a longer AS path than the other networks in AS 2.)

AS 3:
- All traffic TO 1001:30::/32 (R8) from anywhere in the lab topology must enter AS 3 via the R7 – R9 link. (Hint: this means that R8 and R10 must announce 1001:30::/32 with a longer AS path than the other networks in AS 3. R9 needs to announce 1001:30::/32 with a longer AS path in its peering with R11.)

AS 4:
- All traffic TO 1001:41::/32 (R12) from anywhere in the lab topology must enter AS 4 via the R1 – R13 link. (Hint: this means that R11, R12 and R14 must announce 1001:41::/32 with a longer AS path than the other networks in AS 4.)

- All traffic TO 1001:42::/32 (R13) from anywhere in the lab topology must enter AS 4 via the R10 – R12 link. (Hint: this means that R11, R13 and R14 must announce 1001:42::/32 with a longer AS path than the other networks in AS 4.)

AS_PREPEND is commonly used by smaller ISPs who are multihoming to their upstream providers. It is convention on the Internet to add at least two ASes when using AS_PREPEND.

More usually, three ASes are added, especially if the upstream ISPs have links to each other going
through a third party.

Example configuration for the AS 3 scenario above (using AS PATH prepend):

```
ipv6 prefix-list R8-prefix permit 1001:30::/32
 !
route-map set-as-path permit 10
  match ipv6 address prefix-list R8-prefix
  set as-path prepend 3 3 3
 !
 route-map set-as-path permit 20
 !

 router bgp 3
 address-family ipv6
 neighbor 1001:22:ffff:1::1 remote-as 2
 neighbor 1001:22:ffff:1::1 route-map set-as-path out
 !
```

Note: the team with Router 6 has only to remove the configuration which was added in the previous step. They don't require to configure AS path prepends as above because they do not have peerings which require alterations to the inbound policy for their AS.

Checkpoint #4: Call the lab instructor and display the following:

- Each router in an AS will be asked to do a '`traceroute`' to selected destinations, the trace must show packets exiting the AS as specified in the exercise given above.

- Explain your configuration used to achieve the desired result to the instructor. Display the output of "`show bgp ipv6`", and "`show bgp`

`ipv6 x.x.x.x`" for the networks with increased AS path length. Show the output of a trace according to instructions.

- How has AS Path prepend changed the IPv6 BGP table and the Routing decision. Can the decision be overridden using any other BGP configuration within an AS? Answer: review the BGP route selection rules from the slide in the presentation section.

Summary

This module has demonstrated several ways of influencing inbound and outbound routing policy.

Q: What is the difference in the resulting effects using MEDs and AS-PATH prepends?

A: AS PATH prepend affects routing announcements between two ASes, and is visible everywhere, even outside the two ASes which are making use of the prepend information. MEDs only apply between multiple peerings between the same AS. If the peer AS is announcing the local AS onwards, the metric set is that of the peer AS, not the local AS.

Consult the BGP documentation for more information. There are many possible variations on the examples given in this module. Remember the following points:

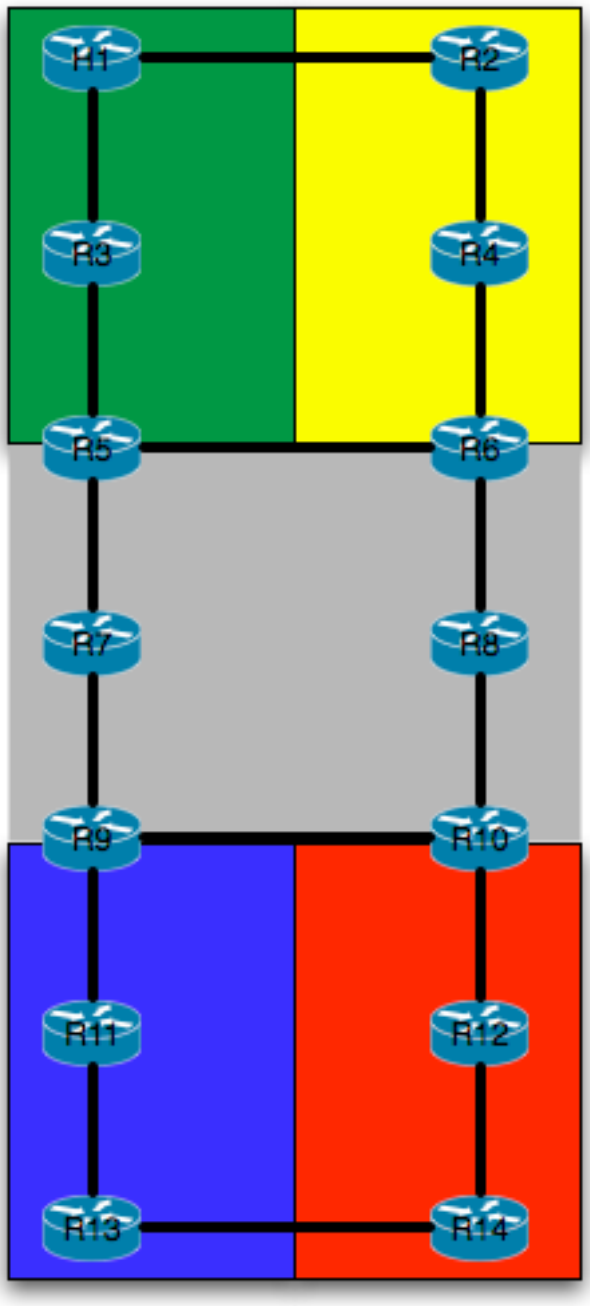
- local preference is used for influencing policy within an AS
- MEDs are used to influence policy over multiple links between the local and an immediately neighbouring AS
- The AS path prepend is used to influence external policy on a global scale (which includes the immediately neighbouring AS). Current Internet practice is to add three of the local AS in any AS prepend – obviously the application determines what is actually deployed.

Review Questions

1. Which is the most effective way of influencing how traffic leaves your network?

2. How useful do you think MEDs are in the real live Internet? Consider the answer to question one before replying!

5.5. IPv6 OSPF areas



This chapter provides an introduction to areas in OSPFv3. The module forms the basis for the Route Reflector Module which follows. It is strongly recommended that the IPv6 OSPF presentation is reviewed prior to starting this module. Prerequisites for this Module are Module 6, and the OSPF presentation. As

before, ask the workshop instructors or refer to the CD Documentation if there is any doubt.

As with the main part of Module 6 preceeding this supplement, the network is split into a core area 0 and several stub areas. The IPv6 OSPF topology will be overlaid on the existing IPv4 OSPF topology.

Functional assignments of routers in the figure above.

- Routers 5 to 10 represent the “core network” and the core interfaces are all in OSPF area 0. In a typical ISP backbone, these routers would carry all the internal link routes known in the ISPs network. Routers 5, 6, 9 and 10 are Area Border Routers, whereas Routers 7 and 8 are Internal Routers.
- Routers 1 and 3 are completely in OSPF area 10. This area is a stub area.Router 5 is the boundary between area 0 and area 10, so requires configuration for both areas.
- Routers 2 and 4 are completely in OSPF area 20. This area is also a stub area. Router 6 is the boundary between area 0 and area 20, so requires configuration for both areas.
- Routers 11 and 13 are completely in OSPF area 30. This area is also a stub area. Router 9 is the boundary between area 0 and area 30, so requires configuration for both areas.
- Routers 12 and 14 are completely in OSPF area 40. This area is also a stub area. Router 10 is the boundary between area 0 and area 40, so requires configuration for both areas.

OSPF Configuration for routers in Area 0 only

Configure OSPFv3 for your router using PID 41. Remember all the tips and techniques which you have learned in previous modules. Recall the use of the network statement, and the passive-interface default command. You only want to configure OSPF with other routers in Area 0. Do not configure OSPF with routers in areas 10 to 40 yet.

Example for Router 5:

```
ipv6 router ospf 41
 passive-interface default
 no passive-interface serial 0/0
 no passive-interface ethernet 0/0
!
interface serial 0/0
```

```
 ipv6 ospf 41 area 0
interface ethernet 0/0
 ipv6 ospf 41 area 0
interface loopback 0
 ipv6 ospf 41 area 0
!
```

OSPF configuration for routers with interfaces in Areas 10 to 40 only

Configure OSPF for your router using PID 41. Remember the tips and techniques which you have learned in previous modules. Recall the use of the network statement, and the passive-interface statement. You only want to configure OSPF with other routers which have interfaces in your own area. Do not configure OSPF with routers in area 0 yet.

