Task 1.1

R6:

```
bridge irb
!
interface FastEthernet0/0.16
 bridge-group 1
!
interface FastEthernet0/0.36
 bridge-group 1
!
interface BVI1
 ip address 136.1.136.6 255.255.255.0
!
bridge 1 protocol ieee
bridge 1 route ip
```

Task 1.1 Breakdown

By default, Cisco routers will route IP and bridge all other protocols on all interfaces. Additionally, a protocol can be either routed or bridged, but not both. By using either the concurrent routing and bridging (CRB) or integrated routing and bridging (IRB) features, this limitation can be overcome.

With CRB, a protocol can be routed on one interface while being bridged on another interface. When CRB is used, traffic in the routed domain cannot be passed on to the bridge domain. With IRB, a protocol can be both routed and bridged on the same interface. Therefore traffic from the routed domain *can* be passed on to the bridge domain.

These features are useful when you want to extend the broadcast domain for one protocol, while maintaining it for another. For example, IPX can be bridged between two LAN segments, while IP is routed on those interfaces (CRB). Additionally, a bridge virtual interface (BVI) can be configured with an IPX address so that other segments running IPX routing can communicate with the IPX bridged network (IRB). CRB is considered a legacy feature since IRB inherits all functionality of CRB, with the addition of the BVI.

In the above example, two LAN segments running IP need to be bridged together. The first step in bridging is to create a transparent bridge group. This is accomplished by issuing the global configuration command **bridge [*num*] protocol ieee**. The *ieee* option specifies that IEEE spanning-tree will be enabled for the bridge group. To apply the bridge-group, use the interface command **bridge-group [*num*]**, where *num* is the bridge group previously created.

Since *ip routing* is enabled by default, the above configuration will only enable transparent bridging for non-IP protocols. To enable the integrated routing and bridging process, use the global configuration command **bridge irb**. Next, choose which protocols you want to route and bridge for the bridge group. This is accomplished by issuing the **bridge [*num*] route [*protocol*]**. In the above

case, IP is both routed and bridged for bridge group 1. Lastly, the BVI is created by issuing the **interface bvi [*num*]**, where *num* is the bridge group number. All traffic that passes from the bridge domain to the routed domain and vice versa must pass through the BVI. This is the interface where logical configuration is placed, such as an IP address.

Task 1.1 Verification

Verify the IRB configuration on R6:

```
Rack1R6#show interface irb | begin FastEthernet0/0
FastEthernet0/0

 Not bridging this sub-interface.

FastEthernet0/0.16

 Routed protocols on FastEthernet0/0.16:
  ip

 Bridged protocols on FastEthernet0/0.16:
  appletalk  clns       decnet     ip

 <output omitted>

FastEthernet0/0.36

 Routed protocols on FastEthernet0/0.36:
  ip

 Bridged protocols on FastEthernet0/0.36:
  appletalk  clns       decnet     ip

<output omitted>

Rack1R6#ping 136.1.136.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 136.1.136.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Rack1R6#ping 136.1.136.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 136.1.136.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Rack1R3#ping 136.1.136.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 136.1.136.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms
```


Task 1.2

SW1:

```
spanning-tree vlan 4,44,52,63 root primary
!
interface FastEthernet0/14
 spanning-tree vlan 4,44,52,63 port-priority 32
!
interface FastEthernet0/15
 spanning-tree vlan 4,44,52,63 port-priority 16
```

Task 1.2 Breakdown

Spanning-tree protocol is used to ensure one loop free path throughout the bridge domain. This single loop free path is a top down tree in which the source of the tree is the *root bridge*. Root bridge election is determined by the bridge-ID. Bridge-ID is a made up of a priority value along with a single burned in MAC address that the switch possesses. The bridge with the lowest bridge-ID will be elected the root bridge. To influence root bridge election, change the bridge's priority by issuing the **spanning-tree vlan [*num*] priority [*priority*]** command. The **spanning-tree vlan [*num*] root [primary | secondary]** command is a macro that automatically sets the bridge priority to an appropriate value.

Note

By default, Cisco switches run per-vlan spanning-tree protocol (PVST+) in which each VLAN runs a separate instance of spanning-tree. Therefore, there is one root bridge election per VLAN.

Once the root bridge election has occurred, each bridge must decide on a single path it will use to get to the root bridge. The outgoing port used to reach the root bridge is known as the *root port*. There are four variables that affect the root port selection. These are: cost, bridge-ID, port priority, and port-id in that order.

