Task 1.1

SW1:
```
interface FastEthernet0/22
 switchport voice vlan dot1p
```

Task 1.1 Verification

```
Rack1SW1#show interfaces fa0/22 switchport | include Voice
Voice VLAN: dot1p
```

Task 1.2

R4:
```
interface Serial0/1/0
 ip address negotiated
 encapsulation ppp
 clockrate 64000
 no shutdown
```

R5:
```
interface Serial0/1/0
 encapsulation ppp
 peer default ip address dhcp
 clockrate 64000
 no shutdown
!
ip address-pool dhcp-proxy-client
ip dhcp-server 139.1.11.100
```

Tasks 1.2 & 7.3 Verification

This task should be verified in conjunction with task 7.3. Apply Task 7.3 solution in order to perform complete verification. The preferred option at this point of the lab would be to temporarily hardcode R4's IP address. Then, after full IP reachability has been obtained, R4's IP address can be learned dynamically. If you use this option, be sure to write down what workaround you have put in place so that later in the lab you will be sure to come back to solve the task correctly.

Enable debugging:

```
Rack1R4#debug ppp negotiation
PPP protocol negotiation debugging is on

Rack1R5#debug dhcp
DHCP client activity debugging is on

Rack1R1#debug ip dhcp server events

Rack1R4(config)#interface s0/1/0
```



```
Rack1R4(config-if)#shutdown
Rack1R4(config-if)#no shutdown

Se0/1/0 PPP: Using default call direction
Se0/1/0 PPP: Treating connection as a dedicated line
Se0/1/0 PPP: Session handle[3E000009] Session id[6]
Se0/1/0 PPP: Phase is ESTABLISHING, Active Open
Se0/1/0 LCP: O CONFREQ [Closed] id 6 len 10
Se0/1/0 LCP:    MagicNumber 0x30A1E593 (0x050630A1E593)
Se0/1/0 LCP: I CONFREQ [REQsent] id 6 len 10
Se0/1/0 LCP:    MagicNumber 0x07F9584E (0x050607F9584E)
Se0/1/0 LCP: O CONFACK [REQsent] id 6 len 10
Se0/1/0 LCP:    MagicNumber 0x07F9584E (0x050607F9584E)
Se0/1/0 LCP: I CONFACK [ACKsent] id 6 len 10
Se0/1/0 LCP:    MagicNumber 0x30A1E593 (0x050630A1E593)
Se0/1/0 LCP: State is Open
Se0/1/0 PPP: Phase is FORWARDING, Attempting Forward
Se0/1/0 PPP: Phase is ESTABLISHING, Finish LCP
Se0/1/0 PPP: Phase is UP
Se0/1/0 IPCP: O CONFREQ [Closed] id 1 len 10
Se0/1/0 IPCP:    Address 0.0.0.0 (0x030600000000)
Se0/1/0 CDPCP: O CONFREQ [Closed] id 1 len 4
Se0/1/0 PPP: Process pending ncp packets
Se0/1/0 IPCP: I CONFREQ [REQsent] id 1 len 10
Se0/1/0 IPCP:    Address 139.1.45.5 (0x03068B012D05)
Se0/1/0 IPCP: O CONFACK [REQsent] id 1 len 10
Se0/1/0 IPCP:    Address 139.1.45.5 (0x03068B012D05)
Se0/1/0 CDPCP: I CONFREQ [REQsent] id 1 len 4
Se0/1/0 CDPCP: O CONFACK [REQsent] id 1 len 4
Se0/1/0 CDPCP: I CONFACK [ACKsent] id 1 len 4
Se0/1/0 CDPCP: State is Open
Se0/1/0 IPCP: I CONFREQ [ACKsent] id 2 len 10
Se0/1/0 IPCP:    Address 139.1.45.5 (0x03068B012D05)
Se0/1/0 IPCP: O CONFACK [ACKsent] id 2 len 10
Se0/1/0 IPCP:    Address 139.1.45.5 (0x03068B012D05)
Se0/1/0 IPCP: TIMEout: State ACKsent
Se0/1/0 IPCP: O CONFREQ [ACKsent] id 2 len 10
Se0/1/0 IPCP:    Address 0.0.0.0 (0x030600000000)
Se0/1/0 IPCP: I CONFNAK [ACKsent] id 1 len 10
Se0/1/0 IPCP:    Address 139.1.45.4 (0x03068B012D04)
Se0/1/0 IPCP: ID 1 didn't match 2, discarding packet
Se0/1/0 IPCP: I CONFNAK [ACKsent] id 2 len 10
Se0/1/0 IPCP:    Address 139.1.45.4 (0x03068B012D04)
Se0/1/0 IPCP: O CONFREQ [ACKsent] id 3 len 10
Se0/1/0 IPCP:    Address 139.1.45.4 (0x03068B012D04)
Se0/1/0 IPCP: I CONFACK [ACKsent] id 3 len 10
Se0/1/0 IPCP:    Address 139.1.45.4 (0x03068B012D04)
Se0/1/0 IPCP: State is Open
Se0/1/0 IPCP: Install negotiated IP interface address 139.1.45.4
Se0/1/0 IPCP: Install route to 139.1.45.5
Se0/1/0 IPCP: Add link info for cef entry 139.1.45.5

Rack1R4#show ip interface s0/1/0
Serial0/1/0 is up, line protocol is up
  Internet address is 139.1.45.4/32
  Broadcast address is 255.255.255.255
```



```
  Address determined by IPCP
  Peer address is 139.1.45.5
<output omitted>

Rack1R5#
DHCP: proxy allocate request
DHCP: new entry. add to queue, interface
DHCP: SDiscover attempt # 1 for entry:
DHCP: SDiscover: sending 292 byte length DHCP packet
DHCP: SDiscover 292 bytes
DHCP: XID MATCH in dhcpc_for_us()
DHCP: Received a BOOTREP pkt
DHCP: offer received from 139.1.15.1
DHCP: SRequest attempt # 1 for entry:
DHCP: SRequest- Server ID option: 139.1.15.1
DHCP: SRequest- Requested IP addr option: 139.1.45.4
DHCP: SRequest placed lease len option: 86400
DHCP: SRequest: 310 bytes
DHCP: SRequest: 310 bytes
DHCP: SRequest attempt # 2 for entry:
DHCP: SRequest- Server ID option: 139.1.15.1
DHCP: SRequest- Requested IP addr option: 139.1.45.4
DHCP: SRequest placed lease len option: 86400
DHCP: SRequest: 310 bytes
DHCP: SRequest: 310 bytes
DHCP: XID MATCH in dhcpc_for_us()
DHCP: Received a BOOTREP pkt
DHCP Proxy Client Pooling: ***Allocated IP address: 139.1.45.4

Rack1R1#
DHCPD: assigned IP address 139.1.45.4 to client
0063.6973.636f.2d31.3339.2e31.2e34.352e.352d.5365.7269.616c.302f.31.

Rack1R1#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address          Client-ID/              Lease expiration
Type
                    Hardware address/
                    User name
139.1.45.4          0063.6973.636f.2d31.    Mar 02 1993 01:24 AM
Automatic
                    3339.2e31.2e34.352e.
                    352d.5365.7269.616c.
                    302f.31
```

Task 2.1

R3:

```
key chain RIP
 key 1
  key-string CISCO
!
interface FastEthernet0/1
 ip rip authentication mode md5
 ip rip authentication key-chain RIP
```



```
!
router rip
 version 2
 network 192.10.1.0
```

Task 2.1 Verification

Verify RIP configuration:

```
Rack1R3#show ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    FastEthernet0/1           2     2                    RIP
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    192.10.1.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.10.1.254         120      00:00:09
  Distance: (default is 120)
```

Verify RIP routes:

```
Rack1R3#show ip route rip
R    222.22.2.0/24 [120/7] via 192.10.1.254, 00:00:06, FastEthernet0/1
R    220.20.3.0/24 [120/7] via 192.10.1.254, 00:00:06, FastEthernet0/1
R    205.90.31.0/24 [120/7] via 192.10.1.254, 00:00:06, FastEthernet0/1
```

Task 2.2

R4:

```
router rip
 version 2
 no validate-update-source
 redistribute connected metric 1 route-map CONNECTED_TO_RIP
 network 139.1.0.0
 network 150.1.0.0
 no auto-summary
!
route-map CONNECTED_TO_RIP permit 10
 match interface FastEthernet0/0
```

R5:

```
router rip
 version 2
 network 139.1.0.0
 network 150.1.0.0
 no auto-summary
```


SW2:
```
ip routing
!
router rip
 version 2
 network 139.1.0.0
 network 150.1.0.0
 no auto-summary
```

