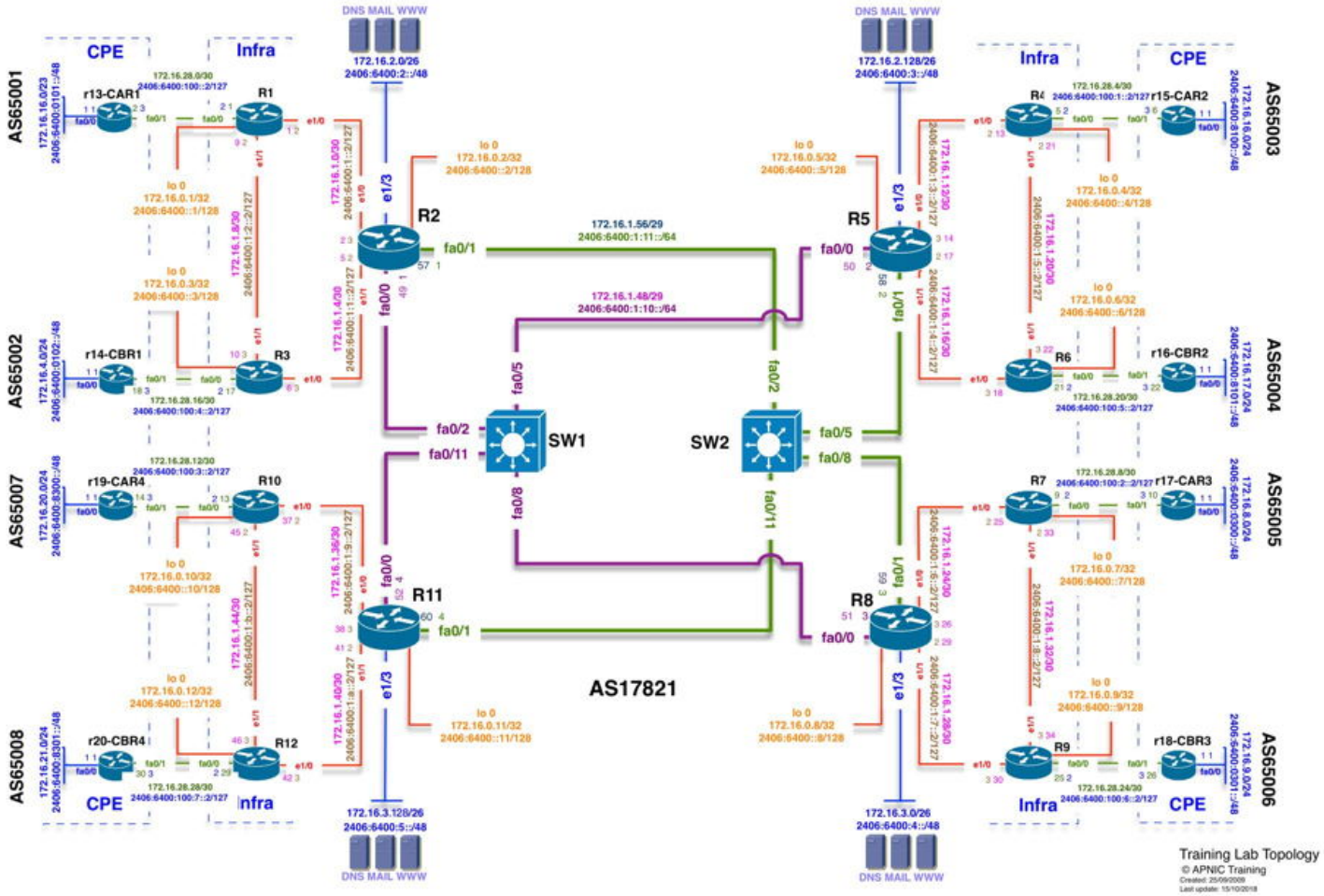
Module 3 - iBGP Configuration(full-mesh)

Topology introduction:

- The topology below shows 4 regional networks comprised of a core POP and 2 aggregation POPs (edge routers).
- Edge routers aggregate downstream customers.
- The regional networks are interconnected with redundant transport links.



Lab Tasks

So far (after completing Module-2), you should be able to reach all routers within your network (loopbacks/P2Ps).

We generally carry customers prefixes in BGP (could be your downstream customers, or retail customers – Wired Broadband, Wireless Broadband, etc).

In this lab, we will configure iBGP on top of the working OSPFv3 network.

Take note of the following for iBGP:

1. BGP relies on TCP (port 179), which means BGP routers need to first establish a TCP session/connection before any BGP session can be initiated.
2. By rule, iBGP routers are not allowed to announce routes/prefixes learned from an iBGP peer to its other iBGP peers to prevent routing loops. Which means, iBGP neighbours need to be fully meshed.
3. Since IGP carries all networks within an AS, for iBGP, neighbour relationships can be established using loopback interfaces (reachable through IGP).
4. By default, a router uses its exit interface address as the source address for all locally originated packets (including BGP update messages). Hence, it is important that the BGP TCP connection is established using the correct addresses between peers. Therefore, if you use the loopback interfaces to establish iBGP sessions, you need to force the router to use the loopback address as the source for all BGP messages using “update-source loopback” command
5. We will carry the following 8 Customer WAN prefixes (for ICMP reachability - monitoring) and 4 Datacenter prefixes in iBGP .

