Task 2.1

R1:
```
interface Serial0/0
 bandwidth 256

!
interface Serial0/1
 bandwidth 1536
!
router eigrp 100
 variance 5
```

R3:
```
interface Serial1/1.23 point-to-point
 bandwidth 1280
```

Task 2.1 Breakdown

EIGRP is the only IGP that supports unequal cost load balancing. In order to enable this load balancing issue the **`variance`** command under the EIGRP process. In order for a path to be considered for unequal cost load balancing it must be a feasible successor with a metric less than or equal to the successor's metric times the variance.

To choose the best path through the network and prevent looping EIGRP's route selection uses the *feasibility condition*. In order to understand this calculation it is important to understand the difference between advertised distance and local distance. Advertised distance is the metric reported by the upstream neighbor as their cost to the destination. Local distance is the metric from the local device to the upstream neighbor.

First the local router looks through all advertised paths and chooses the path with the lowest advertised distance plus local distance. Like other protocols this is simply the lowest end to end metric for the path. The metric for this path is called the *feasible distance*. The path itself called the *successor*. The successor is the best route to the destination.

Once the successor has been found EIGRP does an additional check to see if there may be alternate paths throughout the network. These alternate paths are known as *feasible successors*. These are paths that could be (are feasible to be) the successor if the successor is lost. A path whose advertised distance is lower than the feasible distance of the successor is deemed a feasible successor. In the case that a router is advertising a lower distance than the local device is using as its successor it can be guaranteed that there is not a loop in the topology.

Note

Only routes that are feasible successors can be used for unequal cost load balancing.

Now that the successor and all feasible successors have been chosen the router does a final check based on the input variance value to determine which feasible successors can be installed in the IP routing table along with the successor. If the end to end metric of a feasible successor is less than or equal to the metric of the successor times the variance it is valid to be installed as an additional path.

EIGRP unequal cost load balancing also does efficient traffic sharing. For example if the successor has a metric of one and the feasible successor has a metric of two, two packets will be sent out the successor's path and one packet will be sent out the feasible successor's path. This ensures that higher bandwidth paths are more utilized than lower bandwidth paths.

In the above task, R1 is to be configured to send traffic out to the destination 164.X.26.0/24 to both R3 and R2 in a ratio of 5:1 respectively. In addition to this the question specifies what the underlying bandwidths of the network circuits are. The first step in accomplishing this goal is to set the appropriate **`bandwidth`** statement on the interface. In the above configuration this is done on the outgoing interfaces to reach the destination. Typically the bandwidth value is configured on both ends of the link to be the same value, but in this case it is not required to accomplish the goal.

After the **`bandwidth`** values are set the following output is seen on R1:

```
Rack1R1#show ip eigrp topology 164.1.26.0 255.255.255.0
IP-EIGRP (AS 100): Topology entry for 164.1.26.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is
3026432
  Routing Descriptor Blocks:
  164.1.13.3 (Serial0/1), from 164.1.13.3, Send flag is 0x0
      Composite metric is (3026432/2514432), Route is Internal
      Vector metric:
        Minimum bandwidth is 1280 Kbit
        Total delay is 40100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
  164.1.12.2 (Serial0/0), from 164.1.12.2, Send flag is 0x0
      Composite metric is (10514432/28160), Route is Internal
      Vector metric:
        Minimum bandwidth is 256 Kbit
        Total delay is 20100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
```

From this output we can see that R1 has two paths, one through R3 and one through R2. The path through R3 has a metric of 3026432, while the path through R2 has a metric of 10514432. Since the metric through R3 is less it is the successor. Next the feasibility check is run, and R2's advertised distance of 28160 is compared against the feasible distance of 3026432. Since R2's advertised distance is less than the feasible distance the route through R2 is a feasible successor.

At this point if the **`variance`** command was configured traffic would be load balanced between R3 and R2 in a ratio of 10514432:3026432, or approximately 80:23. This can be seen in the **`show ip route 164.1.26.0`** output on R1:

```
Rack1R1#show ip route 164.1.26.0
Routing entry for 164.1.26.0/24
  Known via "eigrp 100", distance 90, metric 3026432, type internal
  Redistributing via eigrp 101
  Last update from 164.1.13.3 on Serial0/1, 00:04:00 ago
  Routing Descriptor Blocks:
  * 164.1.12.2, from 164.1.12.2, 00:04:00 ago, via Serial0/0
      Route metric is 10514432, traffic share count is 23
      Total delay is 20100 microseconds, minimum bandwidth is 256 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1
    164.1.13.3, from 164.1.13.3, 00:04:00 ago, via Serial0/1
      Route metric is 3026432, traffic share count is 80
      Total delay is 40100 microseconds, minimum bandwidth is 1280 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

In order to achieve the desired ratio of 5:1 we must now modify the metric through R2 to be 5 times that of R3's metric, while still keeping the route through R2 a feasible successor. The easiest way to do this is to change the **delay** on R1's connection to R2 over the Frame Relay cloud. To determine the correct delay value we must first determine how the current composite metric value is derived. EIGRP metric calculation uses the formula:

Metric = [k1 * bandwidth + (k2 * bandwidth)/(256 - load) + k3 * delay] * [k5/(reliability + k4)]

The "k" values are derived from the **`metric weights`** command, where K1 and K3 are 1 by default and all other values are 0. This essentially means that only bandwidth and delay are taken into account. "Bandwidth" is the inverse bandwidth in Kbps times 10^7 ($10^7/BW_{Kbps}$). "Delay" is delay in tens of microseconds ($DLY_{usec}/10$). These values are added together and then scaled by a factor of 256. The composite metric is therefore represented by default as:

Metric = $(10^7/BW_{Kbps} + DLY_{usec}/10) * 256$

Using the output from the **`show ip eigrp topology 164.1.26.0 255.255.255.0`** we can see that the metric through R3 has a minimum bandwidth value of 1280Kbps and a total delay of 40100 microseconds. The metric to R3 is then calculated as:

Metric_through_R3 = $(10^7/1280 + 40100/10) * 256$
Metric_through_R3 = (7812.5 + 4010) * 256
Metric_through_R3 = (11822.5) * 256
Metric_through_R3 ~ (11822) * 256
Metric_through_R3 ~ 3026432

In order to get our ratio of 5:1 we now need to modify our calculation as follows:

Metric_through_R3 * 5 = Metric_through_R2

Or more specifically:

$(10^7/1280 + 40100/10) * 256 * 5 = (10^7/BW_{Kbps\text{-}R2} + DLY_{usec\text{-}R2}/10) * 256$

The value that we will modify through R2 is the delay, so we can use our current BW value to R2 of 256Kbps (as seen from the **`show ip eigrp topology`** output)

$(10^7/1280 + 40100/10) * 256 * 5 = (10^7/BW_{Kbps\text{-}R2} + DLY_{usec\text{-}R2}/10) * 256$
$(10^7/1280 + 40100/10) * 256 * 5 = (10^7/256 + DLY_{usec\text{-}R2}/10) * 256$
$(10^7/1280 + 40100/10) * 5 = (10^7/256 + DLY_{usec\text{-}R2}/10)$
$(7812.5 + 4010) * 5 = (39062.5 + DLY_{usec\text{-}R2}/10)$
$(7812 + 4010) * 5 \sim (39062 + DLY_{usec\text{-}R2}/10)$
$59110 \sim (39062 + DLY_{usec\text{-}R2}/10)$
$20048 \sim DLY_{usec\text{-}R2}/10$
$200480 \sim DLY_{usec\text{-}R2}$

Based on this calculation we can see that if the end to end delay through R2 is 200480 the resulting composite metric through R2 will be five times that of through R3. Looking at the **`show ip eigrp topology 164.1.26.0 255.255.255.0`** output on R2 we can see that R2 already has a delay of 100 microseconds to reach this destination:

```
Rack1R2#show ip eigrp topology 164.1.26.0 255.255.255.0
IP-EIGRP (AS 101): Topology entry for 164.1.26.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 28160
  Routing Descriptor Blocks:
  0.0.0.0 (FastEthernet0/0), from Connected, Send flag is 0x0
      Composite metric is (28160/0), Route is Internal
      Vector metric:
        Minimum bandwidth is 100000 Kbit
        Total delay is 100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 0
```

This means that R1 should have a local delay to R2 of (200480 – 100), or 20038 *tens* of microseconds. Once the **`delay 20038`** command is configured on R1's Serial0/0 interface the traffic share is in a ratio of 5 to 1:

```
Rack1R1#show ip route 164.1.26.0
Routing entry for 164.1.26.0/24
  Known via "eigrp 101", distance 90, metric 3026432, type internal
  Redistributing via eigrp 101
  Last update from 164.1.13.3 on Serial0/1, 00:00:00 ago
  Routing Descriptor Blocks:
  * 164.1.12.2, from 164.1.12.2, 00:00:00 ago, via Serial0/0
      Route metric is 15132160, traffic share count is 1
      Total delay is 200480 microseconds, minimum bandwidth is 256 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1
    164.1.13.3, from 164.1.13.3, 00:00:00 ago, via Serial0/1
      Route metric is 3026432, traffic share count is 5
      Total delay is 40100 microseconds, minimum bandwidth is 1280 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

Further Reading

How Does Unequal Cost Path Load Balancing (Variance) Work in IGRP and EIGRP?

For those of you that are not as good with calculating large equations, another method is to start with a low value and change the delay while looking at the output of **`show ip route`**: Starting without a delay configured, the ratio shows as 80/23.

```
Rack1R1(config-if)#do show ip route 164.1.26.0 | i from|share
  Last update from 164.1.12.2 on Serial0/0, 00:00:08 ago
  * 164.1.13.3, from 164.1.13.3, 00:00:08 ago, via Serial0/1
      Route metric is 3026432, traffic share count is 80
    164.1.12.2, from 164.1.12.2, 00:00:08 ago, via Serial0/0
```

```
      Route metric is 10514432, traffic share count is **23**
```

After changing to a delay of 10,000, the ratio changes to 120/29.

```
Rack1R1(config-if)#delay 10000
Rack1R1(config-if)#do show ip route 164.1.26.0 | i from|share
  Last update from 164.1.12.2 on Serial0/0, 00:00:00 ago
  * 164.1.13.3, from 164.1.13.3, 00:00:00 ago, via Serial0/1
      Route metric is 3026432, traffic share count is 120
    164.1.12.2, from 164.1.12.2, 00:00:00 ago, via Serial0/0
      Route metric is 12562432, traffic share count is 29
```

After changing to a delay of 20000, the ratio changes to 5 to 1. This is not necessarily an exact value, due to how the router handles the forwarding internally, but it is fairly close.

```
Rack1R1(config-if)#delay 20000
Rack1R1(config-if)#do show ip route 164.1.26.0 | i from|share
  Last update from 164.1.12.2 on Serial0/0, 00:00:01 ago
  * 164.1.13.3, from 164.1.13.3, 00:00:01 ago, via Serial0/1
      Route metric is 3026432, traffic share count is 5
    164.1.12.2, from 164.1.12.2, 00:00:01 ago, via Serial0/0
      Route metric is 15122432, traffic share count is 1
```

Task 2.1 Verification

Verify the topology and routing table after load-balancing configuration has been configured:

```
Rack1R1#show ip eigrp topology 164.1.26.0 255.255.255.0
IP-EIGRP (AS 100): Topology entry for 164.1.26.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is
3026432
  Routing Descriptor Blocks:
  164.1.13.3 (Serial0/1), from 164.1.13.3, Send flag is 0x0
      Composite metric is (3026432/2514432), Route is Internal
      Vector metric:
        Minimum bandwidth is 1280 Kbit
        Total delay is 40100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
  164.1.12.2 (Serial0/0), from 164.1.12.2, Send flag is 0x0
      Composite metric is (15132160/28160), Route is Internal
      Vector metric:
        Minimum bandwidth is 256 Kbit
        Total delay is 200480 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1

Rack1R1#show ip route 164.1.26.0
```

```
Routing entry for 164.1.26.0/24
  Known via "eigrp 100", distance 90, metric 3026432, type internal
  Redistributing via eigrp 100
  Last update from 164.1.13.3 on Serial0/1, 00:02:05 ago
  Routing Descriptor Blocks:
  * 164.1.12.2, from 164.1.12.2, 00:02:05 ago, via Serial0/0
      Route metric is 15132160, traffic share count is 1
      Total delay is 200480 microseconds, minimum bandwidth is 256 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1
    164.1.13.3, from 164.1.13.3, 00:02:05 ago, via Serial0/1
      Route metric is 3026432, traffic share count is 5
      Total delay is 40100 microseconds, minimum bandwidth is 1280 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

Task 2.2

R3:
```
router ospf 1
 router-id 150.1.3.3
 network 164.1.34.3 0.0.0.0 area 0
 network 164.1.35.3 0.0.0.0 area 0
```

R4:
```
interface Serial0/0/0xit
 ip ospf network point-to-point
!
router ospf 1
 router-id 150.1.4.4
 network 164.1.34.4 0.0.0.0 area 0
```

R5:
```
interface Serial0/0
 ip ospf network point-to-point
!
router ospf 1
 router-id 150.1.5.5
 network 164.1.5.5 0.0.0.0 area 0
 network 164.1.35.5 0.0.0.0 area 0
 network 164.1.55.5 0.0.0.0 area 0
```

Task 2.2 Verification

Rack1R3#show ip ospf neighbor

```
Neighbor ID     Pri State           Dead Time   Address         Interface
150.1.5.5         0   FULL/  -        00:00:38    164.1.35.5      Serial1/0.35
150.1.4.4         0   FULL/  -        00:00:35    164.1.34.4      Serial1/0.34
```

Verify OSPF routes:

Rack1R3#show ip route ospf
```
     164.1.0.0/16 is variably subnetted, 19 subnets, 3 masks
O IA    164.1.32.0/24 [110/784] via 164.1.34.4, 00:15:26, Serial1/0.34
O IA    164.1.45.0/29 [110/782] via 164.1.34.4, 00:15:26, Serial1/0.34
```

