Module 1 – IPv6 OSPF and iBGP

Objective: Create a basic physical lab interconnection using IPv6 with one OSPF Area and one BGP AS number (optionally running on top of an existing IPv4 infrastructure).

Prerequisites: Knowledge of Cisco router CLI, previous hands on experience.

The following will be the common topology used for this supplement.

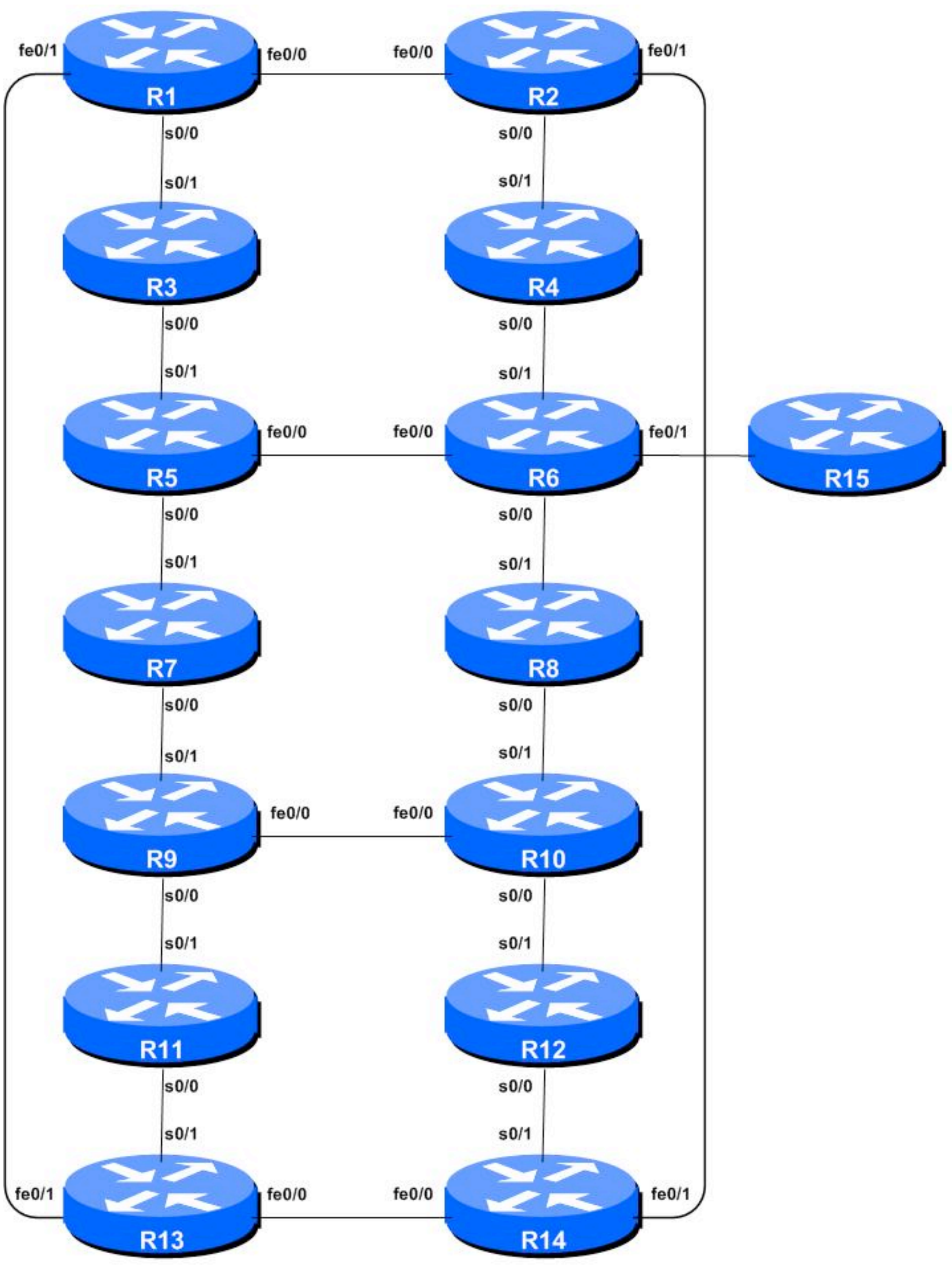

Figure 1 – ISP Lab Basic Configuration

Lab Notes

This Module is intended to supplement Module 1 of the ISP Workshop once that has been completed. The topology and IPv4 configuration should be left exactly as it was at the end of Module 1.

If this lab is attempted without completing the IPv4 version, then the OSPF and BGP router-ids need to be set before OSPFv3 and IPv6 BGP will work (ask the instructors how to do this). However this is not a recommended scenario as almost any service provider will plan to deploy IPv6 in parallel with an existing IPv4 infrastructure (this is called dual stack).