Task 2.2 Breakdown

On R4, the redistribution will allow the Fa0/0 network to be advertised into RIP. Using a network statement with the passive interface command would still accept updates on that interface, which would break the section requirements. Due to the negotiated PPP connection being seen as a /32 locally, the addition of the "no validate-update-source" will prevent the error shown below:

```
RIP: ignored v2 update from bad source 139.1.45.5 on Serial0/1/0
```

Task 2.2 Verification

```
Rack1R4#show ip route rip
     139.1.0.0/16 is variably subnetted, 8 subnets, 2 masks
R       139.1.15.0/24 [120/1] via 139.1.45.5, 00:00:24
R       139.1.5.0/24 [120/1] via 139.1.45.5, 00:00:24
R       139.1.25.0/24 [120/1] via 139.1.45.5, 00:00:24
R       139.1.45.0/24 [120/2] via 139.1.48.8, 00:00:28, FastEthernet0/1
R       139.1.58.0/24 [120/1] via 139.1.48.8, 00:00:28, FastEthernet0/1
                      [120/1] via 139.1.45.5, 00:00:24
     150.1.0.0/24 is subnetted, 3 subnets
R       150.1.5.0 [120/1] via 139.1.45.5, 00:00:24
R       150.1.8.0 [120/1] via 139.1.48.8, 00:00:28, FastEthernet0/1

Rack1R5#show ip route rip
R    204.12.1.0/24 [120/1] via 139.1.45.4, 00:00:28, Serial0/1/0
     139.1.0.0/16 is variably subnetted, 7 subnets, 2 masks
R       139.1.48.0/24 [120/1] via 139.1.58.8, 00:00:20, FastEthernet0/1
                      [120/1] via 139.1.45.4, 00:00:28, Serial0/1/0
     150.1.0.0/24 is subnetted, 3 subnets
R       150.1.4.0 [120/1] via 139.1.45.4, 00:00:28, Serial0/1/0
R       150.1.8.0 [120/1] via 139.1.58.8, 00:00:20, FastEthernet0/1
```

Task 2.3

R4:
```
router rip
 offset-list 0 in 1 Serial0/1/0
```

R5:
```
router rip
```



```
 default-information originate
!
ip route 0.0.0.0 0.0.0.0 null0
```

Task 2.3 Breakdown

RIP goes by hop count for path selection. The routes learned via SW2 will have a hop count that is one higher. By incrementing the routes learned via the serial link, both paths will have the same metric. With RIP, offset list 0 will match all routes without creating an access list.

Task 2.3 Verification

Verify the RIP routes on R4 before the offset-list has been applied:

```
Rack1R4#show ip route rip
     139.1.0.0/16 is variably subnetted, 8 subnets, 2 masks
R       139.1.15.0/24 [120/1] via 139.1.45.5, 00:00:26
R       139.1.5.0/24 [120/1] via 139.1.45.5, 00:00:26
R       139.1.25.0/24 [120/1] via 139.1.45.5, 00:00:26
R       139.1.45.0/24 [120/2] via 139.1.48.8, 00:00:19, FastEthernet0/1
R       139.1.58.0/24 [120/1] via 139.1.48.8, 00:00:19, FastEthernet0/1
                      [120/1] via 139.1.45.5, 00:00:26
     150.1.0.0/24 is subnetted, 3 subnets
R       150.1.5.0 [120/1] via 139.1.45.5, 00:00:26
R       150.1.8.0 [120/1] via 139.1.48.8, 00:00:19, FastEthernet0/1
R*   0.0.0.0/0 [120/1] via 139.1.45.5, 00:00:26
```

Apply offset list and verify the routes again:

```
Rack1R4#show ip route rip
     139.1.0.0/16 is variably subnetted, 8 subnets, 2 masks
R       139.1.15.0/24 [120/2] via 139.1.48.8, 00:00:15, FastEthernet0/1
                      [120/2] via 139.1.45.5, 00:00:26
R       139.1.5.0/24 [120/2] via 139.1.48.8, 00:00:15, FastEthernet0/1
                     [120/2] via 139.1.45.5, 00:00:26
R       139.1.25.0/24 [120/2] via 139.1.48.8, 00:00:15, FastEthernet0/1
                      [120/2] via 139.1.45.5, 00:00:26
R       139.1.45.0/24 [120/2] via 139.1.48.8, 00:00:15, FastEthernet0/1
R       139.1.58.0/24 [120/1] via 139.1.48.8, 00:00:15, FastEthernet0/1
     150.1.0.0/24 is subnetted, 3 subnets
R       150.1.5.0 [120/2] via 139.1.48.8, 00:00:15, FastEthernet0/1
                  [120/2] via 139.1.45.5, 00:00:26
R       150.1.8.0 [120/1] via 139.1.48.8, 00:00:15, FastEthernet0/1
R*   0.0.0.0/0 [120/2] via 139.1.48.8, 00:00:15, FastEthernet0/1
               [120/2] via 139.1.45.5, 00:00:26
```


Task 2.4

R4, R5, and SW2:

```
router rip
 timers basic 3 18 18 24
```

Task 2.4 Breakdown

RIP convergence time is dependent on the update and flush timers. The lower the flush timer is, the sooner the route will be removed out of the table if an update has not been received about it. Under normal circumstances, the age of a prefix will be reset every update timer. In this case, the flush time for the prefix should never be reached. When an update is not received, it is typically due to a lost routing path. In this case, the route is cleared out of the table when the age reaches the flush.

To change these timers, issue the **`timers basic`** RIP process subcommand. The default RIP timers are hello 30, invalid 180, hold down 180, and flush 240. To view these timer values, issue the **`show ip protocols`** command.

Note: Newer IOS versions also have a configuration option for a sleep timer, but there is not a fixed default value configured.

Task 2.4 Verification

Before and after configuration, check timers with show ip protocols.

```
Rack1SW2# show ip protocols | include Sending|Invalid
ROUTING PROTOCOL IS "RIP"
  SENDING UPDATES EVERY 30 SECONDS, NEXT DUE IN 27 SECONDS
  INVALID AFTER 180 SECONDS, HOLD DOWN 180, FLUSHED AFTER 240

Rack1SW2#show ip protocols | include Sending|Invalid
  Sending updates every 3 seconds, next due in 1 seconds
  Invalid after 18 seconds, hold down 18, flushed after 24
```

Task 2.5

R2:

```
router ospf 1
 area 0 range 139.1.0.0 255.255.240.0
```


Task 2.5 Breakdown

By advertising a summary, R2 will be the less preferred path, since R5 will have a more specific route via R1. If the connection to R1 fails, the summary will be the route used, since R5 will no longer have a more specific route.

Task 2.5 Verification

```
Rack1R5#show ip route ospf
     139.1.0.0/16 is variably subnetted, 15 subnets, 3 masks
O IA    139.1.11.0/24 [110/65] via 139.1.15.1, 00:02:49, Serial0/0.501
O IA    139.1.13.0/24 [110/128] via 139.1.15.1, 00:02:49, Serial0/0.501
O IA    139.1.2.0/24 [110/910] via 139.1.15.1, 00:02:49, Serial0/0.501
O IA    139.1.0.0/24 [110/129] via 139.1.15.1, 00:02:49, Serial0/0.501
O IA    139.1.0.0/20 [110/65] via 139.1.25.2, 00:02:49, Serial0/0.502
O IA    139.1.6.0/24 [110/130] via 139.1.15.1, 00:02:49, Serial0/0.501
O IA    139.1.7.0/24 [110/130] via 139.1.15.1, 00:02:49, Serial0/0.501
O IA    139.1.23.0/24 [110/128] via 139.1.25.2, 00:02:49, Serial0/0.502
     150.1.0.0/16 is variably subnetted, 8 subnets, 2 masks
O IA    150.1.7.7/32 [110/130] via 139.1.25.2, 00:02:49, Serial0/0.502
                     [110/130] via 139.1.15.1, 00:02:49, Serial0/0.501
O IA    150.1.6.6/32 [110/130] via 139.1.25.2, 00:02:49, Serial0/0.502
                     [110/130] via 139.1.15.1, 00:02:49, Serial0/0.501
O IA    150.1.3.3/32 [110/129] via 139.1.25.2, 00:02:50, Serial0/0.502
                     [110/129] via 139.1.15.1, 00:02:50, Serial0/0.501
O       150.1.2.2/32 [110/65] via 139.1.25.2, 00:02:50, Serial0/0.502
O       150.1.1.1/32 [110/65] via 139.1.15.1, 00:02:50, Serial0/0.501
```