Example for Router 11:

```
ipv6 router ospf 41
 area 30 stub     !we are a stub area
 passive-interface default
 no passive-interface serial 0/0
 no passive-interface serial 0/1
!
interface serial 0/0
 ipv6 ospf 41 area 30
interface serial 0/1
 ipv6 ospf 41 area 30
interface loopback 0
 ipv6 ospf 41 area 30
!
```

Checkpoint #2: Call the lab instructors and show the function of your router. You should have neighbour relationships with all the routers in your area (called intra-area). You should also demonstrate the output from “show ip route” so that you can see which routes you are hearing from which routers.

Putting it all together!

We have now configured OSPF in the routers which are wholly in a particular area. We now need to join the areas together to rebuild our OSPF network. The next step should be carried out by the appropriate router teams to achieve this.

Connecting Area 0 to the Areas 10, 20, 30 and 40

The teams running Routers 5, 6, 9 and 10, the Area Border Routers in the core of our lab network, now need to set up configuration so that the four other physically connected areas have an OSPF relationship with Area 0.

For example, the team operating Router 5 need to configure an OSPF network statement for the interface which connects to Router 3 in Area 10 so that the routers can try to establish a neighbour relationship. An example configuration might be:

```
ipv6 router ospf 41
 area 10 stub
 no passive-interface serial 0/1
!
interface serial 0/1
 ipv6 ospf 41 area 10
```

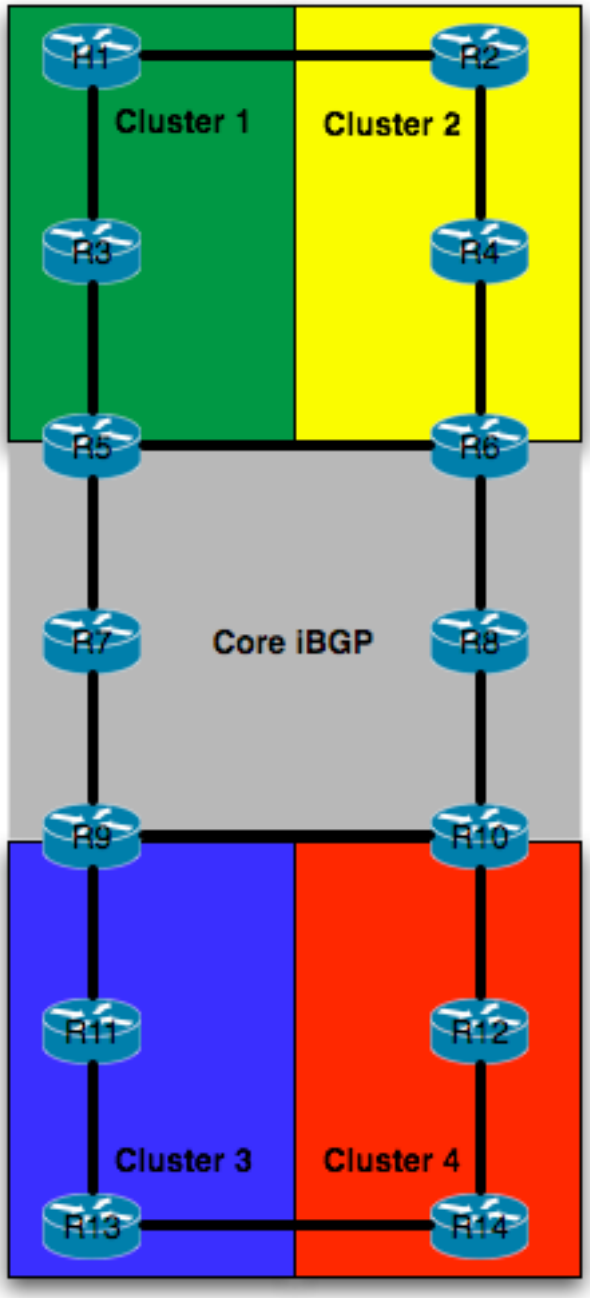
This tells the router that the OSPF neighbour it finds will be in OSPF area 10, which is a stub area.