The routers used for this portion of the workshop must support IPv6. This is basically any IP Plus image from 12.2T onwards (IP Plus was renamed to Advanced IP Services for most platforms as from 12.3 mainline). As always, it is best to check the Cisco Feature Navigator www.cisco.com/go/fn to be absolutely sure which images set and platform supports IPv6. Unfortunately IPv6 is not part of the basic IP only or Service Provider IOS images used by most ISPs.

Note: these labs assume that the routers used are using a minimum of IOS 12.4 mainline. Syntax predating IOS 12.4 is discussed in the optional sections throughout the workshop.

Lab Exercise

1. **Enable IPv6.** Cisco routers with an IOS supporting IPv6 currently do not ship with IPv6 enabled by default. This needs to be done before any of the following exercises can be completed. To do this, use the following command:

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

   The router is now configured to support IPv6 Unicast (as well as IPv4 Unicast which is the default). Save the configuration.

2. **Enable IPv6 CEF.** Unlike IPv4, CEFv6 is not enabled by default. So we now need to enable IPv6 CEF also, using the following command:

   ```
   Router(config)# ipv6 cef
   ```

   Nothing will break if IPv6 CEF is not enabled, but more advanced features such as NetFlow will not function without IPv6 CEF being enabled.

3. **Disable IPv6 Source Routing.** Unless you really believe there is a need for it, source routing should be disabled. This option, enabled by default, allows the router to process packets with source routing header options. This feature is a well-known security risk as it allows remote sites to send packets with different source address through the network (this was useful for troubleshooting networks from different locations on the Internet, but in recent years has been widely abused for miscreant activities on the Internet).

   ```
   Router1 (config)# no ipv6 source-route
   ```

4. **IPv6 Addressing Plans.** Addressing plans in IPv6 are somewhat different from 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 at least 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.

5. **IPv6 Addresses.** As with the IPv4 portion of this Module we are going to assume that each router team 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 router team receives a /32 worth of address space. The teams 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**	**R8**	**1001:30::/32**
R2	**1001:11::/32**	**R9**	**1001:31::/32**
R3	**1001:12::/31**	**R10**	**1001:32::/31**
R4	**1001:20::/32**	**R11**	**1001:40::/32**
R5	**1001:21::/32**	**R12**	**1001:41::/32**
R6	**1001:22::/32**	**R13**	**1001:42::/32**
R7	**1001:23::/32**	**R14**	**1001:43::/32**

   Make a separate note separate from this work book of the address block assigned to your router – reference will be made to it several times throughout the rest of this workshop. Unless, of course, you like searching back through this booklet to find this page.

6. **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 Router Team 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 first /48 out of its /32 address block for its infrastructure. For example, Router 5 will use 1001:21:0::/48 for numbering its "infrastructure". (Note that "0" has been inserted there intentionally – IPv6 convention is that redundant zeros are removed, so the address would really be written as 1001:21::/48.)

See the addressing plan in the Appendices for some recommendations – using the recommended addressing plan will make troubleshooting easier, and help the lab assistants with any troubleshooting issues.

Note: this lab will **not** use EUI-64 interface addressing, but instead will assign absolute addressing to each interface. The latter is much 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 1001:11:1:1::1/64
```

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 RFC4291 which specifies the IPv6 addressing architecture. So all point to point links use a /64, all LANs use a /64, etc.

7. **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.

8. **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

9. **Assign IPv6 Addresses to Loopback Interfaces.** While there is no need for a Loopback interface in this lab yet, it is still useful to configure an IPv6 address for it at this time. The loopback 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 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 the lack of any IPv4 address on the router would problem? Ask the lab instructors to discuss.

   As the minimum subnet size possible for IPv6 is a /64, we will assign the first /64 out of our /48 infrastructure block to be used for loopbacks (again note the deliberate insertion of 0s). 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	**1001:10:0:0::/64**	**R4**	**1001:20:0:0::/64**
R2	**1001:11:0:0::/64**	**R5**	**1001:21:0:0::/64**
R3	**1001:13:0:0::/64**	**R6**	**1001:22:0:0::/64**

Cisco Systems Inc
170 West Tasman Drive.
San Jose, CA 95134-1706
Phone: +1 408 526-4000
Fax: +1 408 536-4100

R7	1001:23:0:0::/64	R11	1001:40:0:0::/64
R8	1001:30:0:0::/64	R12	1001:41:0:0::/64
R9	1001:31:0:0::/64	R13	1001:42:0:0::/64
R10	1001:33:0:0::/64	R14	1001:43:0:0::/64

From these blocks, each Router Team 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::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::1/128
```

The example in Figure 3 might show how the loopback/infrastructure block fits into an ISP network-addressing scheme.