Check the backup path:

```
Rack1R5(config)#interface s0/0.501
Rack1R5(config-subif)#shutdown
%OSPF-5-ADJCHG: Process 1, Nbr 150.1.1.1 on Serial0/0.501 from FULL to
DOWN, Neighbor Down: Interface down or detached

Rack1R5(config-subif)#do sh ip route ospf
     139.1.0.0/16 is variably subnetted, 8 subnets, 3 masks
O IA    139.1.0.0/20 [110/65] via 139.1.25.2, 00:05:15, Serial0/0.502
O IA    139.1.23.0/24 [110/128] via 139.1.25.2, 00:05:15, Serial0/0.502
     150.1.0.0/16 is variably subnetted, 7 subnets, 2 masks
O IA    150.1.7.7/32 [110/130] via 139.1.25.2, 00:05:15, Serial0/0.502
O IA    150.1.6.6/32 [110/130] via 139.1.25.2, 00:05:15, Serial0/0.502
O IA    150.1.3.3/32 [110/129] via 139.1.25.2, 00:05:15, Serial0/0.502
O       150.1.2.2/32 [110/65] via 139.1.25.2, 00:05:15, Serial0/0.502
```

Task 2.6

R3:

```
router ospf 1
 redistribute rip subnets
!
router rip
redistribute ospf 1 metric 1
```



```
auto-summary

R5:
router ospf 1
 redistribute rip subnets
```

Task 2.6 Breakdown

With RIP, auto-summarization is on by default, and will summarize to classful boundaries. If you disabled it during earlier RIP configuration, you can disable it for this step, so that R3 only sends the necessary routes. Since it is the default, "auto-summary" will not show up in the configuration under the RIP process.

Task 2.6 Verification

Verify that R3 sends the minimum required routing information to BB2:

```
Rack1R3#debug ip rip
RIP protocol debugging is on
Rack1R3#
RIP: sending v2 update to 224.0.0.9 via FastEthernet0/1 (192.10.1.3)
RIP: build update entries
   139.1.0.0/16 via 0.0.0.0, metric 1, tag 0
   150.1.0.0/16 via 0.0.0.0, metric 1, tag 0
   204.12.1.0/24 via 0.0.0.0, metric 1, tag 0
```

Finally, to ensure you have full internal connectivity run the following TCL script:

```
foreach i {
139.1.2.2
139.1.25.2
150.1.2.2
139.1.23.2
139.1.13.3
139.1.0.3
150.1.3.3
139.1.23.3
192.10.1.3
150.1.4.4
139.1.45.4
139.1.48.4
139.1.15.5
139.1.5.5
139.1.25.5
150.1.5.5
139.1.45.5
139.1.58.5
139.1.6.6
139.1.0.6
150.1.6.6
139.1.7.7
139.1.0.7
```



```
150.1.7.7
150.1.8.8
139.1.48.8
139.1.58.8
139.1.11.254
139.1.2.22

} { ping $i  }
```

Note that the Frame Relay link between R6 and BB1 is omitted from connectivity test.

Task 2.7

R4:

```
router bgp 100
 network 139.1.5.0 mask 255.255.255.0
 aggregate-address 139.1.0.0 255.255.0.0 summary-only
 neighbor 204.12.1.254 unsuppress-map UNSUPPRESS
 distribute-list prefix DENY_AGGREGATE in
!
ip prefix-list DENY_AGGREGATE seq 5 deny 139.1.0.0/16
ip prefix-list DENY_AGGREGATE seq 10 permit 0.0.0.0/0 le 32
!
ip prefix-list VLAN_5 seq 5 permit 139.1.5.0/24
!
route-map UNSUPPRESS permit 10
 match ip address prefix-list VLAN_5
```

R6:

```
router bgp 100
 network 139.1.6.0 mask 255.255.255.0
 aggregate-address 139.1.0.0 255.255.0.0 summary-only
```

Task 2.7 Verification

Check routes that R4 and R6 advertise to BB3:

```
Rack1R4#show ip bgp neighbors 204.12.1.254 advertised-routes
BGP table version is 15, local router ID is 150.1.4.4
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
r> 139.1.0.0        0.0.0.0                            32768 i
s> 139.1.5.0/24     139.1.45.5               2         32768 ?

Rack1R6#show ip bgp neighbors 54.1.2.254 advertised-routes
BGP table version is 14, local router ID is 150.1.6.6
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
*> 139.1.0.0        0.0.0.0                              32768 i
```

Task 2.7 Breakdown

Start by adding a network to BGP and then configuring a summary on R4 and R6. In order for the more specific route for VLAN 5 to be sent, an unsuppress map is used along with the summary-only keyword on the aggregate, so that the more specific route is unsuppressed before sending to the backbone.

Additionally, if you are sending prefixes out to the backbones at multiple locations, you may want to consider filtering routes inbound, so that you do not learn the same route from another location. Normally, you would probably consider configuring filtering inbound on both R4 and R6, to prevent advertisements from looping back into the topology. Part of the next section includes filtering some routes. Since the filtering for the next section overlaps the routes, filtering is just done on R4 for this task, since R6 will be filtered separately in the next step.

Task 2.8

R4:

```
router rip
 redistribute bgp 100 metric 1 route-map PERMIT_ODD
!
router bgp 100
 bgp router-id 150.1.5.5
 neighbor 204.12.1.254 route-map PERMIT_ODD in
!
ip access-list standard ODD
 permit 1.0.0.0 254.255.255.255
!
route-map PERMIT_ODD permit 10
 match ip address ODD
```

R5:

```
router rip
 redistribute ospf 1 metric 1 route-map OSPF_TO_RIP
!
route-map OSPF_TO_RIP permit 10
 match tag 6
```

R6:

```
router ospf 1
 redistribute bgp 100 subnets tag 6 route-map PERMIT_EVEN
!
router bgp 100
 neighbor 54.1.2.254 route-map PERMIT_EVEN in
!
ip access-list standard EVEN
```



```
 permit 0.0.0.0 254.255.255.255
!
route-map PERMIT_EVEN permit 10
 match ip address EVEN
```

Task 2.8 Breakdown

The BGP synchronization rule states that all iBGP learned routes must have a match in the IGP table in order to be considered for BGP best path selection. Although the BGP synchronization rule is rarely enabled in a production BGP environment, and is effectively considered legacy now, the problem that it was designed to prevent is still valid.

BGP synchronization is designed to prevent the case when non BGP speaking devices are in the transit path of the iBGP network. Since these transit devices are not running BGP, they must have an IGP route in order to send traffic to the final destination. Therefore, the BGP synchronization process first checks the IGP table to see if there is a match for all iBGP learned prefixes. If there are equal IGP matches in the IP routing table, synchronization has occurred, and the iBGP learned prefix can be considered for best path selection. However, if there is no matching IGP prefix for the iBGP prefix, synchronization has not occurred, and the iBGP learned prefix cannot be considered for best path selection.

In the above scenario, BGP synchronization is enabled on R4. Therefore any iBGP learned prefixes on R4 must have matching IGP routes in order to be considered valid. Therefore, BGP prefixes must be injected into the IGP domain in order for this case to occur.

There is an additional issue with OSPF. When you turn synchronization on, and redistribute BGP prefixes into OSPF, you should make sure that OSPF ASBR Router ID matches originating BGP Router ID. This is why we set Router ID of R4 to 150.1.5.5.

Task 2.8 Verification

Verify that R4 accepts only odd first octet prefixes from BB3:

```
Rack1R4#show ip bgp neighbors 204.12.1.254 routes
BGP table version is 21, local router ID is 150.1.4.4
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
*> 113.0.0.0        204.12.1.254                           0 54 50 60 i
*> 115.0.0.0        204.12.1.254                           0 54 i
*> 117.0.0.0        204.12.1.254                           0 54 i
```

```
*> 119.0.0.0        204.12.1.254                           0 54 i
```

Confirm that R6 accepts only prefixes with even first octet from BB1:

```
Rack1R6#show ip bgp neighbors 54.1.2.254 routes
BGP table version is 18, local router ID is 150.1.6.6
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
*> 28.119.16.0/24   54.1.2.254                             0 54 i
*> 28.119.17.0/24   54.1.2.254                             0 54 i
*> 112.0.0.0        54.1.2.254                0            0 54 50 60 i
*> 114.0.0.0        54.1.2.254                0            0 54 i
*> 116.0.0.0        54.1.2.254                0            0 54 i
*> 118.0.0.0        54.1.2.254                0            0 54 i
```