Routers 6, 9, and 10 should enter similar and appropriate configuration for their non-area 0 neighbours.

All router teams should check their routing table

The routing table should look the same as it did in Module 1. All the prefixes in Area 0 will be available in Areas 10 through to 40. Checkpoint #3: Call the lab instructors and show the function of your router. The inter-area OSPF "peerings" will now be up. If you have an area border router, demonstrate the neighbour relationships using "`sh ipv6 ospf neigh`". If you are not in an area border, you should now have a more complete routing table.

5.6. IPv6 BGP Route-reflector lab



The main portion of Module 7 set up the iBGP for the entire network as a Route Reflector system. This supplement now does likewise for the IPv6 routing information.

Routers 5 to 10 represent the “core network” running a fully meshed iBGP, with Routers 5, 6, 9 and 10 being route reflectors. In a typical ISP backbone, these routers would carry all the routes known in the ISPs network, and possibly all the Internet routes too. Routers 1, 3 and 5 represent a cluster – say the “distribution network”, carrying out the function of aggregating customer connections. Routers 2, 4 and 6 form another cluster. Routers 9, 11 and 13 form another cluster. And Routers 10, 12 and 14 form a cluster.

Configure full mesh iBGP in the core network.

The core router should be a route reflector for the distribution router in the same cluster. Core routers should have a fully meshed iBGP among themselves. The routers in a cluster do not require fully meshed iBGP among themselves – they should only peer with the route reflector.

Example for Router 5 using peer groups (recommended):

```
router bgp 10
address-family ipv6
 bgp log-neighbor-changes
 no synchronization
 network 1001:21::/32
 neighbor V6-CORE-IBGP peer-group
 neighbor V6-CORE-IBGP description iBGP for the CORE routers
 neighbor V6-CORE-IBGP remote-as 10
 neighbor V6-CORE-IBGP password cisco
 neighbor V6-CORE-IBGP update-source loopback0
 neighbor 1001:22:ffff:ffff::1 peer-group V6-CORE-IBGP ! ibgp with Router 6
 neighbor 1001:23:ffff:ffff::1 peer-group V6-CORE-IBGP ! ibgp with Router 7
 neighbor 1001:30:ffff:ffff::1 peer-group V6-CORE-IBGP ! ibgp with Router 8
 neighbor 1001:31:ffff:ffff::1 peer-group V6-CORE-IBGP ! ibgp with Router 9
 neighbor 1001:33:ffff:ffff::1 peer-group V6-CORE-IBGP ! ibgp with Router
10
 . . .
```

Configuring route-reflector-client peers

On the routers that will be router-reflectors (Routers 5, 6, 9, and 10), configure peers inside the cluster as route-reflector-clients. Each router team shouldmannounce the /32 network (or /31 in the case of Router 3 and Router 10) which has been assigned to them.

Example for Router 5 using peer groups (recommended):

```
router bgp 10
address-family ipv6
 no synchronization
 bgp log-neighbor-changes
 network 1001:21::/32
 neighbor V6-RR-CLIENT peer-group
 neighbor V6-RR-CLIENT remote-as 10
neighbor V6-RR-CLIENT description iBGP route reflector client configuration
 neighbor V6-RR-CLIENT password cisco
 neighbor V6-RR-CLIENT update-source loopback0
```

```
 neighbor V6-RR-CLIENT route-reflector-client
 neighbor 1001:10:ffff:ffff::1 peer-group V6-RR-CLIENT  ! ibgp with Router
1
 neighbor 1001:13:ffff:ffff::1 peer-group V6-RR-CLIENT  ! ibgp with Router
3
 . . .
```

Note that the clients don't require an iBGP peering between each other – the route reflector announces one client's network to all other clients.

Route-reflector clients should configure iBGP peering to the router reflector inside the cluster.