IPv6

Customer Side P-2-P	Datacentre
R1 => 2406:6400:100:0::/64	R2 => 2406:6400:2::/48
R3 => 2406:6400:100:4::/64	R5 => 2406:6400:3::/48
R4 => 2406:6400:100:1::/64	R8 => 2406:6400:4::/48
R6 => 2406:6400:100:5::/64	R11 => 2406:6400:5::/48
R7 => 2406:6400:100:2::/64	
R9 => 2406:6400:100:6::/64	
R10 => 2406:6400:100:3::/64	
R12 => 2406:6400:100:7::/64	

IPv4

Customer Side P-2-P	Datacentre
R1 => 172.16.28.0/30	R2 => 172.16.2.0/26
R3 => 172.16.28.16/30	R5 => 172.16.2.128/26
R4 => 172.16.28.4/30	R8 => 172.16.3.0/26
R6 => 172.16.28.20/30	R11 => 172.16.3.128/26
R7 => 172.16.28.8/30	
R9 => 172.16.28.24/30	
R10 => 172.16.28.12/30	
R12 => 172.16.28.28/30	

Lab Exercise

Step 1 - iBGP Peering Configuration:

We need to establish iBGP relationship with all routers within our AS (17821).

Example Configuration on R1:

```
config t
router bgp 17821
    !IOS assumes that all iBGP peers will be IPv4 only
no bgp default ipv4-unicast
    !Log neighbor changes (Notification)
bgp log-neighbor-changes
    !Please do the same for IPv4 as well
address-family ipv6 unicast
neighbor 2406:6400::2 remote-as 17821
neighbor 2406:6400::2 update-source loopback 0
neighbor 2406:6400::2 description iBGP with R2
neighbor 2406:6400::2 activate
neighbor 2406:6400::3 remote-as 17821
neighbor 2406:6400::3 update-source loopback 0
neighbor 2406:6400::3 description iBGP with R3
neighbor 2406:6400::3 activate
neighbor 2406:6400::4 remote-as 17821
neighbor 2406:6400::4 update-source loopback 0
neighbor 2406:6400::4 description iBGP with R4
neighbor 2406:6400::4 activate
neighbor 2406:6400::5 remote-as 17821
neighbor 2406:6400::5 update-source loopback 0
neighbor 2406:6400::5 description iBGP with R5
neighbor 2406:6400::5 activate
neighbor 2406:6400::6 remote-as 17821
neighbor 2406:6400::6 update-source loopback 0
neighbor 2406:6400::6 description iBGP with R6
neighbor 2406:6400::6 activate
neighbor 2406:6400::7 remote-as 17821
neighbor 2406:6400::7 update-source loopback 0
neighbor 2406:6400::7 description iBGP with R7
neighbor 2406:6400::7 activate
neighbor 2406:6400::8 remote-as 17821
neighbor 2406:6400::8 update-source loopback 0
neighbor 2406:6400::8 description iBGP with R8
neighbor 2406:6400::8 activate
neighbor 2406:6400::9 remote-as 17821
```

```
neighbor 2406:6400::9 update-source loopback 0
neighbor 2406:6400::9 description iBGP with R9
neighbor 2406:6400::9 activate
neighbor 2406:6400::10 remote-as 17821
neighbor 2406:6400::10 update-source loopback 0
neighbor 2406:6400::10 description iBGP with R10
neighbor 2406:6400::10 activate
neighbor 2406:6400::11 remote-as 17821
neighbor 2406:6400::11 update-source loopback 0
neighbor 2406:6400::11 description iBGP with R11
neighbor 2406:6400::11 activate
neighbor 2406:6400::12 remote-as 17821
neighbor 2406:6400::12 update-source loopback 0
neighbor 2406:6400::12 description iBGP with R12
neighbor 2406:6400::12 activate
```

Step 2 - Network Advertisement:

We will announce the customer WAN prefixes (Ex: R1 - 2406:6400:100:0::/64) and datacenter prefixes (Ex: R2 - 2406:6400:2::/48)in iBGP.

Example Config on R1:

```
config t
router bgp 17821
address-family ipv6 unicast
network 2406:6400:100:0::/64
exit
exit
```

Note: For IPv4, you need to use the mask command instead of prefix length.

The BGP network command requires a route to already exist in the routing table before advertising the route into BGP. Hence, we need a static route of the aggregate prefix pointing to the `Null0` (forces it to exist in the routing table, albeit towards the Null interface, so that BGP can announce it).

```
ipv6 route 2406:6400:0100:0::/64 null 0
```

Please remember to save the configuration.

Step 3 - Verify iBGP Configuration:

Verify IPv4 BGP configuration:

`show bgp ipv4 unicast summary` ! List IPv4 BGP peers

`show bgp ipv4 unicast` ! List IPv4 routes in BGP Table

`show bgp ipv4 unicast <prefix/length>` ! List IPv4 specific routes

`show ip route bgp` ! Check IPv4 routes learned via BGP

`show ip route` ! Check your IPv4 routing table (best paths)

Verify IPv6 BGP configuration:

`show bgp ipv6 unicast summary` ! List IPv6 BGP peers

`show bgp ipv6 unicast` ! List IPv6 routes in BGP Table

`show bgp ipv6 unicast <prefix/length>` ! List IPv6 specific routes

`show ipv6 route bgp` ! Check IPv6 routes learned via BGP

`show ipv6 route` ! Check your IPv6 routing table (best paths)

`show bgp ipv6 unicast neighbors <neighbour-address> advertised-routes` ! Check routes advertised to your neighbour

`show bgp ipv6 unicast neighbors <neighbour-address> routes` ! Check routes learned from your neighbour