Next, verify the BGP redistribution:

```
Rack1R4#show ip route rip
R    118.0.0.0/8 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
                 [120/2] via 139.1.45.5, 00:00:00
R    116.0.0.0/8 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
                 [120/2] via 139.1.45.5, 00:00:00
     139.1.0.0/16 is variably subnetted, 8 subnets, 2 masks
R       139.1.15.0/24 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
                      [120/2] via 139.1.45.5, 00:00:00
R       139.1.5.0/24 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
                     [120/2] via 139.1.45.5, 00:00:00
R       139.1.25.0/24 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
                      [120/2] via 139.1.45.5, 00:00:00
R       139.1.45.0/24 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
R       139.1.58.0/24 [120/1] via 139.1.48.8, 00:00:01, FastEthernet0/1
R    114.0.0.0/8 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
                 [120/2] via 139.1.45.5, 00:00:00
R    112.0.0.0/8 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
                 [120/2] via 139.1.45.5, 00:00:00
     28.0.0.0/24 is subnetted, 2 subnets
R       28.119.17.0 [120/2] via 139.1.48.8, 00:00:02, FastEthernet0/1
                    [120/2] via 139.1.45.5, 00:00:01
R       28.119.16.0 [120/2] via 139.1.48.8, 00:00:02, FastEthernet0/1
                    [120/2] via 139.1.45.5, 00:00:01
     150.1.0.0/24 is subnetted, 3 subnets
R       150.1.5.0 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
                  [120/2] via 139.1.45.5, 00:00:00
R       150.1.8.0 [120/1] via 139.1.48.8, 00:00:01, FastEthernet0/1
R*   0.0.0.0/0 [120/2] via 139.1.48.8, 00:00:01, FastEthernet0/1
               [120/2] via 139.1.45.5, 00:00:00

Rack1R6#show ip route ospf | include E2
O E2 119.0.0.0/8 [110/20] via 139.1.0.3, 00:04:58, FastEthernet0/0
O E2 222.22.2.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2 204.12.1.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2 117.0.0.0/8 [110/20] via 139.1.0.3, 00:04:58, FastEthernet0/0
```



```
O E2 220.20.3.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2    139.1.5.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2    139.1.45.4/32 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2    139.1.45.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2    139.1.58.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2    139.1.48.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2 115.0.0.0/8 [110/20] via 139.1.0.3, 00:04:58, FastEthernet0/0
O E2 113.0.0.0/8 [110/20] via 139.1.0.3, 00:04:58, FastEthernet0/0
O E2 192.10.1.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2    150.1.5.0/24 [110/20] via 139.1.0.3, 00:05:01, FastEthernet0/0
O E2    150.1.4.0/24 [110/20] via 139.1.0.3, 00:05:03, FastEthernet0/0
O E2    150.1.8.0/24 [110/20] via 139.1.0.3, 00:05:03, FastEthernet0/0
O E2 205.90.31.0/24 [110/20] via 139.1.0.3, 00:05:03, FastEthernet0/0
```

Verify BGP synchronization:

```
Rack1R6#show ip bgp 115.0.0.0
BGP routing table entry for 115.0.0.0/8, version 22
Paths: (1 available, best #1, table Default-IP-Routing-Table, RIB-
failure(17))
  Advertised to update-groups:
     2
  54
    150.1.4.4 (metric 20) from 150.1.4.4 (150.1.5.5)
      Origin IGP, metric 0, localpref 100, valid, internal,
synchronized, best
```

```
Rack1R4#show ip bgp 116.0.0.0
BGP routing table entry for 116.0.0.0/8, version 16
Paths: (1 available, best #1, table Default-IP-Routing-Table, RIB-
failure(17))
  Advertised to update-groups:
     1
  54
    150.1.6.6 (metric 2) from 150.1.6.6 (150.1.6.6)
      Origin IGP, metric 0, localpref 100, valid, internal,
synchronized, best
```

Make a final verification by tracerouting to even numbered routes from R4 and odd from R6:

```
Rack1R4#traceroute 116.0.0.1

Type escape sequence to abort.
Tracing the route to 116.0.0.1

  1 139.1.48.8 4 msec
    139.1.45.5 16 msec
    139.1.48.8 8 msec
  2 139.1.25.2 28 msec
    139.1.58.5 12 msec
    139.1.25.2 32 msec
  3 139.1.25.2 24 msec
    139.1.23.3 44 msec
    139.1.25.2 28 msec
```



```
  4 139.1.0.6 44 msec
    139.1.23.3 36 msec
    139.1.0.6 40 msec
  5 139.1.0.6 40 msec
    54.1.2.254 60 msec
    139.1.0.6 40 msec


Rack1R6#traceroute 115.0.0.1

Type escape sequence to abort.
Tracing the route to 115.0.0.1

  1 139.1.0.3 4 msec 0 msec 0 msec
  2 139.1.23.2 16 msec 16 msec 12 msec
  3 139.1.25.5 32 msec 32 msec 28 msec
  4 139.1.45.4 44 msec 40 msec 44 msec
  5 204.12.1.254 44 msec 44 msec 44 msec
  6 172.16.4.1 36 msec *  32 msec
```

Task 2.9

R4:

```
router bgp 100
 neighbor 204.12.1.254 maximum-prefix 150000 90
```

R6:

```
router bgp 100
 neighbor 54.1.2.254 maximum-prefix 150000 90
```

Task 2.9 Breakdown

Large fluctuations in the BGP table can cause devices with limited amounts of memory to crash. These fluctuations usually occur either due to a misconfiguration, or a malicious attack on the BGP table. In order to prevent such a fluctuation from occurring, the **maximum-prefix** option on the BGP neighbor statement can be used to configure a threshold of received routes at which a BGP session will be reset.

Task 2.9 Verification

```
Rack1R6#show ip bgp neighbors 54.1.2.254 | begin Maximum prefixes
  Maximum prefixes allowed 150000
  Threshold for warning message 90%
  Number of NLRIs in the update sent: max 3, min 0
<output omitted>

Rack1R4#show ip bgp neighbors 204.12.1.254 | begin Maximum prefixes
  Maximum prefixes allowed 150000
  Threshold for warning message 90%
  Number of NLRIs in the update sent: max 0, min 0
```



```
<output omitted>
```

Task 3.1

R2:

```
interface FastEthernet0/0
 ipv6 ospf 1 area 1
!
interface Serial0/1
 ipv6 ospf 1 area 0

ipv6 router ospf 1
area 1 range 2001:CC1E:1:0::/62
```

R3:

```
interface FastEthernet0/0
 ipv6 ospf 1 area 0
!
!
interface Serial1/3
 ipv6 ospf 1 area 0
```

R6:

```
interface FastEthernet0/0
 ipv6 ospf 1 area 1
!
interface FastEthernet0/1
 ipv6 ospf 1 area 0

ipv6 router ospf 1
area 1 range 2001:CC1E:1:4::/62
```

Task 3.1 Verification

Configuring a summary will prevent R2 and R6 from seeing the original routes for each other's Fa0/0 interfaces. Verify the routes on R6, R3 and R2:

```
Rack1R2#show ipv6 route ospf
IPv6 Routing Table - 9 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:CC1E:1::/62 [110/0]
     via ::, Null0
O   2001:CC1E:1::/64 [110/65]
     via FE80::3, Serial0/1
OI  2001:CC1E:1:4::/62 [110/66]
```



```
     via FE80::3, Serial0/1
Rack1R2#

Rack1R3#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
OI  2001:CC1E:1::/62 [110/782]
     via FE80::2, Serial1/3
OI  2001:CC1E:1:4::/62 [110/2]
     via FE80::6, FastEthernet0/0
Rack1R3#

Rack1R6#show ipv6 route ospf
IPv6 Routing Table - Default - 8 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, M - MIPv6, R - RIP, I1 - ISIS L1
       I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary, D - EIGRP
       EX - EIGRP external
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OI  2001:CC1E:1::/62 [110/783]
     via FE80::3, FastEthernet0/1
O   2001:CC1E:1:4::/62 [110/0]
     via Null0, directly connected
O   2001:CC1E:1:23::2/127 [110/782]
     via FE80::3, FastEthernet0/1
Rack1R6#
```

Task 3.2

R6:

```
interface FastEthernet0/0
 ipv6 address 2001:CC1E:1:6::/64 eui-64
 ipv6 nd ra-interval 60
 ipv6 nd ra-lifetime 180
```

Task 3.2 Verification

Verify IPv6 ND RA configuration:

```
Rack1R6#show ipv6 interface FastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::215:62FF:FED0:4831
  Global unicast address(es):
    2001:CC1E:1:6:215:62FF:FED0:4831, subnet is 2001:CC1E:1:6::/64
[EUI]
```



```
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::9
    FF02::1:FFD0:4831
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 60 seconds
  ND router advertisements live for 180 seconds
  Hosts use stateless autoconfig for addresses.
```

Task 5.1

R3:

```
interface Tunnel35
 ip unnumbered FastEthernet0/0
 ip pim dense-mode
 tunnel source Loopback0
 tunnel destination 150.1.5.5
```

R5:

```
interface Tunnel35
 ip unnumbered FastEthernet0/0
 ip pim dense-mode
 tunnel source Loopback0
 tunnel destination 150.1.3.3
!
ip mroute 0.0.0.0 0.0.0.0 Tunnel35
```

Task 5.1 Breakdown

The above scenario uses a GRE tunnel to tunnel multicast traffic across non-PIM speaking neighbors. As the tunnel interface is based on the loopback interfaces of R3 and R5, R1 (the non-PIM speaking device) only sees unicast GRE traffic between these loopback interfaces. Therefore, as long as the transit devices have unicast reachability throughout the network, they can be used to transport multicast traffic.