Cost is cumulative throughout the STP domain, and is the sum of all port costs in the path. Port cost is based on a non-linear inverse representation of the bandwidth of the interface (higher bandwidth equals lower cost). Lower total cost is better.

Caution

Each switch's priority defaults to half of the maximum value. This typically results in a tie in priority between all bridges in the spanning-tree domain (some switches such as the 3550 and 3560 offset the priority value with a *system-id-extension*). The tie breaker for the root election is the lower MAC address. This implies that older switches have the tendency to be elected root. When

designing a switch block, be sure to carefully influence the root bridge election. Otherwise, all traffic will be forced to transit the older and most likely lower performing bridges due to spanning-tree.

Bridge-ID priority is the same for the 3550s and 3560s as previously discussed.

Port priority is a value from 1-255, and defaults to half (128). Lower port priority is also better, but priority is only locally significant between two directly connected bridges.

The final tie breaker in the root port election is port ID. Port ID is based on the physical port number (ie Fa0/1 = port 1), and lower is better.

To influence which port is elected the root port, the two user configurable values to change are port cost and port priority. Changing port cost will affect both the local bridge and all downstream bridges. Changing port priority will only affect the directly connected downstream bridge. Keep in mind that port priority is only taken into account if there is a tie in both cost and bridge-ID (a tie in bridge-ID implies that a bridge has multiple connections to the same upstream bridge).

For this task, port-priority is changed on the root bridge (SW1) in order to influence how the downstream bridge (SW2) elects its root port.

Task 1.2 Verification

```
Rack1SW2#show spanning-tree vlan 44

VLAN0044
  Spanning tree enabled protocol ieee
  Root ID     Priority    24592
              Address     0019.55e6.6580
              Cost        19 ← cost to root
              Port        17 (FastEthernet0/15) ← root port
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32812  (priority 32768 sys-id-ext 44)
              Address     0016.9d31.8380
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------
Fa0/13           Altn BLK 19  ← tie  128.15   P2p
Fa0/14           Altn BLK 19  ← in   128.16   P2p
Fa0/15           Root FWD 19  ← cost 128.17   P2p
                   ↑
                 root port

Rack1SW2#show spanning-tree vlan 44 detail
```

```
Port 15 (FastEthernet0/13) of VLAN0044 is blocking
 <output deleted>
   Designated port id is 128.15, designated path cost 0
<output deleted>          ↑
                 upstream port priority

Port 16 (FastEthernet0/14) of VLAN0044 is blocking
<output deleted>
   Designated port id is 32.16, designated path cost 0
<output deleted>          ↑
                 upstream port priority

Port 17 (FastEthernet0/15) of VLAN0044 is forwarding
<output deleted>
   Designated port id is 16.17, designated path cost 0
<output deleted>          ↑
                 upstream port priority
                     lowest wins
```

Task 1.3

SW2:

```
spanning-tree uplinkfast
```

Task 1.3 Breakdown

Spanning-tree uplinkfast provides fast reconvergence in the event of a direct failure of the root port. During the initial root port election, a bridge running uplinkfast notes which ports can be used as alternate paths to the root bridge. When the root port fails, the alternate port immediately comes out of blocking state and transitions to forwarding. Also, to ensure convergence of the upstream CAM table, all known MAC addresses are flooded out the new root port as dummy multicast frames. This process typically takes three to five seconds, and reduces convergence time considerably. Uplinkfast is only supported when running PVST+. To configure uplinkfast, use the global configuration command **spanning-tree uplinkfast**.

Task 1.3 Verification

Verify that UplinkFast is enabled:

```
Rack1SW2#show spanning-tree uplinkfast
UplinkFast is enabled

Station update rate set to 150 packets/sec.

UplinkFast statistics
-----------------------
Number of transitions via uplinkFast (all VLANs)            : 0
Number of proxy multicast addresses transmitted (all VLANs) : 0

Name                 Interface List
```

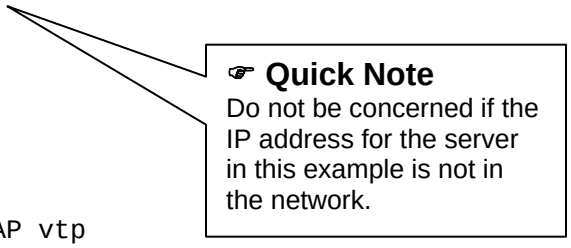


```
-------------------- ------------------------------------
VLAN0001             Fa0/13(fwd), Fa0/14, Fa0/15
<output omitted>
VLAN0063             Fa0/15(fwd), Fa0/13, Fa0/14
```

