IPv6 Module 1a – OSPF

Objective: Create a basic physical lab interconnection using IPv6 with one OSPF Area running on top of an existing IPv4 infrastructure.

Prerequisites: The setup section of the IPv6 Module 1.

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

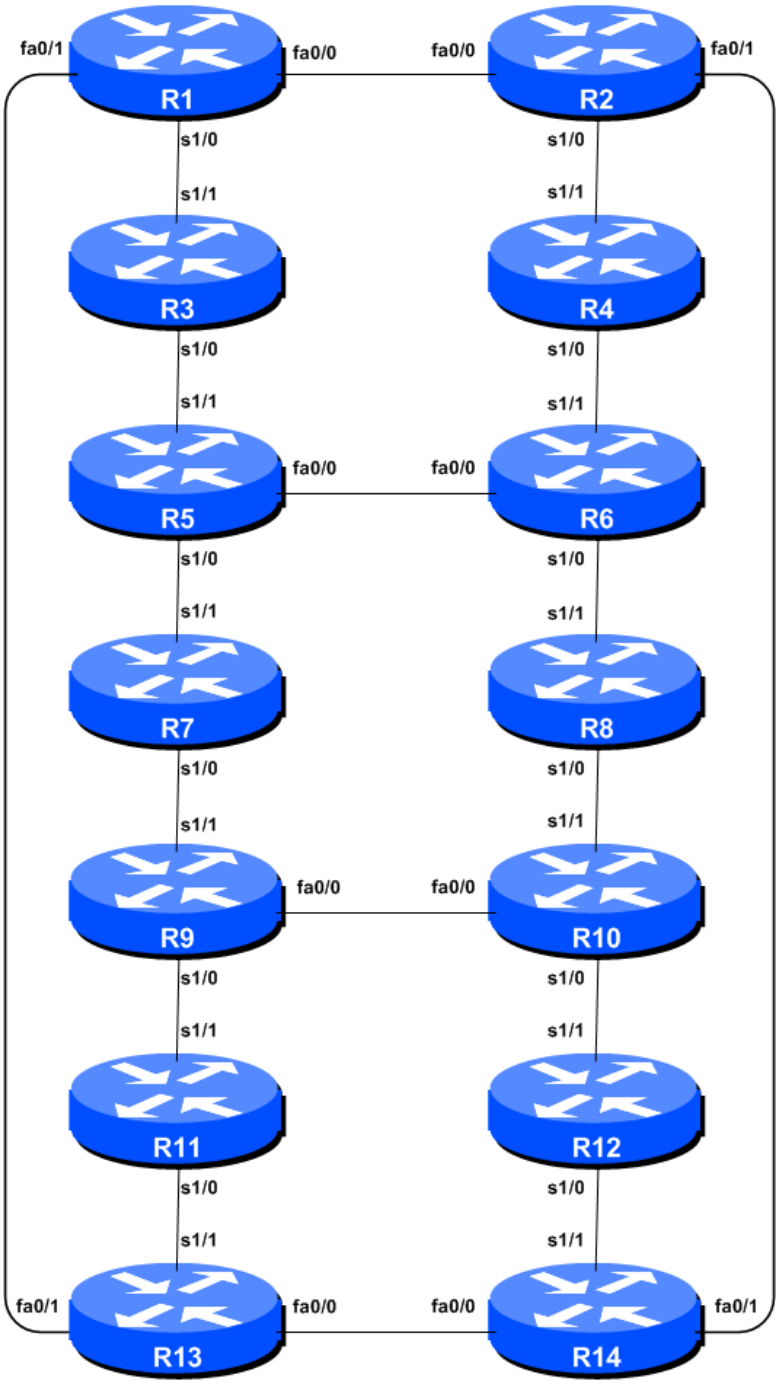

Figure 1 – ISP Lab Basic Configuration

Lab Notes

This lab continues from the previous one by adding IS-IS to the configured, addressed and confirmed working interfaces. Please refer to the Setup Module for further information and reference points about the purpose of Module 1.

Lab Exercise

1. **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 1/0
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 1/0
Router1(config-rtr)#no passive-interface fastethernet 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.

2. **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
```

3. **Avoiding Traffic Blackhole on Reboot[1].** When a router restarts after being taken out of service, OSPF will start distribute prefixes as soon as adjacencies are established with its neighbours. In the next part of the workshop lab, we will be introducing iBGP. So if a router restarts, OSPF will start up well before the iBGP mesh is re-established. This will result in the router landing in the transit path for traffic, with out the routing table being completed by BGP. There will not be complete routing information on the router, so any transit traffic (from customer to peer or upstream, or vice-versa) will be either dropped, or resulting in packets bouncing back and forth between adjacent routers. To avoid this problem, we require the router to not announce it is availability until the iBGP mesh is up and running. To do this, we have to provide the following command:

```
Router1(config)#ipv6 router ospf 41
Router1(config-router)#max-metric router-lsa on-startup wait-for-bgp
```

   This sets up OSPF such that all IPv6 routes via this router will be marked as unreachable (very high metric) until iBGP is up and running. Once iBGP is running, the prefixes distributed by OSPF will revert to standard metric values, and the router will pass transit traffic as normal.

4. **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 #2: *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.*

5. **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 2001:db8::c
```

   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 2001:db8::c

Type escape sequence to abort.
Tracing the route to 2001:db8::c
```

[1] This feature may not be available for OSPFv3 on all IOS releases.

```
  1 2001:db8:0:3::1    4 msec
    2001:db8:0:2::1    0 msec
    2001:db8:0:3::1    0 msec
  2 2001:db8:0:f::1    4 msec
    2001:db8:0:8::1    4 msec
    2001:db8:0:f::1    0 msec
  3 2001:db8:0:13::    4 msec *  4 msec
Router1>
```

6. **Other Features in OSPF.** Review the documentation or use command line help by typing *?* to see other *show* commands and other OSPF 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?