Task 5.1 Verification

Join multicast groups 239.2.2.2 with R2 FastEthernet0/0 and 239.5.5.5 with R5 FastEthernet 0/0:

R2:
```
interface FastEthernet0/0
 ip igmp join-group 239.2.2.2
```

R5:
```
interface FastEthernet0/0
 ip igmp join-group 239.5.5.5
```

Enable mpacket debugging at R3:

```
Rack1R3#debug ip mpacket
IP multicast packets debugging is on
```

Simulate multicast traffic from R6 to 239.2.2.2, add the Fa0/1 interface on R6 as a PIM dense mode interface to test.

```
Rack1R6#ping 239.2.2.2 repeat 6

Type escape sequence to abort.
Sending 6, 100-byte ICMP Echos to 239.2.2.2, timeout is 2 seconds:

Reply to request 0 from 139.1.23.2, 32 ms
Reply to request 1 from 139.1.23.2, 32 ms
Reply to request 2 from 139.1.23.2, 32 ms
Reply to request 3 from 139.1.23.2, 32 ms
Reply to request 4 from 139.1.23.2, 32 ms
Reply to request 5 from 139.1.23.2, 36 ms
```

Look at R3's debugging output:

```
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.2.2.2 (Serial1/3) id=22,
ttl=254, prot=1, len=100(100), mforward
Rack1R3#
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.2.2.2 (Serial1/3) id=23,
ttl=254, prot=1, len=100(100), mforward
Rack1R3#
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.2.2.2 (Serial1/3) id=24,
ttl=254, prot=1, len=100(100), mforward
Rack1R3#
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.2.2.2 (Serial1/3) id=25,
ttl=254, prot=1, len=100(100), mforward
Rack1R3#
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.2.2.2 (Serial1/3) id=26,
ttl=254, prot=1, len=100(100), mforward
Rack1R3#
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.2.2.2 (Serial1/3) id=27,
ttl=254, prot=1, len=100(100), mforward

Rack1R3#show ip mroute
IP Multicast Routing Table
<snip>
```



```
(*, 239.2.2.2), 00:04:59/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel35, Forward/Dense, 00:04:59/00:00:00
    Serial1/3, Forward/Dense, 00:04:59/00:00:00

(139.1.0.6, 239.2.2.2), 00:01:26/00:02:38, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1/3, Forward/Dense, 00:01:27/00:00:00
    Tunnel35, Prune/Dense, 00:01:27/00:01:32

(*, 224.0.1.40), 00:20:35/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel35, Forward/Dense, 00:13:52/00:00:00
    Serial1/3, Forward/Dense, 00:20:35/00:00:00
```

Next, enable additional debugging at R3, and send multicast traffic from R6 to 239.5.5.5:

```
Rack1R6#ping 239.5.5.5 repeat 6

Type escape sequence to abort.
Sending 6, 100-byte ICMP Echos to 239.5.5.5, timeout is 2 seconds:

Reply to request 0 from 139.1.5.5, 68 ms
Reply to request 1 from 139.1.5.5, 68 ms
Reply to request 2 from 139.1.5.5, 80 ms
Reply to request 3 from 139.1.5.5, 68 ms
Reply to request 4 from 139.1.5.5, 68 ms
Reply to request 5 from 139.1.5.5, 88 ms

Rack1R3#debug ip packet detail 100
IP packet debugging is on (detailed) for access list 100
```

Note how GRE traffic is load balanced. There are two debugs running on R3: debug ip mpacket and debug ip packet detail for the GRE traffic.

```
Rack1R3#
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.5.5.5 (Tunnel35) id=46,
ttl=254, prot=1, len=100(100), mforward
IP: s=150.1.3.3 (Tunnel35), d=150.1.5.5 (Serial1/2), len 124, sending,
proto=47
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.5.5.5 (Tunnel35) id=47,
ttl=254, prot=1, len=100(100), mforward
IP: s=150.1.3.3 (Tunnel35), d=150.1.5.5 (Serial1/2), len 124, sending,
proto=47
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.5.5.5 (Tunnel35) id=48,
ttl=254, prot=1, len=100(100), mforward
IP: s=150.1.3.3 (Tunnel35), d=150.1.5.5 (Serial1/3), len 124, sending,
proto=47
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.5.5.5 (Tunnel35) id=49,
ttl=254, prot=1, len=100(100), mforward
```



```
IP: s=150.1.3.3 (Tunnel35), d=150.1.5.5 (Serial1/2), len 124, sending,
proto=47
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.5.5.5 (Tunnel35) id=50,
ttl=254, prot=1, len=100(100), mforward
IP: s=150.1.3.3 (Tunnel35), d=150.1.5.5 (Serial1/3), len 124, sending,
proto=47
IP: s=150.1.3.3 (Tunnel35), d=150.1.5.5 (Serial1/2), len 78, sending,
proto=47
IP(0): s=139.1.0.6 (FastEthernet0/0) d=239.5.5.5 (Tunnel35) id=51,
ttl=254, prot=1, len=100(100), mforward
IP: s=150.1.3.3 (Tunnel35), d=150.1.5.5 (Serial1/3), len 124, sending,
proto=47
```

Task 5.2

R1, R2:

```
ip multicast rpf backoff 10 1000
ip multicast route-limit 100
```

Task 5.2 Breakdown

Here, we are just modifying some miscellaneous settings for R1 and R2. We aren't given a minimum value, so you can pick something arbitrarily for the RPF backoff.

Task 6.1

R3:

```
interface FastEthernet0/1
 ip access-group FILTER_IN in
 ip access-group FILTER_OUT out
 no ip unreachables
!
ip access-list extended FILTER_IN
 deny   icmp any any echo log
 permit ip any any
!
ip access-list extended FILTER_OUT
 deny   icmp any any time-exceeded log
 deny   icmp any any port-unreachable log
 permit ip any any
```


R4:

```
interface FastEthernet0/0
 ip access-group FILTER_IN in
 ip access-group FILTER_OUT out
 no ip unreachables
!
ip access-list extended FILTER_IN
 deny   icmp any any echo log
 permit ip any any
!
ip access-list extended FILTER_OUT
 deny   icmp any any time-exceeded log
 deny   icmp any any port-unreachable log
 permit ip any any
```

Task 6.1 Breakdown

Double check the ACL, and make sure that you have a “permit any” at the end, so that you are not dropping any legitimate traffic. Blocking the ICMP echo traffic will affect ping testing for connectivity. If you are checking connectivity at the end of the lab, make sure to take note of any situations like this where you are specifically asked to block the traffic.

Task 6.2

R5:

```
ip inspect tcp synwait-time 10
ip inspect name INTERCEPT tcp
!
interface FastEthernet 0/0
 ip inspect INTERCEPT out
```

Task 6.2 Verification

```
Rack1R5#show ip inspect all
Session audit trail is disabled
Session alert is enabled
one-minute (sampling period) thresholds are [400:500] connections
max-incomplete sessions thresholds are [400:500]
max-incomplete tcp connections per host is 50. Block-time 0 minute.
tcp synwait-time is 10 sec -- tcp finwait-time is 5 sec
tcp idle-time is 3600 sec -- udp idle-time is 30 sec
dns-timeout is 5 sec
Inspection Rule Configuration
 Inspection name INTERCEPT
    tcp alert is on audit-trail is off timeout 3600

Interface Configuration
 Interface FastEthernet0/0
  Inbound inspection rule is not set
  Outgoing inspection rule is INTERCEPT
```



```
    tcp alert is on audit-trail is off timeout 3600
  Inbound access list is not set
  Outgoing access list is not set
```

Task 6.3

R5:

```
ip inspect max-incomplete low 81
ip inspect max-incomplete high 100
!
ip inspect one-minute low 40
ip inspect one-minute high 60
!
ip inspect tcp max-incomplete host 20 block-time 2
ip inspect tcp finwait-time 2
```

Task 6.3 Verification

```
Rack1R5#show ip inspect config
Session audit trail is disabled
Session alert is enabled
one-minute (sampling period) thresholds are [40:60] connections
max-incomplete sessions thresholds are [81:100]
max-incomplete tcp connections per host is 20. Block-time 2 minutes.
tcp synwait-time is 10 sec -- tcp finwait-time is 2 sec
tcp idle-time is 3600 sec -- udp idle-time is 30 sec
dns-timeout is 5 sec
Inspection Rule Configuration
 Inspection name INTERCEPT
    tcp alert is on audit-trail is off timeout 3600
```

Task 6.3 Breakdown

Watch your thresholds carefully. Thresholds need to be crossed. For the rising thresholds 100 and 60, the wording in the section is exceeds and above. For the one minute falling, the section says below. For the incomplete threshold, the section states “reaches 80”. Since the threshold of 80 would not be crossed until it dropped below 80, setting the threshold to 81 will allow the clamping to stop when that threshold is crossed, and the number of connections falls to 80.