Task 1.4

SW1 and SW2:

```
access-list 50 permit 136.1.2.100
!
snmp-server community CISCORO RO 50
snmp-server community CISCORW RW 50
snmp-server location San Jose, CA US
snmp-server contact CCIE Lab SW1
snmp-server chassis-id 221-787878
snmp-server enable traps vtp
snmp-server host 136.1.2.100 CISCOTRAP vtp
```



☞ **Quick Note**
Do not be concerned if the IP address for the server in this example is not in the network.

Task 1.4 Verification

Verify that SNMP is configured correctly:

```
Rack1SW1#show snmp
Chassis: 221-787878
Contact: CCIE Lab SW1
Location: San Jose, CA US
<output omitted>
SNMP logging: enabled
    Logging to 136.1.2.100.162, 0/10, 0 sent, 0 dropped.
SNMP agent enabled
```

Task 2.1

R1:

```
router ospf 1
 network 150.1.1.1 0.0.0.0 area 0
```

R2:

```
interface Serial0/0
 ip ospf network point-to-multipoint
!
router ospf 1
 network 150.1.2.2 0.0.0.0 area 0
```

R4:

```
interface Serial0/0/0
 ip ospf network point-to-multipoint
!
router ospf 1
 network 150.1.4.4 0.0.0.0 area 0
```

R5:

```
interface Serial0/0/0.245 multipoint
 ip ospf network point-to-multipoint
!
```



```
router ospf 1
 network 150.1.5.5 0.0.0.0 area 0
```

R1, R2

```
!
! R4 and R5 Loopbacks should appear as /32 for MPLS VPN peering
!
interface Loopback0
 ip ospf network point-to-point
```

Task 2.1 Verification

Verify the basic OSPF configuration and network types:

```
Rack1R5#show ip protocols
Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 150.1.5.5
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    136.1.245.5 0.0.0.0 area 0
  Routing Information Sources:
    Gateway         Distance      Last Update
    150.1.4.4            110      00:00:53
    150.1.2.2            110      00:00:53
    150.1.5.5            110      00:00:53
  Distance: (default is 110)

Rack1R5#show ip ospf neighbor

Neighbor ID Pri   State     Dead Time   Address         Interface
150.1.4.4     0   FULL/  -  00:01:40    136.1.245.4     Serial0/0/0.245
150.1.2.2     0   FULL/  -  00:01:56    136.1.245.2     Serial0/0/0.245

Rack1R5#show ip ospf interface
Serial0/0/0.245 is up, line protocol is up
  Internet Address 136.1.245.5/24, Area 0
  Process ID 1, Router ID 150.1.5.5, Network Type POINT_TO_MULTIPOINT,
Cost: 64
<output omitted>
    Adjacent with neighbor 150.1.4.4
    Adjacent with neighbor 150.1.2.2
```

Verify that R2 could reach R4 via R5, by the virtue of /32 route:

```
Rack1R2#show ip route ospf
     136.1.0.0/16 is variably subnetted, 6 subnets, 2 masks
O       136.1.245.4/32 [110/128] via 136.1.245.5, 00:04:56, Serial0/0
O       136.1.245.5/32 [110/64] via 136.1.245.5, 00:04:56, Serial0/0

Rack1R2#ping 136.1.245.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 136.1.245.4, timeout is 2 seconds:
!!!!!
```



```
Success rate is 100 percent (5/5), round-trip min/avg/max = 88/90/96 ms

Verify Loopback advertisements:

Rack1R5#show ip ospf interface loopback 0
Loopback0 is up, line protocol is up
  Internet Address 150.1.5.5/24, Area 0
  Process ID 1, Router ID 150.1.5.5,Network Type POINT_TO_POINT,Cost: 1
<output omitted>

Rack1R5#show ip route ospf | inc 150.1.
     150.1.0.0/24 is subnetted, 4 subnets
O       150.1.4.0 [110/65] via 136.1.245.4, 00:00:48, Serial0/0.245
O       150.1.2.0 [110/65] via 136.1.245.2, 00:00:48, Serial0/0.245
O       150.1.1.0 [110/65] via 136.1.15.1, 00:00:48, Serial0/0.15
```