```
O IA    164.1.47.0/24 [110/782] via 164.1.34.4, 00:15:26, Serial1/0.34
O IA    164.1.43.0/24 [110/784] via 164.1.34.4, 00:15:26, Serial1/0.34
O       164.1.55.0/24 [110/782] via 164.1.35.5, 00:15:26, Serial1/0.35
O       164.1.5.0/24 [110/782] via 164.1.35.5, 00:15:26, Serial1/0.35
O IA    164.1.7.0/24 [110/783] via 164.1.34.4, 00:15:26, Serial1/0.34
O IA    164.1.14.0/24 [110/783] via 164.1.34.4, 00:15:26, Serial1/0.34
O IA    164.1.31.0/24 [110/783] via 164.1.34.4, 00:15:26, Serial1/0.34
O IA    164.1.24.0/24 [110/784] via 164.1.34.4, 00:15:26, Serial1/0.34
     150.1.0.0/16 is variably subnetted, 9 subnets, 2 masks
O IA    150.1.5.0/24 [110/782] via 164.1.35.5, 00:15:26, Serial1/0.35
O IA    150.1.4.0/24 [110/782] via 164.1.34.4, 00:15:26, Serial1/0.34
O IA    150.1.10.10/32 [110/784] via 164.1.34.4, 00:15:27, Serial1/0.34
O IA    150.1.9.9/32 [110/784] via 164.1.34.4, 00:15:27, Serial1/0.34
O IA    150.1.7.7/32 [110/783] via 164.1.34.4, 00:15:27, Serial1/0.34
Rack1R3#
```

Task 2.3

R6:

```
router rip
 version 2
 no auto-summary
 network 54.0.0.0
 redistribute eigrp 100 metric 1
!
router eigrp 100
 redistribute rip metric 10000 1000 1 255 1500
```

Task 2.3 Verification

Verify the RIP routes received from BB1:

```
Rack1R6#show ip route rip
R    212.18.1.0/24 [120/1] via 54.1.2.254, 00:00:01, Serial0/0/0
R    212.18.0.0/24 [120/1] via 54.1.2.254, 00:00:01, Serial0/0/0
R    212.18.3.0/24 [120/1] via 54.1.2.254, 00:00:01, Serial0/0/0
R    212.18.2.0/24 [120/1] via 54.1.2.254, 00:00:01, Serial0/0/0
```

Verify redistribution:

```
Rack1R1#show ip route eigrp | include D EX
D EX    54.1.2.0 [170/15388160] via 164.1.12.2, 00:00:40, Serial0/0
D EX 212.18.1.0/24 [170/15388160] via 164.1.12.2, 00:00:40, Serial0/0
D EX 212.18.0.0/24 [170/15388160] via 164.1.12.2, 00:00:40, Serial0/0
D EX 212.18.3.0/24 [170/15388160] via 164.1.12.2, 00:00:40, Serial0/0
D EX 212.18.2.0/24 [170/15388160] via 164.1.12.2, 00:00:40, Serial0/0
```

```
Rack1R6#show ip rip database | include redistributed
150.1.1.0/24    redistributed
150.1.2.0/24    redistributed
150.1.3.0/24    redistributed
150.1.4.0/23    redistributed
<snip>
```

Task 2.4

R3:

```
interface Serial1/1.23
 ip summary-address eigrp 100 150.1.4.0 255.255.254.0
!
interface Serial1/2
 ip summary-address eigrp 100 150.1.4.0 255.255.254.0
!
router ospf 1
 redistribute eigrp 100 subnets metric 10
!
router eigrp 100
 redistribute ospf 1 metric 10000 1000 1 255 1500
```

SW2:

```
interface FastEthernet0/15
 ip summary-address eigrp 100 150.1.4.0 255.255.254.0 5
!
router ospf 1
 redistribute eigrp 100 subnets
!
router eigrp 100
 redistribute ospf 1 metric 10000 1000 1 255 1500
```

Task 2.4 Verification

```
Rack1R1#show ip route 150.1.4.0
Routing entry for 150.1.4.0/23
  Known via "eigrp 100", distance 90, metric 514560, type internal
  Redistributing via eigrp 100
  Last update from 164.1.18.8 on FastEthernet0/0, 00:00:51 ago
  Routing Descriptor Blocks:
  * 164.1.18.8, from 164.1.18.8, 00:00:51 ago, via FastEthernet0/0
      Route metric is 514560, traffic share count is 1
      Total delay is 10100 microseconds, minimum bandwidth is 10000
Kbit
      Reliability 1/255, minimum MTU 1500 bytes
      Loading 255/255, Hops 1
```

Task 2.5

R3:

```
router eigrp 100
 redistribute ospf 1 metric 10000 1000 1 255 1500 route-map OSPF->EIGRP
!
router ospf 1
 redistribute eigrp 100 subnets tag 390 metric 10
!
route-map OSPF->EIGRP deny 10
 match tag 890
!
route-map OSPF->EIGRP permit 20
```

SW2:

```
router eigrp 100
 redistribute ospf 1 metric 10000 1000 1 255 1500 route-map OSPF->EIGRP
!
router ospf 1
 redistribute eigrp 100 subnets tag 890
```

```
!
route-map OSPF->EIGRP deny 10
 match tag 390
!
route-map OSPF->EIGRP permit 20
```

Task 2.5 Breakdown

Commonly, with route redistribution there is more than one possible solution to resolve most issues. In this task, route tags were used to ensure that any new routes redistributed into EIGRP on R6 will not be passed back into EIGRP from OSPF on R3 or SW2. You may notice that suboptimal routing may occur on R3 or SW2 to reach the routes redistributed on R6, but unless specifically asked for in the task, suboptimal routing is not necessarily an issue that needs to be resolved. Remember that the lab is just looking for reachability and not “optimal reachability”.

☠ Pitfall

Note that the EIGRP to OSPF redistribution metric is lower on R3 than it is on SW2. R3’s Loopback0 network is advertised as an internal route into EIGRP only, which means the OSPF domain will have to choose between R3 and SW2 as the exit points to reach the external route. If the prefix is routed out via SW2, reachability problems will occur once the BGP section is complete.

Although this configuration technically doesn’t relate directly to the points associated with the redistribution section, remember that stable IGP reachability is always a requirement of a fully functional BGP network.

Task 2.6

R3:
```
router ospf 1
 default-information originate route-map CONDITIONAL_DEFAULT
!
ip prefix-list R1_or_R2 seq 5 permit 164.1.13.0/24
ip prefix-list R1_or_R2 seq 10 permit 164.1.23.0/24
!
route-map CONDITIONAL_DEFAULT permit 10
 match ip address prefix-list R1_or_R2
```

Task 2.6 Verification

Check default route, when both R3’s EIGRP-enabled links are up:

```
Rack1R5#show ip route ospf | include 0.0.0.0
O*E2 0.0.0.0/0 [110/1] via 164.1.35.3, 00:00:24, Serial0/0
```

Shutdown both of the EIGRP enabled links at R3 and observe the output *from the debug:*

```
Rack1R3#debug ip ospf lsa-generation
OSPF summary lsa generation debugging is on
Rack1R3#conf t
Rack1R3(config)#interface s1/1.23
Rack1R3(config-subif)#shutdown
Rack1R3(config)#interface s1/2
Rack1R3(config-if)#shutdown

OSPF: Generate external LSA 0.0.0.0, mask 0.0.0.0, type 5, age 3600,
metric 16777215, tag 1, metric-type 2, seq 0x80000002
OSPF: 0.0.0.0/0 type: 5 is already maxaged
```

Verify that OSPF domain lost default route:

```
Rack1R5#show ip route ospf | include 0.0.0.0
Rack1R5#
```

Use the TCL script below to test reachability:

```
foreach i {
150.1.1.1
164.1.12.1
164.1.13.1
164.1.18.1
150.1.2.2
164.1.12.2
164.1.23.2
164.1.26.2
164.1.34.3
164.1.35.3
150.1.3.3
164.1.13.3
164.1.23.3
164.1.34.4
164.1.45.4
164.1.47.4
150.1.4.4
164.1.35.5
164.1.45.5
164.1.55.5
150.1.5.5
164.1.5.5
54.1.2.6
150.1.6.6
164.1.26.6
164.1.47.7
150.1.7.7
164.1.7.7
164.1.14.7
164.1.31.7
150.1.8.8
164.1.24.8
164.1.32.8
164.1.18.8
164.1.43.9
150.1.9.9
164.1.31.9
164.1.32.9
164.1.43.10
150.1.10.10
164.1.14.10
164.1.24.10

} { ping $i  }
```

Note that VLAN43, VLAN62, and VLAN3 are not a part of any IGP and are not tested for reachability.

Task 2.7

R3:

```
router bgp 200
 network 164.1.3.0 mask 255.255.255.0
```

R4:

```
router bgp 100
 aggregate-address 164.1.0.0 255.255.0.0 summary-only
```

R6:

```
router bgp 200
 aggregate-address 164.1.0.0 255.255.0.0 summary-only
```

Task 2.7 Verification

Verify the summary generation. For instance on R6:

```
Rack1R6#show ip bgp | include 164|Net
   Network          Next Hop            Metric LocPrf Weight Path
*> 164.1.0.0        0.0.0.0                            32768 i
s>i164.1.3.0/24     164.1.23.3               0    100      0 i
```

Task 2.8

```
R1:
router bgp 300
 neighbor 164.1.18.8 default-originate
 neighbor 164.1.18.8 prefix-list DEFAULT out
!
ip prefix-list DEFAULT seq 5 permit 0.0.0.0/0

SW2:
router ospf 1
 distribute-list PREFER_DEFAULT_VIA_BGP in
!
ip access-list standard PREFER_DEFAULT_VIA_BGP
 deny   0.0.0.0
 permit any
```

Task 2.8 Breakdown

Recall from the previous OSPF configuration that R3 originated a default route into OSPF. When SW2 learns the iBGP default route from R1, and the OSPF External default route from SW3 and SW4, the OSPF route with the lower administrative distance is preferred. To ensure that the BGP route is installed, a distribute-list is applied to the routing table on SW2 to stop the OSPF default route from being used. The result of this is that the iBGP learned default route from R1 makes it into the table.

Task 2.8 Verification

```
Rack1R1#show ip bgp neighbors 164.1.18.8 advertised-routes
BGP table version is 36, local router ID is 150.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

Originating default network 0.0.0.0

   Network          Next Hop            Metric LocPrf Weight Path

Total number of prefixes 0
```

Verify BGP routes on SW2:

```
Rack1SW2#show ip route bgp
B*   0.0.0.0/0 [200/0] via 164.1.18.1, 00:01:53
```

Task 2.9

```
R2:
ip as-path access-list 1 permit ^$
!
router bgp 200
 neighbor 164.1.12.1 filter-list 1 out

R3:
ip as-path access-list 1 permit ^$
!
router bgp 200
 neighbor 164.1.13.1 filter-list 1 out
```

Task 2.9 Breakdown

The above task states that AS 200 cannot be used as transit for users in AS 300. Therefore by only advertising prefixes that were originated inside AS 200, AS 300 cannot use AS 200 to reach any other ASs. In the above solution this is accomplished through the usage of filtering based on AS-Path information.

Since the AS-Path of a prefix is not added until the prefix leaves the AS, prefixes which have been originated within the AS will have an empty AS-Path. This can be easily matched with a regular expression which specifies that the end of the line comes immediately after the end of the line, and is denoted as ^$.

☑ Verification

The following route server output shows a match for locally originated routes using the regular expression ^$

```
[root@CoachZ /]#telnet route-server.net
############## route-server.xx.net ###############
     #########  xx Route Monitor  ###########

This router maintains peerings with customer-facing routers
throughout the xx Backbone:

 <output deleted>

This router has the global routing table view from each of the above
routers, providing a glimpse to the Internet routing table from the
xx network's perspective.

route-server>show ip bgp regexp ^$
BGP table version is 28963851, local router ID is 209.1.220.234
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i24.241.191.0/24  208.172.146.29                100      0 i
```

```
* i                 208.172.146.30                100      0 i
*>i62.208.90.0/24   208.172.146.29                100      0 i
* i                 208.172.146.30                100      0 i
*>i62.208.125.0/24  208.172.146.29                100      0 i
* i                 208.172.146.30                100      0 i
*>i62.221.5.144/28  208.172.146.29                100      0 i
* i                 208.172.146.30                100      0 i
*>i62.221.5.208/28  208.172.146.29                100      0 i
* i                 208.172.146.30                100      0 i
*>i62.221.5.224/28  208.172.146.29                100      0 i
* i                 208.172.146.30                100      0 i
*>i63.128.32.0/20   208.172.146.29                100      0 i
* i                 208.172.146.30                100      0 i
*>i63.128.32.68/32  208.172.146.29                100      0 i
* i                 208.172.146.30                100      0 i
```

Task 2.9 Verification

Verify the routes that R2 and R3 advertise to AS 300:

```
Rack1R2#show ip bgp neighbors 164.1.12.1 advertised-routes
BGP table version is 17, local router ID is 150.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i164.1.0.0        164.1.26.6               0    100      0 i
*>i164.1.3.0/24     164.1.23.3               0    100      0 i

Total number of prefixes 2

Rack1R3#show ip bgp neighbors 164.1.13.1 advertised-routes
BGP table version is 17, local router ID is 150.1.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i164.1.0.0        164.1.26.6               0    100      0 i
*> 164.1.3.0/24     0.0.0.0                  0         32768 i