Task 6.4

SW1:

```
ip dhcp snooping vlan 367
ip dhcp snooping
!
interface FastEthernet 0/3
 ip dhcp snooping trust
```

R3:

```
ip dhcp relay information policy keep
int fa0/0
```



```
ip dhcp relay info trust
```

R1:

```
ip dhcp relay information trust-all
```

Task 6.4 Verification

```
Rack1SW1#show ip dhcp snooping
Switch DHCP snooping is enabled
DHCP snooping is configured on following VLANs:
367
Insertion of option 82 is enabled
   circuit-id format: vlan-mod-port
    remote-id format: MAC
Option 82 on untrusted port is not allowed
Verification of hwaddr field is enabled
Interface                    Trusted     Rate limit (pps)
------------------------     -------     ----------------
FastEthernet0/3              yes         unlimited


Rack1R1#show ip dhcp relay info trust
All interfaces are trusted source of relay agent information option
```

Note: With the earlier configuration as shown, the helper address is tied to the active HSRP device. For testing, you can create an access list to filter debugging as shown below:

R3:

```
ip access-list 102 permit udp any any range 67 68

Rack1R3#debug ip packet 102 detail
Rack1R3#debug ip dhcp server
```

First, take a look at the output when R3 is not active. It receives the DHCP request, but does not forward.

```
19:03:39.982: IP: s=0.0.0.0 (FastEthernet0/0), d=255.255.255.255, len
344, rcvd 2
 19:03:39.982:     UDP src=68, dst=67
 19:03:39.982: DHCPD: message is from trusted interface FastEthernet0/0
```

Next, take a look at how the output changes when R3 is active for the HSRP group. For this test, R6's FastEthernet interface has been shut down, and R3 has been given time to take over for the HSRP group.

```
19:35:42.642: IP: s=0.0.0.0 (FastEthernet0/0), d=255.255.255.255, len
362, rcvd 2
 19:35:42.642:     UDP src=68, dst=67
 19:35:42.646: DHCPD: message is from trusted interface FastEthernet0/0
 19:35:42.646: DHCPD: Finding a relay for client
0063.6973.636f.2d30.3031.322e.3030.6630.2e62.3861.302d.4661.302f.30 on
interface FastEthernet0/0.
```



```
 19:35:42.646: DHCPD: setting giaddr to 139.1.0.3.
 19:35:42.650: IP: tableid=0, s=139.1.0.3 (local), d=139.1.13.1
(Serial1/2), routed via FIB
 19:35:42.650: IP: s=139.1.0.3 (local), d=139.1.13.1 (Serial1/2), len
362, sending
 19:35:42.650:     UDP src=67, dst=67
 19:35:42.650: DHCPD: BOOTREQUEST from
0063.6973.636f.2d30.3031.322e.3030.6630.2e62.3861.302d.4661.302f.30
forwarded to 139.1.13.1.
 19:35:42.758: IP: tableid=0, s=139.1.13.1 (Serial1/2), d=139.1.0.3
(FastEthernet0/0), routed via RIB
 19:35:42.758: IP: s=139.1.13.1 (Serial1/2), d=139.1.0.3, len 385, rcvd
4
 19:35:42.758:     UDP src=67, dst=67
 19:35:42.758: DHCPD: forwarding BOOTREPLY to client 0012.00f0.b8a0.
 19:35:42.762: DHCPD: broadcasting BOOTREPLY to client 0012.00f0.b8a0.
 19:35:42.762: IP: s=139.1.0.3 (local), d=255.255.255.255
(FastEthernet0/0), len 385, sending broad/multicast
 19:35:42.762:     UDP src=67, dst=68
Rack1R3#
```

Task 6.5

R5:

```
ip domain-name INE.com
username cisco password cisco
crypto key gen rsa mod 1024

object-group network TELSSH
150.1.1.1 /32
150.1.2.2 /32
150.1.3.3 /32
150.1.4.4 /32
150.1.7.7 /32
150.1.8.8 /32

Access-list 105 permit tcp obj TELSSH any range 22 23
Line vty 0 807
 Access-class 105 in
```

Task 6.5 Verification

Try to telnet from various addresses. Attempting from R6's lo0 should be blocked, as well as from R4 when not sourcing from the loopback0 interface.

```
Rack1R6#telnet 150.1.5.5 /sou lo0
Trying 150.1.5.5 ...
% Connection refused by remote host

Rack1R6#
Rack1R4#telnet 150.1.5.5
Trying 150.1.5.5 ...
```



```
% Connection refused by remote host

Rack1R4#telnet 150.1.5.5 /sou lo0
Trying 150.1.5.5 ... Open


User Access Verification

Username:
Password:
Rack1R5>Rack1SW1#show ip dhcp snooping
```

Task 7.1

R6:

```
snmp-server enable traps bgp
snmp-server host 139.1.2.100 CISCOBGP
```

R3 and R4:

```
logging 139.1.5.100
logging facility local6
```

Task 7.1 Verification

```
Rack1R3#show logging | beg Trap
    Trap logging: level informational, 85 message lines logged
        Logging to 139.1.5.100 (udp port 514, audit disabled, link up),
2 message lines logged, xml disabled,
               filtering disabled
Rack1R3#

Rack1R4#show loggin | beg Trap
    Trap logging: level informational, 86 message lines logged
        Logging to 139.1.5.100  (udp port 514,  audit disabled,
              authentication disabled, encryption disabled, link up),
              2 message lines logged,
              0 message lines rate-limited,
              0 message lines dropped-by-MD,
              xml disabled, sequence number disabled
              filtering disabled
Rack1R4#
```

Task 7.2

R6:

```
interface FastEthernet0/1
 ip nbar protocol-discovery
```


R5:

```
flow monitor TEST
 statistics packet protocol
 statistics packet size
 record netflow ipv4 protocol-port-tos

int fa0/1
ip flow monitor TEST output
ip accounting output-packets
```

Task 7.2 Verification

To see how NBAR collects statistics temporarily enable NBAR on interfaces FastEthernet 0/0:

```
Rack1R6#show ip nbar protocol-discovery interface Fa0/0 top-n 3

 FastEthernet0/0
                        Input                    Output
                        -----                    ------
   Protocol             Packet Count             Packet Count
                        Byte Count               Byte Count
                        5min Bit Rate (bps)      5min Bit Rate (bps)
                        5min Max Bit Rate (bps)  5min Max Bit Rate (bps)
   -------------------- ------------------------ ------------------------
   icmp                 200                      0
                        22800                    0
                        0                        0
                        0                        0
   ospf                 23                       10
                        2298                     1040
                        0                        0
                        0                        0
   bgp                  4                        0
                        266                      0
                        0                        0
                        0                        0
   unknown              0                        0
                        0                        0
                        0                        0
                        0                        0
   Total                227                      10
                        25364                    1040
                        0                        0
0
```

Alternatively, IP accounting and Netflow can also be used to gather traffic statistics, as shown on R5's configuration. Generate some transit traffic to test.

```
Rack1R5#show flow mon TEST statistics
  Cache type:                           Normal
  Cache size:                             4096
  Current entries:                           0
```



```
  High Watermark:                          2

  Flows added:                             3
  Flows aged:                              3
    - Active timeout   (  1800 secs)       0
    - Inactive timeout (    15 secs)       3
    - Event aged                           0
    - Watermark aged                       0
    - Emergency aged                       0

Packet size distribution (869 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416
   .000 .884 .000 .115 .000 .000 .000 .000 .000 .000 .000 .000 .000

    448  480  512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

Protocol         Total    Flows   Packets Bytes Packets Active(Sec) Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt    /Sec       /Flow    /Flow
TCP-Telnet           2      0.0       384    40      0.0       77.1      15.5
ICMP                 1      0.0       100   100      0.0        6.2      15.3
Total:               3      0.0       289    47      0.0       53.5      15.4


Rack1R5#show ip accounting
   Source           Destination             Packets               Bytes
 139.1.15.1       150.1.8.8                   870               40896

Accounting data age is 6
Rack1R5#
```

Task 7.3

```
R1:
ip dhcp excluded-address 139.1.45.0 139.1.45.3
ip dhcp excluded-address 139.1.45.5 139.1.45.255
!
ip dhcp pool R4
  network 139.1.45.0 255.255.255.0
!
ip route 139.1.45.5 255.255.255.255 139.1.15.5

R5:
no ip dhcp-server 139.1.11.100
ip dhcp-server 139.1.15.1
```

☞ Quick Note
Task states that installed server is not valid. Use R1 instead.