Task 2.2

```
R4:
interface Serial0/1/0
 ip ospf cost 65534
!
router ospf 1
 area 45 virtual-link 150.1.5.5
 network 136.1.45.4 0.0.0.0 area 45

R5:
interface Serial0/1/0
 ip ospf cost 65534
!
router ospf 1
 area 45 virtual-link 150.1.4.4
 network 136.1.45.5 0.0.0.0 area 45
```

Task 2.2 Breakdown

When R4 loses its connection to the Frame Relay cloud, OSPF areas 4 and 44 lose their connection to area 0. Additionally, since R4's Loopback 0 interface is advertised into OSPF area 0, area 0 becomes discontiguous when the Frame Relay connection of R4 is down. Frame Relay RFC 2328 dictates that OSPF area 0 must be contiguous throughout the OSPF domain. In addition to this requirement, all other areas must be connected to area 0. For situations where physical connectivity cannot be obtained, a *virtual-link* can be used as a logical connection to area 0.

Virtual-links can be used to repair broken connections to area 0, connect two discontiguous areas to area 0, and connect discontiguous area 0s. To configure a virtual-link, use the routing process subcommand **area [*transit_area*] virtual-link [*ABR_router-ID*]**, where *transit_area* is the area the virtual-link will transit and *ABR_router-ID* is the router-ID of the area border router on the other side of the link.

In this particular case, we manually configured OSPF cost on the interface. Alternatively, we could have configured the interface for a lower bandwidth value, to make it less preferred.

Caution

A virtual-link *is* an interface in area 0. Therefore, all attributes of area 0 are inherited by routers attached to the virtual-link. This includes area 0 authentication and stipulations on area summarization. Remember that a router that terminates a virtual-link is an area 0 router.

Task 2.2 Verification

```
Rack1R5#show ip ospf interface s0/1/0
Serial0/1/0 is up, line protocol is up
  Internet Address 136.1.45.5/24, Area 45
  Process ID 1, Router ID 150.1.5.5, Network Type POINT_TO_POINT, Cost:
65534
  <output omitted>

Rack1R5#show ip ospf neighbor

Neighbor ID  Pri   State       Dead Time   Address         Interface
150.1.4.4      0   FULL/  -       -        136.1.45.4      OSPF_VL0
<output omitted>
150.1.4.4      0   FULL/  -    00:00:37    136.1.45.4      Serial0/1

Rack1R5#show ip ospf virtual-links
Virtual Link OSPF_VL0 to router 150.1.4.4 is up
  Run as demand circuit
  DoNotAge LSA allowed.
  Transit area 45, via interface Serial0/1, Cost of using 65534
```

Task 2.3

R1:

```
interface Serial0/0
 ip ospf message-digest-key 1 md5 CISCO
!
router ospf 1
 area 0 authentication message-digest
```

R2:

```
interface Serial0/0
 ip ospf message-digest-key 1 md5 CISCO
!
router ospf 1
 area 0 authentication message-digest
```

R4:

```
interface Serial0/0/0
 ip ospf message-digest-key 1 md5 CISCO
```



```
!
router ospf 1
 area 0 authentication message-digest
 area 45 virtual-link 150.1.5.5 message-digest-key 1 md5 CISCO

R5:
interface Serial0/0/0.15 point-to-point
 ip ospf message-digest-key 1 md5 CISCO
!
interface Serial0/0/0.245 multipoint
 ip ospf message-digest-key 1 md5 CISCO
!
router ospf 1
 area 0 authentication message-digest
 area 45 virtual-link 150.1.4.4 message-digest-key 1 md5 CISCO
```

Task 2.3 Breakdown

OSPF supports both clear text and MD5 authentication. Both of these authentication types can be applied to an OSPF area as a whole, or on an individual interface basis. When area authentication is enabled, all adjacencies in the area must be authenticated with the defined authentication type. In the above case, MD5 authentication is enabled in area 0. This implies that all area 0 adjacencies must authenticate using MD5, unless otherwise overridden.

Pitfall

A virtual-link is an area 0 adjacency. If authentication is required for all OSPF area 0 adjacencies, then it must also be configured on all virtual-links.