Total number of prefixes 2
```

Task 2.10

```
R4:
router ospf 1
 redistribute bgp 100 subnets route-map BGP2OSPF
!
ip as-path access-list 1 permit ^54_
!
route-map BGP2OSPF permit 10
 match as-path 1
```

Task 2.10 Breakdown

The above task describes a case where reachability is lost to certain BGP networks when the primary Frame Relay connection of R4 is down. When the Frame Relay connection is down, all of R4's traffic destined to R3 must transit R5. The problem, however, is that R5 does not participate in BGP routing. Therefore, although BGP network layer reachability information is successfully transmitted throughout the network, traffic may be black holed when it reaches R5.

In order to resolve this issue, BGP has been redistributed into IGP. R4 has been configured to redistribute all BGP information learned from AS 54 into OSPF. For traffic in the opposite direction, it doesn't matter, since R3 is originating a default route.

Task 2.10 Verification

Before applying the solution, verify reachability to AS54's prefixes when R4's Frame Relay link is up:

```
Rack1R3#trace 28.119.16.0

Type escape sequence to abort.
Tracing the route to 28.119.16.0

  1 164.1.34.4 28 msec 32 msec 28 msec
  2 204.12.1.254 32 msec 28 msec 32 msec
```

Now shutdown R4's Serial0/0/0 interface and repeat the traceroute:

```
Rack1R3#trace 28.119.16.0

Type escape sequence to abort.
Tracing the route to 28.119.16.0

  1 164.1.35.5 32 msec 28 msec 28 msec
  2 164.1.35.3 72 msec 57 msec 60 msec
```

Try traceroute after redistribution has been applied:

```
Rack1R3#trace 28.119.16.0

Type escape sequence to abort.
Tracing the route to 28.119.16.0

  1 164.1.35.5 32 msec 28 msec 32 msec
  2 164.1.45.4 28 msec 28 msec 32 msec
  3 204.12.1.254 32 msec 32 msec 32 msec
```

Verify the OSPF routes on R5:

```
Rack1R5#show ip route ospf | i \.4,
O E2    28.119.17.0 [110/1] via 164.1.45.4, 00:02:37, Serial0/1/0
O E2    28.119.16.0 [110/1] via 164.1.45.4, 00:02:37, Serial0/1/0
O       150.1.4.0/24 [110/65] via 164.1.45.4, 02:17:03, Serial0/1/0
Rack1R5#
```

Task 3.1

R1:
```
ipv6 unicast-routing
!
interface Serial0/1
 ipv6 address 2001:164:1:13::1/64
!
interface Serial0/0
 ipv6 address 2001:164:1:12::1/64
 frame-relay map ipv6 2001:164:1:12::2 102 broadcast
```

R2:
```
ipv6 unicast-routing
!
interface Serial0/0.12 point-to-point
 ipv6 address 2001:164:1:12::2/64
!
interface Serial0/0.23 point-to-point
 ipv6 address 2001:164:1:23::2/64
```

R3:
```
ipv6 unicast-routing
!
interface Serial1/2
 ipv6 address 2001:164:1:13::3/64
!
interface Serial1/1.23 point-to-point
 ipv6 address 2001:164:1:23::3/64
```

Task 3.1 Verification

```
Rack1R2#show ipv6 interface brief
FastEthernet0/0            [up/up]
Serial0/0                  [up/up]
Serial0/0.12               [up/up]
    FE80::20D:BCFF:FE16:2720
    2001:164:1:12::2
Serial0/0.23               [up/up]
    FE80::20D:BCFF:FE16:2720
    2001:164:1:23::2
Serial0/1                  [administratively down/down]
Loopback0                  [up/up]

Rack1R2#ping 2001:164:1:12::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:164:1:12::1, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms

Rack1R2#ping 2001:164:1:23::3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:164:1:23::3, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/60 ms

Rack1R1#ping 2001:164:1:13::3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:164:1:13::3, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
```

Task 3.2

```
R1:
interface Serial0/0
 ipv6 ospf 1 area 123
 ipv6 ospf network point-to-point
 frame map ipv6 fe80::2 102
!
interface Serial0/1
 ipv6 ospf 1 area 123
 ipv6 ospf cost 455

R2:
interface Serial0/0.12 point-to-point
 ipv6 ospf 1 area 123
 ipv6 address fe80::2 link-local
!
interface Serial0/0.23
 ipv6 ospf 1 area 123
```

```
R3:
interface Loopback100
 ipv6 address 2001:150:1:3::3/64
 ipv6 ospf 1 area 123
!
interface Serial1/1.23 point-to-point
 ipv6 ospf 1 area 123
!
interface Serial1/2
 ipv6 ospf 1 area 123
```

Task 3.2 Verification

```
Rack1R1#show ipv6 route ospf
IPv6 Routing Table - 8 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS
summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
O   2001:150:1:3::3/128 [110/454]
     via FE80::2, Serial0/0
O   2001:164:1:23::/64 [110/454]
     via FE80::2, Serial0/0
Rack1R1#


Rack1R1#traceroute 2001:150:1:3::3

Type escape sequence to abort.
Tracing the route to 2001:150:1:3::3

  1 2001:164:1:12::2 40 msec 40 msec 44 msec
  2 2001:150:1:3::3 48 msec 88 msec 49 msec
```

Task 3.2 Verification

There are a couple things to watch for here. There will be a network type mismatch between R2 and R1, and the metrics need to be manipulated so that R1 prefers the connection to R3's loopback via R2. Since R1 will have the network learned from R2 resolved in the routing table "via" the link-local address of R2, make sure that you also have a frame map for that address. Here, the link-local address is manually configured for simplicity.

Task 4.1

```
SW1, SW2, SW3, SW4:
vlan 99

SW3:
interface vlan 99
 ip address 99.99.99.3 255.255.255.0

SW4:
interface vlan 99
 Ip address 99.99.99.4 255.255.255.0
```

You may want to ping at this point to verify connectivity, before adding the VRFs, to verify the traffic can flow between SW3 and SW4. Depending on how your root bridge is elected, the traffic may pass through SW1 or SW2, so make sure you have defined VLAN 99 on those switches.

```
Rack1SW4#ping 99.99.99.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 99.99.99.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Rack1SW4#
```

```
SW3, SW4:
Sdm prefer extended-match

SW3:
ip vrf SW3TEST
 Rd 99:3
!
interface vlan 99
 ip vrf forwarding SW3TEST
 ip address 99.99.99.3

SW4:
ip vrf SW4TEST
 rd 99:4
!
interface vlan 99
 ip vrf forwarding SW4TEST
```

Make sure that you test by pinging from within the VRF.