Example for Router 11 using peer groups (recommended):

```
router bgp 10
address-family ipv6
 no synchronization
 bgp log-neighbor-changes
 network 1001:40::/32
 neighbor V6-RR peer-group
 neighbor V6-RR remote-as 10
 neighbor V6-RR description iBGP configuration for RR clients
 neighbor V6-RR password cisco
 neighbor V6-RR update-source loopback0
 neighbor 1001:31:ffff:ffff::1 peer-group V6-RR
. . .
```

Use "`show bgp ipv6 <address>`" to see how reflected prefixes show up on the clients

How do you explain the path choices which you see?

Checkpoint #2: Call the lab instructors and show the function of your router. You should have peerings with all the routers in your cluster, and any peers/clients. You should also demonstrate the output from "`show ip route`" so that you can see which routes you are hearing from which routers.

6. Host and OS based exercises

Login for FreeBSD is `root` with the password SecrecT. On all FreeBSD computers there are two editors installed `vi`, and `pico`. If you have not used `vi` before, `pico` is recommended.

6.1. Exercise 1 - Windows XP and Teredo

To administer Windows XP from the command shell instead of from the control panel, click “Start”, select run, and then type “cmd”. This will launch the command shell in a new Window. In order to activate IPv6 under Windows XP, then type as below in the command sell

```
netsh
interface
ipv6
```

You have now enabled IPv6 and is ready to configure Teredo for IPV6 transport. To this, you need to select the type of client you have. The command to activate Teredo is

```
set teredo type=client servername=default
```

You can now follow the process as the Teredo client probes the type of NAT you are running behind. You can follow this process with the command

```
show teredo
```

You will then see a number of lines that describes the status of Teredo. The interesting ones are

State
.
.
.
NAT:

While the probing is in process, you will see how Teredo in turns try Cone NAT and then restricted NAT.

If the Teredo session is successful you will see that you have an Teredo interface. You can verify this with the command

```
show interface
```

or with the command

```
ipconfig /all
```

in the command shell.

Do note however that today Teredo is not a reliable service, and this is not guaranteed to give you IPv6 connectivity. Today only implementations of the Teredo server exists, and no Teredo relay. The Teredo client in Windows XP will by default try a Teredo server called teredo.ipv6.microsoft.com. An alternative server that might work is teredo.ipv6.6wind.com. You can change the Teredo server in use with the command

```
set teredo type=client servername=<IP adress eller namn>
```

6.2. Excersie 2 – Activating IPv6 on FreeBSD

In this exercise we will activate IPv6 on our FreeBSD system. To do this we must edit the file /etc/rc.conf to add the following statements

```
ipv6_enable="YES"
ipv6_ifconfig_fxp0="<ipv6 adress>"
```

This will activate IPv6, and confingure an static IPv6 address on the Ethernet interface fxp0. In order for the changes to take effect you must run

```
group1# sh /etc/rc.d/network_ipv6 restart
```

Verify that you are successful with the command

```
group1# ifconfig fxp0
```

If you are successful you should see something like

```
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::202:b3ff:fe1a:7365%fxp0 prefixlen 64 scopeid 0x1
        inet 192.71.80.79 netmask 0xfffffe0 broadcast 192.71.80.95
        inet6 2001:698:9:401:202:b3ff:fe1a:7365 prefixlen 64
        ether 00:02:b3:1a:73:65
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
```

Each group should use addresses from 2001:670:87:50::/64

Group	Suffix
1	1
2	2
3	3
4	4
5	5
6	6

Verify that you can reach each other with the `ping6` command.

6.3. Exercise 3 – Configure FreeBSD to advertise an prefix (stateless address autoconfiguration)

In this exercise we want our FreeBSD server to act as an router for the other FreeBSD machines and through router advertisement advertise prefix and properties for the prefix to the other machines.

You do that by configuring FreeBSD's rtadvd(8) deamon. This deamon makes FreeBSD act just as a router running autoconf would. In order to have this process start automatically at reboot we add a statement to `/etc/rc.conf` as follows

```
rtadvd_enable="YES"
```

we also specify which interface it should advertise the prefix on with

```
rtadvd_interfaces="fxp0"
```

You can specify properties for the prefix you want to advertise by editing the rtdadvd configuration file `/etc/rtadvd.conf`. For example

```
fxp0:\
:addrs#1:addr="2001:471:1f11:246::":prefixlen#64:tc=ether:
```