To enable OSPF area authentication, issue the routing process subcommand **area 0 authentication [message-digest]**. Adding the *message-digest* keyword indicates MD5 authentication. Without this command, authentication will be clear-text. Next, specify the authentication key on the interface with either the **ip ospf authentication-key** or the **ip ospf message-digest-key** depending on whether clear-text or MD5 authentication is enabled. To authenticate a virtual-link, add the keyword **authentication-key** or **message-digest-key** to the virtual-link statement. Authentication keys are locally significant to an interface, and therefore may differ on a per interface basis.

Note

Interface level authentication overrides area authentication. Therefore adjacencies within an area may be configured for clear-text authentication while a specific interface in the area is configured for MD5 authentication or NULL (no) authentication. To enable interface authentication, issue the interface level

command **ip ospf authentication** or **ip ospf authentication message-digest**. Note that interface level authentication is also available on a virtual-link.

Task 2.3 Verification

```
Rack1R5#show ip ospf
 Routing Process "ospf 1" with ID 150.1.5.5
<output omitted>
    Area BACKBONE(0)
        Number of interfaces in this area is 4
        Area has message digest authentication
<output omitted>

Rack1R5#show ip ospf virtual-links
Virtual Link OSPF_VL0 to router 150.1.4.4 is up
<output omitted>
  Message digest authentication enabled
    Youngest key id is 1
```

Verify that we still have OSPF neighbors:

```
Rack1R5#show ip ospf neighbor

Neighbor ID  Pri   State       Dead Time  Address       Interface
150.1.4.4      0   FULL/  -       -       136.1.45.4    OSPF_VL0
150.1.1.1      0   FULL/  -    00:00:30   136.1.15.1    Serial0/0/0.15
150.1.4.4      0   FULL/  -    00:01:34   136.1.245.4   Serial0/0/0.245
150.1.2.2      0   FULL/  -    00:01:51   136.1.245.2   Serial0/0/0.245
150.1.4.4      0   FULL/  -    00:00:39   136.1.45.4    Serial0/1/0
```

Double check to see if we are receiving authenticated packets. There are three authentication possibilities in the debug output, 0 is no authentication, 1 is plaintext, and 2 is MD5. (Note that 150.1.4.4 packets are not authenticated):

```
Rack1R5#debug ip ospf packet
*OSPF: rcv. v:2 t:1 l:48 rid:150.1.4.4
      aid:0.0.0.45 chk:B761 aut:0 auk: from Serial0/1/0
*OSPF: rcv. v:2 t:1 l:48 rid:150.1.1.1
      aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x2B91730E from
Serial0/0/0.15
*OSPF: rcv. v:2 t:1 l:48 rid:150.1.2.2
      aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x2B917303 from
Serial0/0/0.245
```

Task 2.4

R1, R2, R4, and R5:

```
router ospf 1
 auto-cost reference-bandwidth 20000
```

Task 2.4 Breakdown

OSPF cost calculation is based on the following formula:

$$COST = \frac{Reference\_Bandwitdh}{Interface\_Bandwidth}$$

Reference_Bandwidth defaults to 100Mbps and *Interface_Bandwidth* is the configured **bandwidth** statement of an interface. If *Interface_Bandwidth* is greater than *Reference_Bandwidth,* the resulting cost value is 1. Since the reference bandwidth defaults to 100Mbps, this implies that all interfaces with a bandwidth above 100Mbps (OC3, GigE, etc) will have a cost of 1. In networks with high speed links, this calculation may result in suboptimal forwarding. To adjust how OSPF calculates its cost, change the reference bandwidth value by using the routing process subcommand **auto-cost reference-bandwidth [*Reference_Mbps*]**.

Task 2.4 Verification

Verify that the costs have been recalculated:

```
Rack1R4#show interfaces FastEthernet0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdP2, address is 0030.947e.e581 (bia 0030.947e.e581)
  Internet address is 136.1.4.4/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
<output omitted>

Rack1R4#show ip ospf interface Fa0/0
FastEthernet0/0 is up, line protocol is up
  Internet Address 136.1.4.4/24, Area 4
  Process ID 1, Router ID 150.1.4.4, Network Type BROADCAST, Cost: 2000
<output omitted>
```