Task 7.3 Breakdown

Verification for this task is shown with section 1.2. Make sure to exclude the addresses before defining the address pool.

Task 7.4

R1:

```
ip dhcp excluded-address 139.1.0.0 139.1.0.99
ip dhcp excluded-address 139.1.0.201 139.1.0.255
!
ip dhcp pool VLAN_367
 network 139.1.0.0 255.255.255.0
 default-router 139.1.0.1
 domain-name InternetworkExpert.com
 lease infinite
!
```

R3:

```
!
interface FastEthernet0/0
 standby 1 name HSRP
 ip helper-address 139.1.13.1 redundancy HSRP
 standby 1 ip 139.1.0.1
 standby 1 preempt
```

R6:

```
interface FastEthernet0/1
 standby 1 name HSRP
 ip helper-address 139.1.13.1 redundancy HSRP
 standby 1 ip 139.1.0.1
 standby 1 priority 101
 standby 1 preempt
```

Task 7.4 Verification

Verify the standby configuration:

```
Rack1R6#show standby
FastEthernet0/1 - Group 1
  State is Active
    1 state change, last state change 00:04:38
  Virtual IP address is 139.1.0.1
  Active virtual MAC address is 0000.0c07.ac01
    Local virtual MAC address is 0000.0c07.ac01 (v1 default)
  Hello time 3 sec, hold time 10 sec
    Next hello sent in 0.048 secs
  Preemption enabled
  Active router is local
  Standby router is 139.1.0.3, priority 100 (expires in 8.052 sec)
  Priority 101 (configured 101)
  IP redundancy name is "HSRP" (cfgd)
```

Verify DHCP address assignment and the redundancy configuration:

Use SW2 to simulate a host in VLAN367:

```
Rack1SW2(config)#interface vl367
%LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan367, changed state
to up
```



```
Rack1SW2(config-if)#ip address dhcp
Rack1SW2(config-if)#
DHCP: DHCP client process started: 10
RAC: Starting DHCP discover on Vlan367
DHCP: Try 1 to acquire address for Vlan367
DHCP: allocate request
DHCP: new entry. add to queue
DHCP: SDiscover attempt # 1 for entry:
DHCP: SDiscover: sending 300 byte length DHCP packet
DHCP: SDiscover 300 bytes
            B'cast on Vlan367 interface from 0.0.0.0
DHCP: SDiscover attempt # 2 for entry:
DHCP: SDiscover: sending 300 byte length DHCP packet
DHCP: SDiscover 300 bytes
            B'cast on Vlan367 interface from 0.0.0.0
DHCP: Received a BOOTREP pkt
DHCP: offer received from 139.1.13.1
DHCP: SRequest attempt # 1 for entry:
DHCP: SRequest- Server ID option: 139.1.13.1
DHCP: SRequest- Requested IP addr option: 139.1.0.100
DHCP: SRequest placed lease len option: 4294967295
DHCP: SRequest: 318 bytes
DHCP: SRequest: 318 bytes
            B'cast on Vlan367 interface from 0.0.0.0
DHCP: Received a BOOTREP pkt
DHCP: offer received from 139.1.13.1
DHCP: offer received in bad state: Requesting  punt
DHCP: Received a BOOTREP pkt
DHCP: offer received from 139.1.13.1
DHCP: offer received in bad state: Requesting  punt
DHCP: Received a BOOTREP pkt
DHCP: offer received from 139.1.13.1
DHCP: offer received in bad state: Requesting  punt
DHCP: Received a BOOTREP pkt
Interface Vlan367 assigned DHCP address 139.1.0.100, mask 255.255.255.0

DHCP Client Pooling: ***Allocated IP address: 139.1.0.100
DHCP: Received a BOOTREP pkt
DHCP: rcv ack in Bound state: punt
Allocated IP address = 139.1.0.100  255.255.255.0

Rack1R1#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address          Client-ID/              Lease expiration
Type
                    Hardware address/
                    User name
139.1.0.100         0063.6973.636f.2d30.    Infinite
Automatic
                    3030.662e.3866.6232.
                    2e65.3830.302d.566c.
                    3336.37
139.1.45.4          0063.6973.636f.2d31.    Mar 02 1993 01:24 AM
Automatic
                    3339.2e31.2e34.352e.
                    352d.5365.7269.616c.
```



```
                    302f.31

Rack1R6(config)#interface Fa0/1
Rack1R6(config-if)#shutdown

Rack1R3#show standby
FastEthernet0/0 - Group 1
  State is Active
    5 state changes, last state change 00:00:18
  Virtual IP address is 139.1.0.1
  Active virtual MAC address is 0000.0c07.ac01
    Local virtual MAC address is 0000.0c07.ac01 (v1 default)
  Hello time 3 sec, hold time 10 sec
    Next hello sent in 2.412 secs
  Preemption enabled
  Active router is local
  Standby router is unknown
  Priority 100 (default 100)
  IP redundancy name is "HSRP" (cfgd)

Rack1SW2#ping 139.1.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 139.1.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms
```

Task 7.4 Breakdown

R1 is supposed to hand out addresses for VLAN367, but is not directly connected. R3 and R6 can forward the traffic by using a helper address. By tying the helper address to the HSRP group name with the redundancy keyword, only the active HSRP device will forward the traffic. Configuring HSRP will allow one device to take over for the other and act as the gateway. Make sure to have R1 configured with the HSRP address as the gateway. The section also states to not rely on client specific methods. If that was not a restriction, two methods that could be used would be specifying multiple addresses for the default router option on the DHCP scope or IRDP. With IRDP, the end devices need to be IRDP-aware. With multiple default routers specified, the clients need to determine that the first one is unreachable and decide to use the next one.

Task 7.5

SW1 and SW2:

```
logging file flash:log.txt informational
```

Task 7.5 Verification

```
Rack1SW2#show logging
```

```
Syslog logging: enabled (0 messages dropped, 3 messages rate-limited, 0
flushes, 0 overruns, xml disabled, filtering disabled)
    Console logging: level debugging, 58 messages logged, xml disabled,
                     filtering disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging: level debugging, 60 messages logged, xml disabled,
                    filtering disabled
    Exception Logging: size (4096 bytes)
    Count and timestamp logging messages: disabled
    File logging: file flash:log.txt,
        max size 0, min size 0,
        level informational, 1 messages logged
    Trap logging: level informational, 63 message lines logged
<output omitted>
```

Task 8.1

R2:

```
access-list 101 permit udp any any
access-list 102 permit tcp any any
!
 class-map match-all ICMP
  match protocol icmp
!
 class-map match-all UDP
  match access-group 101
!
class-map match-all TCP
  match access-group 102
!
 policy-map MQC_CAR
  class ICMP
   drop
  class UDP
   police cir 128000 bc 2000
     conform-action transmit
     exceed-action set-prec-transmit 0
  class TCP
   police cir 256000 bc 4000
     conform-action transmit
     exceed-action set-prec-transmit 0
!
interface FastEthernet0/0
 service-policy input MQC_CAR
```

Task 8.1 Verification

Verify the policy map application on the interface. For ICMP, you can match with the “match protocol ICMP” rather than by using an access list. Since both the

conform action and exceed actions are both drop, you can use the MQC 'drop' keyword for the traffic in that class.

```
Rack1R2#show policy-map int fa0/0
 FastEthernet0/0

  Service-policy input: MQC_CAR

    Class-map: ICMP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol icmp
      drop

    Class-map: UDP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 101
      police:
          cir 128000 bps, bc 2000 bytes
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          set-prec-transmit 0
        conformed 0 bps, exceed 0 bps

    Class-map: TCP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 102
      police:
          cir 256000 bps, bc 4000 bytes
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          set-prec-transmit 0
        conformed 0 bps, exceed 0 bps

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
Rack1R2#
```

Task 8.2

R5:

```
class-map match-all HTTP_RESPONSES
match access-group name HTTP_RESPONSES
!
!
policy-map DLCI_501
class HTTP_RESPONSES
bandwidth percent 80
```



```
!
interface Serial0/0/0
 bandwidth 384
 bandwidth inherit
 frame-relay traffic-shaping
!
interface Serial0/0/0.501 point-to-point
 frame-relay class DLCI_501
!
ip access-list extended HTTP_RESPONSES
permit tcp any eq www 443 139.1.11.0 0.0.0.255
!
map-class frame-relay DLCI_501
 frame-relay cir 384000
 frame-relay mincir 384000
 service-policy output DLCI_501
```

Task 8.2 Breakdown

This is a fairly straightforward configuration, using a MQC policy for frame traffic shaping. The "bandwidth inherit" command will pass configured bandwidth values to a subinterface to match what is configured on the primary interface. If you manually configure a bandwidth value on the subinterface, it will override the inherited value.