The statement above says that on interface fxp0 I want to announce 2001:471:1f11:246::/64. That said, the file above is not needed to run rtadvd. If it doesn't rtadvd will read the hosts routing table and for each prefix configured for an interface, rtadvd will advertise that prefix as on-link. As you are all on the

same subnet, we want you to advertise different prefixes. To get a more detailed description of rtadvd.conf read the manual page by doing

```
group1# man rtdavd
```

The groups should advertise prefixes according to

Group	Prefix
1	2001:670:87:51::/64
2	2001:670:87:52::/64
3	2001:670:87:53::/64
4	2001:670:87:54::/64
5	2001:670:87:55::/64
6	2001:670:87:56::/64

You start the process by entering

```
kurspc# /usr/sbin/rtadvd
```

Can you see the configured prefixes also on your Windows XP computers?

6.4. Excersise 4 – Configuring FreeBSD as a client for stateless address autoconfiguration

In this exercise we will configure FreeBSD to listen to the prefixes advertised by the others through stateless address autoconfiguration as configured in Excersie 3.

You do this by editing `/etc/rc.conf` and adding

```
ipv6_enable="YES"
```

Verify that you are receiving the advertised prefixes from the other machines with the `ifconfig` command.

6.5. Excersice 5 – Addresses / EUI64 encoded address

Use the command "`ifconfig fxp0`" to look at the addresses on the interface. You should now see the MAC-address, the link-local address and the addresses from the stateless address autoconfiguration. Can you determine the EUI64 encoding? Determine the u and g bits.

6.6. Excersie 6 – Configuration of tunnelled IPv6 on FreeBSD

In this excersie we will create tunnels between all the FreeBSD machines. This is done by configuring tunnel interfaces, gif interfaces, as follows

```
# ifconfig gif0 create
# ifconfig gif0
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
# ifconfig gif0 tunnel MY_IPv4_ADDR   HIS_IPv4_ADDR
# ifconfig gif0 inet6 aliasMY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR
```

The IPv4 addresses for the other groups you can find above. Once you have those and have configured tunnels to all the other groups, you need to configure IPv6. Configure the tunnels with static IPv6 addresses as follows

From Group	Src Addr	To Group	Dst Addr
1	2001:670:87:57::1/64	2	2001:670:87:57::2/64
1	2001:670:87:58::1/64	3	2001:670:87:58::2/64
1	2001:670:87:59::1/64	4	2001:670:87:59::2/64
1	2001:670:87:5A::1/64	5	2001:670:87:5A::2/64
1	2001:670:87:5B::1/64	6	2001:670:87:5B::2/64
2	2001:670:87:5C::1/64	3	2001:670:87:5C::2/64
2	2001:670:87:5D::1/64	4	2001:670:87:5D::2/64
2	2001:670:87:5E::1/64	5	2001:670:87:5E::2/64
2	2001:670:87:5F::1/64	6	2001:670:87:5F::2/64
3	2001:670:87:60::1/64	4	2001:670:87:60::2/64
3	2001:670:87:61::1/64	5	2001:670:87:61::2/64
3	2001:670:87:62::1/64	6	2001:670:87:62::2/64
4	2001:670:87:63::1/64	5	2001:670:87:63::2/64
4	2001:670:87:63::1/64	6	2001:670:87:63::2/64
5	2001:670:87:64::1/64	6	2001:670:87:63::2/64

Verify that you have working IPV6 over the tunnels with the ping6 command. What would you need to do, to do this with 6to4?

6.7. Excerise 7 – Identifying packets and headers

With the command

```
group1# tcpdump -i fxp0 -vvv ip6
```

you can view the IPv6 packets on the local network. Try to find the various IPv6 header fields and any extension headers on some of the packets. For example you can

```
kurspc# tcpdump -i fxp0 -vvv ip6 protochain 6
```

to view only TCP packets. Try to look at this while you are running some IPv6 TCP traffic.

Can you find some of the Teredo packets with tcpdump?

6.8. Exercise 8 – IPv6 based services

On each of the FreeBSD machines there is an account called "group1" through "group5". Configure some services on your FreeBSD machine to use IPv6. Do this for example by editing `/etc/inetd.conf.` For example you can enable telnet ftp etc. Ask the other groups to log in as "group<nr>" on one of the IPv6 addresses.