And for serial interface:

```
Rack1R4#show interfaces s0/0
Serial0/0/0 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 136.1.245.4/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
```



```
Rack1R4#show ip ospf interface s0/0
Serial0/0/0 is up, line protocol is up
  Internet Address 136.1.245.4/24, Area 0
  Process ID 1, Router ID 150.1.4.4, Network Type POINT_TO_MULTIPOINT,
Cost: 12953

Rack1R1#show ip route eigrp
     136.1.0.0/16 is variably subnetted, 10 subnets, 2 masks
D       136.1.23.2/32 [90/20514560] via 136.1.136.3, 00:03:06,
FastEthernet0/0
D       136.1.23.0/24 [90/20514560] via 136.1.136.3, 00:03:06,
FastEthernet0/0
     150.1.0.0/24 is subnetted, 6 subnets
D       150.1.6.0 [90/156160] via 136.1.136.6, 00:02:53,
FastEthernet0/0
D       150.1.3.0 [90/156160] via 136.1.136.3, 00:03:06,
FastEthernet0/0
```

Task 2.5

R5:

```
router ospf 1
 timers lsa arrival 2000
 timers pacing lsa-group 40
```

Task 2.5 Verification

```
Rack1R5#show ip ospf | include arrival|pacing
 Minimum LSA arrival 2000 msecs
 LSA group pacing timer 40 secs
 Interface flood pacing timer 33 msecs
 Retransmission pacing timer 66 msecs
```

Task 2.6

R1:

```
router eigrp 100
 redistribute ospf 1 metric 10000 1000 100 1 1500
!
router ospf 1
 redistribute eigrp 100 subnets route-map EIGRP2OSPF
 distance ospf external 171
!
route-map EIGRP2OSPF permit 10
 match tag 1
 set metric 1
!
route-map EIGRP2OSPF permit 1000
```

R2:

```
router eigrp 100
 redistribute ospf 1 metric 10000 1000 100 1 1500
!
router ospf 1
 redistribute eigrp 100 subnets route-map EIGRP2OSPF
 distance ospf external 171
```



```
!
route-map EIGRP2OSPF permit 10
 match tag 3
 set metric 1
!
route-map EIGRP2OSPF permit 1000
```

R5:

```
router ospf 1
 redistribute rip subnets
!
router rip
 distance 109
 redistribute ospf 1 metric 1
```

R6:

```
router eigrp 100
 redistribute rip metric 10000 1000 100 1 1500 route-map RIP2EIGRP
!
router rip
  redistribute eigrp 100 metric 1
!
ip prefix-list RIP_FROM_BB1 seq 5 permit 212.18.0.0/22 ge 24 le 24
!
ip prefix-list RIP_FROM_BB3 seq 5 permit 30.0.0.0/14 ge 16 le 16
ip prefix-list RIP_FROM_BB3 seq 10 permit 31.0.0.0/14 ge 16 le 16
!
!
route-map RIP2EIGRP permit 10
 match ip address prefix-list RIP_FROM_BB1
 set tag 1
!
route-map RIP2EIGRP permit 20
 match ip address prefix-list RIP_FROM_BB3
 set tag 3
!
route-map RIP2EIGRP permit 30
```

Task 2.6 Breakdown

The above redistribution configuration utilizes routing tags to distinguish the origin of the specified prefixes. To set a routing tag, simply issue the **set tag [*tag*]** command under a route-map used for redistribution.

The above task states that R5 should use R1 to get to the routes learned from BB1, while using R2 to get to the routes learned form BB3. In order to accomplish this, routes from BB1 and BB3 are set with separate tag values when redistributed into EIGRP on R6. As EIGRP is redistributed into OSPF on both R1 and R2, these tag values are *matched*, and an appropriate OSPF cost value is assigned.

Task 2.6 Verification

Verify the BB1 prefixes on R5:

```
Rack1R5#show ip route | inc 212.18
O E2 212.18.1.0/24 [110/1] via 136.1.15.1, 00:01:29, Serial0/0/0.15
O E2 212.18.0.0/24 [110/1] via 136.1.15.1, 00:01:29, Serial0/0/0.15
O E2 212.18.3.0/24 [110/1] via 136.1.15.1, 00:01:29, Serial0/0/0.15
O E2 212.18.2.0/24 [110/1] via 136.1.15.1, 00:01:29, Serial0/0/0.15
```