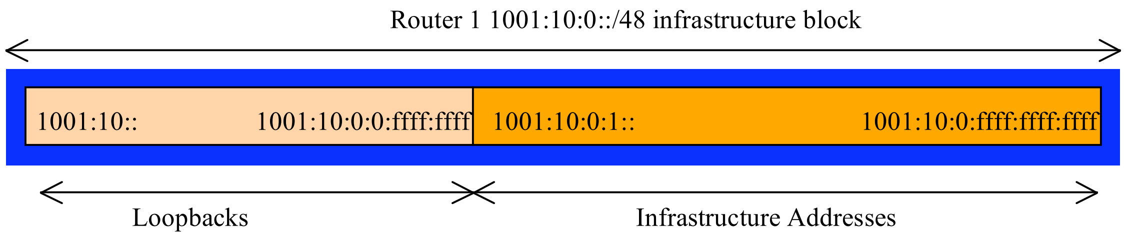

Figure 2 – Loopback Addressing Scheme: Picking the first /64 in the allocated block of /48

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!

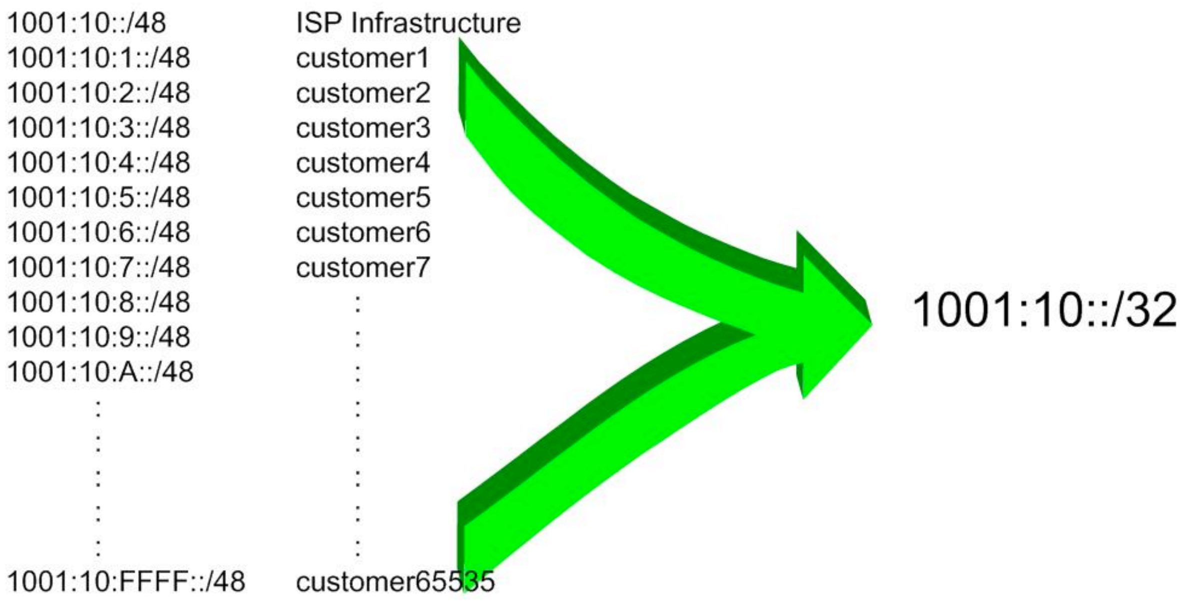

Figure 3 – Extract from ISP addressing plan

10. OSPF within the same AS. Each router Team should enable OSPF for IPv6 on their router. As with the IPv4 lab, 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
```

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...
```

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.

11. **OSPF Router ID.** If this module is being used with out any IPv4 configuration being present on the router, then each router team will need to set the router-id for the OSPF process. In IOS, the router-id is taken from the IPv4 address of the loopback interface (if configured) otherwise the highest IPv4 address on the router. If there is no IPv4 address configured, the router will not be able to set the OSPFv3 router-id, so the OSPFv3 process will not function.

    If there is no IPv4 configured on the router, now set the OSPFv3 router-id. For this exercise, the value chosen should be 192.168.0.<router-number>.

```
Router1(config)#ipv6 router ospf 41
Router1(config-rtr)#router-id 192.168.0.1
...etc...
```

12. **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 detail
```

13. **(Optional if DNS Server available). Enable OSPF name lookups on the routers.** 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.

14. **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.*

15. **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 new IPv6 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
```