You can also try SSH over IPv6 with

```
group1# ssh -6 <ipv6 adress>
```

Try even without the -6 flag. Does ssh use IPv4 or IPv6? How do you know?

6.9. Exercise 9 - Duplicate Address Detection

Try to configure one of the other groups static addresses from exercise 1 on your interface. Follow what happens with `tcpdump`. Can you see AD being performed? Note that not all groups can change address at the same time… ☺

If you still have time after this you can also do

7. Extra exercises

In this part you will learn how to install and configure common services to use IPv6.

7.1. Extra Exercise 10 – route6d

`route6d` is an RIPng implementation for FreeBSD. RIP is generally something that you should avoid running at all cost. We are here using it only to illustrate the routing protocol. We will now use route6d to distribute our networks over the tunnels to the other groups. You launch the deamon with

```
group1# /usr/sbin/route6d
```

If you want to have more debugging output to follow what happens, you can do that by launching the deamon with

```
kurspc# /usr/sbin/route6d –d
```

The route6d process will then run in the foreground printing debugging output directly on the screen. You should now see all the other groups tunnel prefixes as well. You can view the local routing table with

```
kurspc# netstat -rn
```

7.2. Extra Exercise 11 – Configuring DNS/bind

Now you have to fetch and install bind9 on your FreeBSD machine. You can do this by using the ports feature as follows

```
group1# cd/usr/ports/dns/bind9/
group1# make install
```

Once this is done create a `/etc/named.conf` file with a zone statement for your own zone (for example group1.net). Add a forward DNS statement to the zone for your IPv6 address that you had in exercise 1. Also create a reverse zone file for the same IP address. Example configurations.

/etc/named.conf:

```
options {
//      directory "/etc/namedb";

        // Directory for named
        directory "/var/named";

        listen-on { any; };
        listen-on-v6 { any; };

}

zone "kurtis.pp.se" in {
        type master;
        file "master/db.kurtis.pp.se";
};
```

/var/named/master/db.kurtis.pp.se :

```
$ORIGIN .
$TTL 86400      ; 1 day
kurtis.se.              IN SOA  mariehamn.kurtis.pp.se.
hostmaster.kurtis.pp.se. (
                                2005092801 ; serial
                                28800      ; refresh (8 hours)
                                7200       ; retry (2 hours)
                                604800     ; expire (1 week)
                                86400      ; minimum (1 day)
                                )
                        NS      vovve.besserwisser.org.
                        NS      casper.besserwisser.org.
                        NS      mariehamn.kurtis.pp.se.

                        MX      10 lemland.kurtis.pp.se.
                        MX      50 mariehamn.kurtis.pp.se.
                        MX      100 casper.besserwisser.org.


www                     A       194.15.141.69
                        AAAA    2001:670:87:1:226:54ff:fe08:9e4c
```

Verify that the DNS-server answers IPv6 queries by asking for the name in your zone

```
group1# dig @<ipv6 adress> <namn>
```

7.3. Extra Exercise 12 – Configuring a web-server

Now install the web-server Apache v2 by entering

```
group1# cd /usr/ports/www/apache2
group1# make install
```

Configure it by editing the file

```
/usr/local/etc/apache2/httpd.conf
```

In that file you need to tell Apache that it should listen on your IPv6 address by adding the following statement to the configuration file

```
Listen [<er IPv6 adress>]:80
```

Then start the web-servers `httpd` process by

```
group1# /usr/local/sbin/apachectl start
```

Verify that the process answers on IPv6 by

```
group1# telnet -6 <er ipv6 adress>
```

7.4. Extra Exercise 13 – Mail server

Now we will install the postfix mail-server package to accept and deliver mail over IPv6. First we need to install it, which we do with the following commands

```
group1# cd /usr/ports/mail/postfix
group1# make install
```

Postfix will after the installation automatically listen on the IPv6 address. You can verity this as follows. First make sure that the sendmail process is not running with

```
group1# ps auxw | grep sendmail
```

```
group1#
```

That should show nothing but a new prompt. Then launch the `postfix` process with

```
kurspc# /usr/local/sbin/postfix start
```

Verify that postfix is indeed running IPv6 with

```
kurspc# telnet -6 <er ipv6 adress>
```