Verify the BB3 prefixes on R5:

```
Rack1R5#show ip route | inc ( 31| 30).*via
O E2    31.3.0.0 [110/1] via 136.1.245.2, 00:03:18, Serial0/0/0.245
O E2    31.2.0.0 [110/1] via 136.1.245.2, 00:03:18, Serial0/0/0.245
O E2    31.1.0.0 [110/1] via 136.1.245.2, 00:03:18, Serial0/0/0.245
O E2    31.0.0.0 [110/1] via 136.1.245.2, 00:03:18, Serial0/0/0.245
O E2    30.2.0.0 [110/1] via 136.1.245.2, 00:03:18, Serial0/0/0.245
O E2    30.3.0.0 [110/1] via 136.1.245.2, 00:03:18, Serial0/0/0.245
O E2    30.0.0.0 [110/1] via 136.1.245.2, 00:03:18, Serial0/0/0.245
O E2    30.1.0.0 [110/1] via 136.1.245.2, 00:03:18, Serial0/0/0.245
```

V*erify connectivity between the internal routers:*

```
tclsh
foreach i {
136.1.136.1
136.1.15.1
150.1.1.1
136.1.245.2
150.1.2.2
136.1.23.2
136.1.136.3
150.1.3.3
136.1.23.3
136.1.245.4
136.1.4.4
150.1.4.4
136.1.45.4
136.1.44.4
136.1.245.5
136.1.15.5
150.1.5.5
136.1.45.5
136.1.57.5
192.10.1.5
136.1.136.6
54.1.3.6
150.1.6.6
204.12.1.6
150.1.7.7
136.1.57.7
} { puts "exec [ping $i]" }
```

Note that VLAN3 and VLAN29 are not advertised by an IGP protocol. This can create possible BGP next hop issues with the BGP peering session.


```
Finally verify connectivity to the backbone IGP networks:

foreach i {
212.18.1.1
212.18.0.1
212.18.3.1
212.18.2.1
31.3.0.1
31.2.0.1
31.1.0.1
31.0.0.1
30.2.0.1
30.3.0.1
30.0.0.1
30.1.0.1
192.10.1.254
54.1.3.254
204.12.1.254
} { puts "exec [ping $i]" }
```

Task 2.7

```
R6:
router bgp 100
 neighbor 54.1.3.254 route-map NO_EXPORT in
 neighbor 136.1.136.1 send-community
 neighbor 136.1.136.3 send-community
 neighbor 204.12.1.254 route-map NO_EXPORT in
!
route-map NO_EXPORT permit 10
 set community no-export
```

Task 2.7 Breakdown

In order to prevent your AS from being used as transit, prefixes which are learned from a provider should not be advertised out to another provider. In the above case, R6 is learning prefixes from AS 54 through two entry points. In order to prevent other ASs (such as AS 300) from using AS 100 as transit to reach these prefixes, AS 100 must prevent them from being advertised. However the task in question states that this "configuration should be done only on R6." Since R6 is not peering with AS 300 or AS 200, this poses a problem. In order to affect whether or not other peers in AS 100 advertise these prefixes, R6 is setting the community to *no-export.*

Communities are special tag values that can be applied to prefixes in order to group them in common categories. In addition to these arbitrary numerical communities, various *well-known* communities have been defined.

Well Known Community	Behavior
Internet	All routes belong to this community by default. The Internet community has no special behavior.

No-advertise	Do not advertise to *any* BGP neighbor.
No-export	Do not advertise to any *EBGP* neighbor.
Local-AS	Do not advertise outside of sub-AS. This is a special case of no-export for use inside of a confederation.

By setting the prefixes learned from AS 54 to *no-export*, R1 and R3 will not advertise them to their EBGP neighbors. This will prevent AS 100 from being used as transit to reach these prefixes.

☠ Pitfall

BGP community attributes are not included in advertisements by default. In order to enable the sending of the community attribute along with a prefix, use the BGP routing process subcommand **neighbor [*neighbor*] send-community**. Unless this command is included, the community value will be stripped from an advertisement, regardless of whether it is going to an iBGP or EBGP neighbor.

Task 2.7 Verification

Verify that the communities are being sent:

```
Rack1R6#show ip bgp neighbors 136.1.136.3 | include Comm
  Community attribute sent to this neighbor
```

Check that R1 and R3 are actually receiving prefixes from AS 54

```
Rack1R3#show ip bgp regexp _54$
<output omitted>

   Network          Next Hop            Metric LocPrf Weight Path
*>i28.119.16.0/24   204.12.1.254             0    100      0 54 i
*>i28.119.17.0/24   204.12.1.254             0    100      0 54 i
*>i114.0.0.0        204.12.1.254             0    100      0 54 i
<output omitted>
```