16. **Configuring iBGP neighbours (Part 2).** Now we can configure the IPv6 iBGP neighbours, as in the example for Router 4 below. 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::1 remote-as 10
Router4(config-router-af)#neighbor 1001:10::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:10::1 description iBGP with Router1
Router4(config-router-af)#neighbor 1001:10::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:11::1 remote-as 10
Router4(config-router-af)#neighbor 1001:11::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:11::1 description iBGP with Router2
Router4(config-router-af)#neighbor 1001:11::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:13::1 remote-as 10
Router4(config-router-af)#neighbor 1001:13::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:13::1 description iBGP with Router3
Router4(config-router-af)#neighbor 1001:13::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:21::1 remote-as 10
Router4(config-router-af)#neighbor 1001:21::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:21::1 description iBGP with Router5
Router4(config-router-af)#neighbor 1001:21::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:22::1 remote-as 10
Router4(config-router-af)#neighbor 1001:22::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:22::1 description iBGP with Router6
Router4(config-router-af)#neighbor 1001:22::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:23::1 remote-as 10
Router4(config-router-af)#neighbor 1001:23::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:23::1 description iBGP with Router7
Router4(config-router-af)#neighbor 1001:23::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:30::1 remote-as 10
Router4(config-router-af)#neighbor 1001:30::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:30::1 description iBGP with Router8
Router4(config-router-af)#neighbor 1001:30::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:31::1 remote-as 10
Router4(config-router-af)#neighbor 1001:31::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:31::1 description iBGP with Router9
Router4(config-router-af)#neighbor 1001:31::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:33::1 remote-as 10
Router4(config-router-af)#neighbor 1001:33::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:33::1 description iBGP with Router10
Router4(config-router-af)#neighbor 1001:33::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:40::1 remote-as 10
Router4(config-router-af)#neighbor 1001:40::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:40::1 description iBGP with Router11
Router4(config-router-af)#neighbor 1001:40::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:41::1 remote-as 10
Router4(config-router-af)#neighbor 1001:41::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:41::1 description iBGP with Router12
```

```
Router4(config-router-af)#neighbor 1001:41::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:42::1 remote-as 10
Router4(config-router-af)#neighbor 1001:42::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:42::1 description iBGP with Router13
Router4(config-router-af)#neighbor 1001:42::1 activate
Router4(config-router-af)#
Router4(config-router-af)#neighbor 1001:43::1 remote-as 10
Router4(config-router-af)#neighbor 1001:43::1 update-source loopback 0
Router4(config-router-af)#neighbor 1001:43::1 description iBGP with Router14
Router4(config-router-af)#neighbor 1001:43::1 activate
```

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

Use *show bgp ipv6 unicast 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 Router Team for that neighbour connection to troubleshoot the problem.

17. **BGP Router ID.** If this module is being used with out any IPv4 configuration being present on the router, then each router team will need to set the router-id for the BGP process. In IOS, the router-id is taken from the IPv4 address of the loopback interface (if configured) otherwise the highest IPv4 address on the router. If there is no IPv4 address configured, the router will not be able to set the BGP router-id, so the BGP process will not function.

    If there is no IPv4 configured on the router, now set the BGP router-id. For this exercise, the value chosen should be 192.168.0.<router-number>.

```
Router4(config)#router bgp 10
Router4(config-rtr)#bgp router-id 192.168.0.4
...etc...
```

18. **Checking the Configuration.** Do a *show run | begin bgp* to check and see what the BGP configuration looks like. Notice how the router has separated the generic part of the BGP configuration from the specific IPv6 address family content. An example of the output follows (IPv4 configuration from the IPv4 module has been omitted):

```
Router4#sh run | b bgp
router bgp 10
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 1001:10::1 remote-as 10
 neighbor 1001:10::1 description iBGP with Router2
 neighbor 1001:10::1 update-source Loopback0
…
 address-family ipv6
 neighbor 1001:10::1 activate
 neighbor 1001:11::1 activate
 neighbor 1001:12::1 activate
…
 exit-address-family
!
```

19. **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 the router has learned
`show bgp ipv6 unicast summary`	: see a list of BGP IPv6 **peers** that the router sees
`show bgp ipv6 unicast`	: 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 unicast***? If not, why not? Are there any routes tagged "B" when you do a ***show ipv6 route***?

20. 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 unicast* 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 unicast 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 ipv6 route* to check the local forwarding table. If not, why not?

21. Add the following commands to BGP:

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

Q. Does the network appear in the router's forwarding table? Now use the command *show ipv6 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?

<u>Checkpoint #2 :</u> *call the lab assistant to verify the connectivity.*

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

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

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

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

23. **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.

Converting the BGP CLI (For Routers running IOS <12.4)

24. **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 configuration from the first instance of the word "bgp". What do you see? Write your answer here:

25. **Converting the BGP CLI.** 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 is obsolete as from 12.4 IOS) 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)#
```

26. **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::1 remote-as 1
 neighbor 1001:11::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::1 activate
 neighbor 1001:13::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.

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?

CONFIGURATION NOTES

Documentation is critical! You should record the configuration at each ***Checkpoint***, as well as the configuration at the end of the module.