Task 8.2 Verification

Watch your ACL creation carefully, we are specifically told to watch for HTTP replies. Verify the policy configuration:

```
Rack1R5#show frame-relay pvc 501

PVC Statistics for interface Serial0/0/0 (Frame Relay DTE)

DLCI = 501, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0.501

  input pkts 2353          output pkts 5770         in bytes 213730
  out bytes 1786756        dropped pkts 7           in pkts dropped 7
  out pkts dropped 0                 out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 5504      out bcast bytes 1727736
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 1000 bits/sec, 0 packets/sec
  pvc create time 03:40:46, last time pvc status changed 03:40:46
  cir 384000    bc 384000    be 0         byte limit 6000   interval
125
  mincir 384000    byte increment 6000  Adaptive Shaping none
  pkts 112       bytes 41576     pkts delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  service policy DLCI_501
```

```
 Serial0/0/0.501: DLCI 501 -

  Service-policy output: DLCI_501

    Class-map: HTTP_RESPONSES (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name HTTP_RESPONSES
      Queueing
        Output Queue: Conversation 41
        Bandwidth 80 (%)
        Bandwidth 307 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      109 packets, 40580 bytes
      5 minute offered rate 1000 bps, drop rate 0 bps
      Match: any
  Output queue size 0/max total 600/drops 0
```

Task 8.3

R1:

```
map-class frame-relay DLCI_105
 frame-relay cir 512000
 frame-relay bc 5120
 frame-relay fragment 640
!
interface Serial0/0
 frame-relay traffic-shaping
 frame-relay class DLCI_105
```

R5:

```
Interface Serial0/0/0
 Bandwidth 512
interface Serial0/0/0.502 point-to-point
 frame-relay class DLCI_502
!
map-class frame-relay DLCI_501
 frame-relay cir 512000
 frame-relay bc 5120
 frame-relay fragment 640
!
map-class frame-relay DLCI_502
 frame-relay cir 512000
 frame-relay mincir 128000
```

Task 8.3 Breakdown

Here we have some additional configuration between R5 and R1. In the earlier step, we were just given the CIR for the circuit, but not given the port speed. Here, we have the additional information for the port. By setting bc to 1% of the

cir, we are configuring an interval of 10ms. By default, enabling traffic shaping will set circuits to a rate of 56k. In order to have DLCI 502 not be adversely affected, a basic class can be configured for that DLCI.

Task 8.3 Verification

Verify the Frame-Relay PVC shaping parameters:

```
Rack1R5#show frame-relay pvc 501 | begin fragment type
fragment type end-to-end fragment size 640
  cir 512000    bc   5120      be 0         limit 640    interval 10
  mincir 384000    byte increment 640   BECN response no  IF_CONG no
  frags 261       bytes 97278     frags delayed 0       bytes delayed 0
  shaping inactive
  traffic shaping drops 0


Rack1R5#show frame-relay pvc 502 | begin cir
cir 512000    bc 512000    be 0      byte limit 8000   interval 125
  mincir 128000    byte increment 8000  Adaptive Shaping none
  pkts 577       bytes 223590    pkts delayed 2       bytes delayed 166
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: fifo
  Output queue 0/40, 0 drop, 0 dequeued

Rack1R1#show frame-relay pvc 105 | begin fragment type
fragment type end-to-end fragment size 640
  cir 512000    bc   5120      be 0         limit 640    interval 10
  mincir 256000    byte increment 640   BECN response no  IF_CONG no
  frags 56        bytes 5070      frags delayed 0       bytes delayed 0
  shaping inactive
  traffic shaping drops 0
```

Task 8.4

R3:

```
interface FastEthernet0/0
 ip policy route-map POLICY_ROUTING
!
ip access-list extended FROM_VLAN_367_TO_VLAN_43
 permit ip 139.1.0.0 0.0.0.255 204.12.1.0 0.0.0.255
!
route-map POLICY_ROUTING permit 10
 match ip address FROM_VLAN_367_TO_VLAN_43
 match length 1251 1500
 set ip next-hop 139.1.23.2
```

R5:

```
interface FastEthernet0/1
 ip policy route-map POLICY_ROUTING
!
interface Serial0/1/0
```



```
 ip policy route-map POLICY_ROUTING
!
ip access-list extended FROM_VLAN_43_TO_VLAN_367
 permit ip 204.12.1.0 0.0.0.255 139.1.0.0 0.0.0.255
!
route-map POLICY_ROUTING permit 10
 match ip address FROM_VLAN_43_TO_VLAN_367
 match length 1251 1500
 set ip next-hop 139.1.25.2
```

Task 8.4 Verification

Generate packets of different sizes from R6 to BB3 and then enable policy route debugging at R3:

```
Rack1R3#debug ip policy
Policy routing debugging is on
Rack1R3#

Rack1R6#ping 204.12.1.254

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.12.1.254, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 88/91/92 ms

IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 100, FIB policy
match
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 100, FIB policy
rejected(deny) - normal forwarding
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 100, FIB policy
match
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 100, FIB policy
rejected(deny) - normal forwarding
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 100, FIB policy
match
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 100, FIB policy
rejected(deny) - normal forwarding

Rack1R6#ping 204.12.1.254 size 1300

Type escape sequence to abort.
Sending 5, 1300-byte ICMP Echos to 204.12.1.254, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
1008/1018/1060 ms

Rack1R3#
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 1300, FIB policy
match
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, g=139.1.23.2, len
1300, FIB policy routed
Rack1R3#
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 1300, FIB policy
match
```



```
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, g=139.1.23.2, len
1300, FIB policy routed
Rack1R3#
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 1300, FIB policy
match
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, g=139.1.23.2, len
1300, FIB policy routed
Rack1R3#
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 1300, FIB policy
match
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, g=139.1.23.2, len
1300, FIB policy routed
Rack1R3#
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, len 1300, FIB policy
match
IP: s=139.1.0.6 (FastEthernet0/0), d=204.12.1.254, g=139.1.23.2, len
1300, FIB policy routed
```

You can also check the output of show route-map on R3 and R5 and verify matches:

```
Rack1R5#show route-map
route-map POLICY_ROUTING, permit, sequence 10
  Match clauses:
    ip address (access-lists): FROM_VLAN_43_TO_VLAN_367
    length 1251 1500
  Set clauses:
    ip next-hop 139.1.25.2
  Policy routing matches: 300 packets, 379500 bytes
```

Task 8.5

R5:

```
map-class frame-relay DLCI_502
 frame-relay cir 512000
 frame-relay bc 5120
 frame-relay fragment 640
 frame-relay ip rtp priority 16384 16383 512
```

R2:

```
interface Serial0/0
 frame-relay traffic-shaping
 frame-relay class DLCI_205
!
map-class frame-relay DLCI_205
 frame-relay cir 512000
 frame-relay bc 5120
 frame-relay fragment 640
 frame-relay ip rtp priority 16384 16383 512
```

Task 8.5 Verification

Verify the VoIP QoS configuration:

```
Rack1R5#show frame-relay pvc 502 | include Queueing|fragment|rtp
```

```
Queueing strategy: weighted fair
  fragment type end-to-end fragment size 640
  ip rtp priority parameters 16384 32767 512000

Rack1R2#show frame-relay pvc 205| include Queueing|fragment|rtp
Queueing strategy: weighted fair
 fragment type end-to-end fragment size 640
 ip rtp priority parameters 16384 32767 512000
```

Task 8.6

Find SW2's MAC address:

```
Rack1SW2#show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  139.1.48.8               -   0019.55cb.c341  ARPA
FastEthernet0/20
```

R4:

```
class-map SW2
 match destination mac 0019.55cb.c341

policy-map SWOUT
 class SW2
  set precedence 7

interface fastEthernet 0/1
service-policy output SWOUT
```


Task 8.6 Verification

Verify by pinging through from BB3. By matching on the destination MAC, traffic to other hosts on VLAN 24 will not be affected.

```
Rack1R4#show policy-map int fa0/1
 FastEthernet0/1

  Service-policy output: SWOUT

    Class-map: SW2 (match-all)
      106 packets, 12030 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: destination-address mac 0019.55CB.C341
      QoS Set
        precedence 7
          Packets marked 105

    Class-map: class-default (match-any)
      523 packets, 120825 bytes
      5 minute offered rate 1000 bps, drop rate 0 bps
      Match: any
Rack1R4#
```