```
Rack1SW4#ping vrf SW4TEST 99.99.99.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 99.99.99.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms

Rack1SW4#ping vrf SW4TEST 99.99.99.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 99.99.99.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Rack1SW4#

Rack1SW4#show ip route vrf SW4TEST

Routing Table: SW4TEST
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     99.0.0.0/24 is subnetted, 1 subnets
C       99.99.99.0 is directly connected, Vlan99
Rack1SW4#
```

Task 5.1

```
R3:
interface Loopback0
 ip pim sparse-dense-mode

R2, R3, R4, and SW1:
ip pim rp-address 150.1.3.3 3
!
access-list 3 permit 225.10.0.0 0.48.255.255
```

Task 5.1 Breakdown

To find the minimum amount of statements to match these groups first examine the groups:

```
225.10.0.0 - 225.10.255.255
```

```
225.26.0.0 - 225.26.255.255
225.42.0.0 - 225.42.255.255
225.58.0.0 - 255.58.255.255
```

From this output it is evident that the first octet will always be 225, and the third and fourth octets can be anything. Next write out the second octet in binary for comparison:

```
10 = 00001010
26 = 00011010
42 = 00101010
58 = 00111010
```

These four networks differ in only the 3rd and 4th most significant bits, and can be matched with a wildcard mask as follows:

```
      10 = 00001010
      26 = 00011010
      42 = 00101010
      58 = 00111010
Wildcard = 00110000
```

Resulting in: access-list 3 permit 225.10.0.0 0.48.255.255

Task 5.2

```
R4:
interface Loopback0
 ip pim sparse-dense-mode

R2, R3, R4, and SW1:
ip pim rp-address 150.1.4.4 4
!
access-list 4 permit 226.37.0.0 1.8.255.255
```

Task 5.2 Breakdown

To find the minimum amount of statements to match these groups first examine the groups:

```
226.37.0.0 - 226.37.255.255
226.45.0.0 - 226.45.255.255
227.37.0.0 - 227.37.255.255
227.45.0.0 - 227.45.255.255
```

From this output it is evident that the third and fourth octets can be anything. Next write out the first and second octets in binary for comparison:

```
226 = 11100010
227 = 11100011

 37 = 00100101
 45 = 00101101
```

These bit patterns result in four combinations which can be matched as follows:

```
                   226 = 11100010
                   227 = 11100011
              Wildcard = 00000001

                   37 = 00100101
                   45 = 00101101
             Wildcard = 00001000

             226.37 = 11100010.00100101
             226.45 = 11100011.00101101
             227.37 = 11100010.00100101
             227.45 = 11100011.00101101
           Wildcard = 00000001.00001000
```

Resulting in: `access-list 4 permit 226.37.0.0 1.8.255.255`

Tasks 5.1 – 5.2 Verification

Verify static RP configuration, for instance on R2:

```
Rack1R2#show ip pim rp mapping
PIM Group-to-RP Mappings

Acl: 3, Static
    RP: 150.1.3.3 (?)
Acl: 4, Static
    RP: 150.1.4.4 (?)

Rack1R2#show ip access-lists 3
Standard IP access list 3
    10 permit 225.10.0.0, wildcard bits 0.48.255.255

Rack1R2#show ip access-lists 4
Standard IP access list 4
    10 permit 226.37.0.0, wildcard bits 1.8.255.255
```

Task 5.3

R3:

```
interface FastEthernet0/1
 ip multicast boundary 1
 ip igmp query-max-response-time 3
 ip igmp query-interval 5
!
access-list 1 deny   226.37.1.1
access-list 1 permit any
```

Task 5.3 Verification

Verify IGMP configuration at R3:

```
Rack1R3#show ip igmp interface fa0/1
```

```
FastEthernet0/1 is up, line protocol is up
  Internet address is 164.1.3.3/24
  IGMP is enabled on interface
  Current IGMP host version is 2
  Current IGMP router version is 2
  IGMP query interval is 5 seconds
  IGMP querier timeout is 10 seconds
  IGMP max query response time is 3 seconds
  Last member query count is 2
  Last member query response interval is 1000 ms
  Inbound IGMP access group is not set
  IGMP activity: 1 joins, 0 leaves
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 164.1.3.3 (this system)
  IGMP querying router is 164.1.3.3 (this system)
  No multicast groups joined by this system
```

Verify the multicast boundary configuration:

```
Rack1R3#show ip multicast interface fa0/1
FastEthernet0/1 is up, line protocol is up
  Internet address is 164.1.3.3/24
  Multicast routing: enabled
  Multicast switching: fast
  Multicast packets in/out: 0/0
                      1
  Multicast TTL threshold: 0
  Multicast Tagswitching: disabled
```

Task 6.1

SW1:

```
username RDP password 0 CISCO
!
interface Vlan41
 ip access-group REMOTE_DESKTOP in
!
interface FastEthernet0/13
 ip access-group REMOTE_DESKTOP in
!
interface Port-channel14
 ip access-group REMOTE_DESKTOP in
!
ip access-list extended REMOTE_DESKTOP
 dynamic RDP timeout 10 permit tcp any host 164.1.7.100 eq 3389
 deny   tcp any host 164.1.7.100 eq 3389
 permit ip any any
!
line vty 0 4
 login local
 autocommand access-enable host
```

Task 6.1 Breakdown

This type of access-list configuration is known as a lock-and-key, or a dynamic access-list. When the access-list is applied, the dynamic entry does not yet exist in the list. This is similar to how an entry can be inactive when referencing a time range. When the command **`access-enable`** is executed, all dynamic entries are inserted into the access-list.

The command **`autocommand access-enable`** means that when a user logs in via the VTY line, the command **`access-enable`** will automatically execute. This is simply a way to automate the running of the command. The **`autocommand access-enable`** command can also be placed in the local user database on a per user basis. In the above case the autocommand applies to anyone entering the VTY lines.

The **`host`** option of the access-enable statement dictates that only the host that authenticated will be allowed access through the dynamic statement. This is accomplished by dynamically creating a copy of the configured dynamic entry, or entries, with the source IP address as the authenticated address.

The timeout value of 10 minutes applied to the access-list is an absolute timeout, and the user will have to re-authenticate once this timer expires.

Task 6.1 Verification

```
Rack1SW1#show ip access-lists REMOTE_DESKTOP
Extended IP access list REMOTE_DESKTOP
    10 Dynamic RDP permit tcp any host 164.1.7.100 eq 3389
    20 deny tcp any host 164.1.7.100 eq 3389
    30 permit ip any any (8 matches)
Rack1SW1#

Rack1R4#telnet 164.1.47.7
Trying 164.1.47.7 ... Open

User Access Verification

Username: RDP
Password: CISCO

Rack1SW1#show ip access-lists REMOTE_DESKTOP
Extended IP access list REMOTE_DESKTOP
    10 Dynamic RDP permit tcp any host 164.1.7.100 eq 3389
    10   permit tcp host 164.1.47.4 host 164.1.7.100 eq 3389
    20 deny tcp any host 164.1.7.100 eq 3389
    30 permit ip any any (31 matches)
```

Task 6.2

```
SW1:
username NOC password 0 CISCO
!
access-list 100 permit tcp any any eq telnet
access-list 100 permit tcp any any eq 3023
!
line vty 0
 login local
 autocommand access-enable host
!
line vty 1 4
 no autocommand access-enable
 access-class 100 in
 rotary 23
```

Task 6.2 Breakdown

Since the command **`autocommand access-enable`** applies to all users starting an exec process through the VTY line, regular telnet access at port 23 is no longer available for the management on the CLI. In order to still allow users to be able to telnet into the router to manage it, the properties applied to the VTY lines have been split into two.

The first VTY line (VTY 0) is left with the **`autocommand access-enable`** command. All users that telnet to the router at port 23 will hit this line. The **`rotary`** command under the VTY line allows the router to listen for telnet sessions at higher port ranges (30xx, 50xx, 70xx, 100xx, where x is the configured rotary option), so users can still telnet in to access the CLI.

Task 6.2 Verification

```
Rack1SW1#telnet 150.1.7.7 3023
Trying 150.1.7.7, 3023 ... Open


User Access Verification

Username: NOC
Password: CISCO
Rack1SW1>
```

Task 6.3

R6:
```
ip access-list extended FILTER
 permit udp 164.1.0.0 0.0.255.255 any eq 53
!
ip traffic-export profile EXPORT
  interface FastEthernet0/1
  incoming access-list FILTER
  mac-address 1234.5678.9abc
  exit
!
interface FastEthernet 0/1
 ip traffic-export apply EXPORT
```

Task 6.3 Verification

```
Rack1R6#show ip traffic-export
Router IP Traffic Export Parameters
Monitored Interface             FastEthernet0/1
        Export Interface                FastEthernet0/1
        Destination MAC address 1234.5678.9abc
        bi-directional traffic export is off
Input IP Traffic Export Information     Packets/Bytes Exported    0/0
        Packets Dropped           3
        Sampling Rate             one-in-every 1 packets
        Access List        FILTER [named extended IP]
        Profile EXPORT is Active
```

Task 6.4

R4:
```
Load protocol system:/fpm/phdf/ip.phdf
Load protocol system:/fpm/phdf/udp.phdf

class-map type stack match-all MYUDP
 match field ip prot eq 0x11 next udp

class-map type access-control match-all SLAM
 match field udp dest-port eq 0x59A
 match field ip length eq 0x194
 match start l3-start offset 224 size 4 eq 0x4011010

policy-map type access-control MYFPM
  class SLAM
   drop
policy-map type access-control MYINT
 class MYUDP
  service-policy MYFPM
interface Fa0/1
 service-policy type access-control input MYINT
```

Task 6.4 Breakdown

Copyright © 2009 Internetwork Expert www.INE.com

Flexible Packet Matching is a very powerful tool on the router. Some people call it an "access-list on steroids". Configuration can be a bit tricky, however, so it is recommended that you save before starting. Newer versions of IOS include the PHDF files in the system folder, so you may not need to copy the files to flash. After loading the PHDF files, the fields can be matched in the class maps. A nested policy is used, referencing the stack class map in the parent policy.

```
Rack1R4#show policy-map type access-con int fa0/1
 FastEthernet0/1
  Service-policy access-control input: MYINT
    Class-map: MYUDP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps
      Match: field IP protocol eq 0x11 next UDP

      Service-policy access-control : MYFPM

        Class-map: SLAM (match-all)
          0 packets, 0 bytes
          5 minute offered rate 0 bps
          Match: field UDP dest-port eq 0x59A
          Match: field IP length eq 0x194
          Match: start l3-start offset 224 size 4 eq 0x4011010
      drop

        Class-map: class-default (match-any)
          0 packets, 0 bytes
          5 minute offered rate 0 bps, drop rate 0 bps
          Match: any
    Class-map: class-default (match-any)
      7 packets, 472 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
Rack1R4#
```

Task 6.5

R6:

```
class-map type inspect match-any TCPUDP
 match protocol tcp
 match protocol udp
!
class-map type inspect match-any ICMP
 match protocol icmp
!
policy-map type inspect TEST
 class TCPUDP
  inspect
 class ICMP
```

```
  pass
!
policy-map type inspect ICMP
 class ICMP
  pass
!
zone security inside
zone security outside
!
zone-pair security INOUT source inside destination outside
 service-policy type inspect TEST
!
zone-pair security OUTIN source outside destination inside
 service-policy type inspect ICMP
!
interface Fa0/0
 zone-member security inside
!
interface Fa0/1
 zone-member security inside
!
interface Serial 0/0/0
 zone-member security outside
```

Task 6.5 Breakdown

Start with class maps to define the traffic, configure policy maps with actions for the traffic classes, and apply the policies to the zone pair. Configure the interfaces for the appropriate zones.

Task 6.5 Verification

To generate some traffic, you can telnet to BB1 from R2 and issue “show ip route”. For ICMP traffic, you can ping from R2 to BB1.

```
Rack1R6#show policy-map type inspect zone-pair INOUT

policy exists on zp INOUT
 Zone-pair: INOUT

  Service-policy inspect : TEST

    Class-map: TCPUDP (match-any)
      Match: protocol tcp
        1 packets, 24 bytes
        30 second rate 0 bps
      Match: protocol udp
        0 packets, 0 bytes
        30 second rate 0 bps

   Inspect
        Packet inspection statistics [process switch:fast switch]
        tcp packets: [0:99]
        Session creations since subsystem startup or last reset 1
        Current session counts (estab/half-open/terminating) [1:0:0]
        Maxever session counts (estab/half-open/terminating) [1:1:0]
        Last session created 00:00:37
        Last statistic reset never
        Last session creation rate 1
        Maxever session creation rate 1
        Last half-open session total 0


    Class-map: ICMP (match-any)
      Match: protocol icmp
        584 packets, 46720 bytes
        30 second rate 8000 bps
      Pass
        584 packets, 46720 bytes

    Class-map: class-default (match-any)
      Match: any
      Drop
        0 packets, 0 bytes

Rack1R6#**show policy-map type inspect zone-paiR INOUT session**

policy exists on zp INOUT
 Zone-pair: INOUT

  Service-policy inspect : TEST

    Class-map: TCPUDP (match-any)
      Match: protocol tcp
        1 packets, 24 bytes
        30 second rate 0 bps
      Match: protocol udp
        0 packets, 0 bytes
        30 second rate 0 bps

   Inspect

      Number of Established Sessions = 1
      Established Sessions
        Session 4960EAA0 (164.1.26.2:17775)=>(54.1.2.254:23) tcp
SIS_OPEN
          Created 00:00:46, Last heard 00:00:25
          Bytes sent (initiator:responder) [52:9817]


    Class-map: ICMP (match-any)
      Match: protocol icmp
        0 packets, 0 bytes
        30 second rate 0 bps
      Pass
        0 packets, 0 bytes

    Class-map: class-default (match-any)
      Match: any
      Drop
        0 packets, 0 bytes
Rack1R6#
```

Task 6.6

```
R5:
username CISCO secret CISCO
username INTERN secret INTERN
aaa new-model
aaa authentication login default none
aaa authentication login VTY local
line vty 0 988
 login authent VTY
```

Telnet to R5, and log in with one of the user accounts.
Start by switching to VIEW mode, with the command **enable VIEW**. Note: This is a global command, not a configuration command.

```
R5:
enable VIEW
parser view INT
 secret cisco
 commands exec include show int
 commands exec include show clock
aaa authorization exec VTY local
username CISCO view root
username INTERN view INT
line vty 0 988
 authorization exec VTY
```

Task 6.6 Breakdown

Role based CLI allows you to be very granular for the access that you are giving people. Commands can be assigned to views, and then the users tied to the individual roles. You can also create superviews, which bundle the capability of multiple individual views. Role-based CLI does require AAA to be configured.

Task 7.1

```
R4:
ntp master 2
!
interface FastEthernet0/0
 ntp broadcast

SW1:
interface Vlan41
 ntp broadcast client
```

Task 7.1 Verification

```
Rack1SW1#show ntp status
Clock is synchronized, stratum 3, reference is 164.1.47.4
nominal freq is 119.2092 Hz, actual freq is 119.2094 Hz, precision is
2**17
reference time is CE5C491B.7AEF095A (05:35:23.480 UTC Thu Sep 17 2009)
clock offset is -0.0261 msec, root delay is 1.53 msec
root dispersion is 16812.73 msec, peer dispersion is 15875.02 msec
```

Task 7.2

```
R3:
interface FastEthernet0/0
 ip dhcp client hostname ROUTER3
 ip dhcp client lease 1 4 0
 ip address dhcp
```

Task 7.2 Breakdown

Make sure that you read the task carefully. The task is just asking you to configure the interface for DHCP, not to configure the DHCP server.

Task 7.3

```
R3:
kron occurrence TASK7.3-O in 3:0 recurring
 policy-list TASK7.3
!
kron policy-list TASK7.3
 cli renew dhcp FastEthernet 0/0
```

Task 7.3 Verification

Kron will allow you to schedule a recurring task.

```
Rack1R3#show kron sched
Kron Occurrence Schedule
TASK7.3-O inactive, will run again in 0 days 02:59:29
```

Task 7.4

```
R5:
interface FastEthernet 0/0
 dampening 30 1000 2000 30 restart 2000
!
interface FastEthernet 0/1
 dampening 30 1000 2000 30 restart 2000
```

Task 7.4 Verification

```
Rack1R5#show interfaces dampening
FastEthernet0/0
  Flaps Penalty    Supp ReuseTm   HalfL  ReuseV   SuppV  MaxSTm    MaxP Restart
      0       0   FALSE       0      30    1000    2000      30    2000    2000
FastEthernet0/1
  Flaps Penalty    Supp ReuseTm   HalfL  ReuseV   SuppV  MaxSTm    MaxP Restart
      0       0   FALSE       0      30    1000    2000      30    2000    2000
```

Task 7.4 Breakdown

For the dampening, a restart penalty of 2000 and reuse value of 1000 means that a half life period will bring the penalty from 2000 to 1000. Since the half life is 30, it will take 30 seconds. Reload the router and run the command **show interface dampening** a few times when the router first comes up.

```
Rack1R5#show int damp
FastEthernet0/0
  Flaps Penalty    Supp ReuseTm   HalfL  ReuseV   SuppV  MaxSTm    MaxP
Restart
      1    1289    TRUE      11      30    1000    2000      30    2000
2000
Rack1R5#

Rack1R5#show int damp
FastEthernet0/0
  Flaps Penalty    Supp ReuseTm   HalfL  ReuseV   SuppV  MaxSTm    MaxP
Restart
      1    1046    TRUE       2      30    1000    2000      30    2000
2000

%IFDAMP-5-UPDOWN: interface interface is %ssuppressed upda
te FastEthernet0/0 state to IP Routing, interface is UPsuppressed
Rack1R5#
```

Task 8.1

R3:
```
interface Serial1/0
 frame-relay traffic-shaping
!
interface Serial1/0.34
 frame-relay interface-dlci 304
  class DLCI_304
!
interface Serial1/0.35
 frame-relay interface-dlci 305
  class DLCI_305
!
map-class frame-relay DLCI_304
 frame-relay cir 256000
 frame-relay bc 2560
 frame-relay fragment 320
!
map-class frame-relay DLCI_305
 frame-relay cir 256000
 frame-relay bc 2560
 frame-relay fragment 320
```

R4:
```
interface Serial0/0/0
 frame-relay traffic-shaping
 frame-relay interface-dlci 403
  class DLCI_403
!
map-class frame-relay DLCI_403
 frame-relay cir 256000
 frame-relay bc 2560
 frame-relay fragment 320
```

```
R5:
interface Serial0/0
 frame-relay traffic-shaping
 frame-relay interface-dlci 503
  class DLCI_503
!
map-class frame-relay DLCI_503
 frame-relay cir 256000
 frame-relay bc 2560
 frame-relay fragment 320
```

Task 8.1 Verification

Verify Frame-Relay traffic-shaping configuration:

```
Rack1R3#show traffic-shape

Interface   Se1/0
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
302           56000     875    7000      0         125       875       -
301           56000     875    7000      0         125       875       -

Interface   Se1/0.34
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
304           256000    320    2560      0         10        320       -

Interface   Se1/0.35
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
305           256000    320    2560      0         10        320       -
```

Task 8.2

```
R3:
class-map match-all VoIP
  match access-group name VoIP
!
policy-map LLQ
  class VoIP
   priority 200
!
ip access-list extended VoIP
 permit udp any any range 16384 32767
!
map-class frame-relay DLCI_304
 frame-relay mincir 256000
 service-policy output LLQ

R4:
class-map match-all VoIP
  match access-group name VoIP
!
policy-map LLQ
  class VoIP
   priority 200
!
ip access-list extended VoIP
```

```
 permit udp any any range 16384 32767
!
map-class frame-relay DLCI_403
 frame-relay mincir 256000
 service-policy output LLQ
```

Note

The MQC uses the **`mincir`** value in the Frame Relay map-class to determine the available bandwidth on a VC. Since MINCIR defaults to half of the configured CIR, it may be required to adjust the MINCIR value higher if the reserved bandwidth exceeds half of the configured CIR, regardless of whether adaptive shaping is enabled.

Task 8.2 Verification

Verify the policy-map for LLQ configuration:

```
Rack1R3#show policy-map interface
 Serial1/0.34: DLCI 304 -

  Service-policy output: LLQ

    Class-map: VoIP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name VoIP
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 200 (kbps) Burst 5000 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: class-default (match-any)
      1210 packets, 120910 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

 Rack1R4#show policy-map int
 Serial0/0/0: DLCI 403 -

  Service-policy output: LLQ

    queue stats for all priority classes:

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0

    Class-map: VoIP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name VoIP
      Priority: 200 kbps, burst bytes 5000, b/w exceed drops: 0


    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
Rack1R4#
```