Check that AS 54 prefixes are not being exported to eBGP peers:

```
Rack1R3#show ip bgp 114.0.0.0
BGP routing table entry for 114.0.0.0/8, version 37
Paths: (1 available, best #1, table Default-IP-Routing-Table, not
advertised to EBGP peer)
  Not advertised to any peer
  54
    204.12.1.254 (metric 537600) from 136.1.136.6 (150.1.6.6)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Community: no-export
```


Task 2.8

R1:

```
router bgp 100
 neighbor 136.1.15.5 route-map TO_R5 out
!
ip prefix-list VLAN3 seq 5 permit 136.1.3.0/24
!
route-map TO_R5 permit 10
 match ip address prefix-list VLAN3
 set as-path prepend 100 100
!
route-map TO_R5 permit 1000
```

R3:

```
router bgp 100
 network 136.1.3.0 mask 255.255.255.0
```

Task 2.8 Breakdown

In the above, task AS 100 is trying to affect the inbound traffic flow from its BGP peers. Recall which attribute should be set to affect BGP path selection:

Attribute	Direction Applied	Traffic Flow Affected
Weight	Inbound	Outbound
Local-Preference	Inbound	Outbound
AS-Path	Outbound	Inbound
MED	Outbound	Inbound

To affect inbound traffic flow, you must either prepend the AS-path attribute or set the multi-exit discriminator (MED) value as the prefix is advertised outside the AS. However, since MED is only compared by default on prefixes learned from the same AS, AS-path prepending must be used in this case.

Since AS 100 wants traffic to come in from AS 300, it is prepending its own AS number out twice when sending updates to AS 200. Therefore, when AS 200 compares the AS-path length on the prefixes learned from AS 100 and AS 300, it will prefer the path through AS 300.

Task 2.8 Verification

Verify R5's BGP table and RIB for the VLAN3 subnet:

```
Rack1R5#show ip bgp 136.1.3.0
BGP routing table entry for 136.1.3.0/24, version 26
Paths: (2 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Advertised to update-groups:
     1          2
  300 100
    136.1.245.2 from 136.1.245.2 (150.1.2.2)
```



```
      Origin IGP, localpref 100, valid, external, best
  100 100 100
    136.1.15.1 from 136.1.15.1 (150.1.1.1)
      Origin IGP, localpref 100, valid, external

Rack1R5#show ip route 136.1.3.0
Routing entry for 136.1.3.0/24
  Known via "bgp 200", distance 20, metric 0
  Tag 300, type external
  Last update from 136.1.245.2 00:01:26 ago
  Routing Descriptor Blocks:
  * 136.1.245.2, from 136.1.245.2, 00:01:26 ago
      Route metric is 0, traffic share count is 1
      AS Hops 2
      Route tag 300
```

Task 2.9

R2:

```
router bgp 300
 network 136.1.29.0 mask 255.255.255.0
```

R5:

```
router bgp 200
 neighbor 136.1.245.2 route-map FROM_R2 in
!
ip prefix-list VLAN29 seq 5 permit 136.1.29.0/24
!
route-map FROM_R2 permit 10
 match ip address prefix-list VLAN29
 set weight 100
!
route-map FROM_R2 permit 1000
```

Task 2.9 Verification

Verify that VLAN29 has a weight of 100 in the BGP table and no other prefixes (e.g. 205.90.31.0) are affected:

```
Rack1R5#show ip bgp
<output omitted>

Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop       Metric LocPrf Weight Path
*  136.1.29.0/24    136.1.15.1                           0 100 300 i
*>                  136.1.245.2         0           100 300 i
*  136.1.3.0/24     136.1.15.1                           0 100 100 100 i
*>                  136.1.245.2                          0 300 100 i
*> 205.90.31.0      192.10.1.254        0                0 254 ?
*> 220.20.3.0       192.10.1.254        0                0 254 ?
*> 222.22.2.0       192.10.1.254        0                0 254
<output omitted>
```


Task 2.10

R2:

```
router bgp 300
 network 136.1.23.0 mask 255.255.255.0
 neighbor 136.1.245.5 advertise-map ADVERTISE non-exist-map NON_EXIST
!
ip prefix-list SERIAL seq 5 permit 136.1.23.0/24
!
ip prefix-list VLAN29 seq 5 permit 136.1.29.0/24
!
route-map NON_EXIST permit 10
 match ip address prefix-list SERIAL
!
route-map ADVERTISE permit 10
 match ip address prefix-list VLAN29
```

Task 2.10 Breakdown

The order of the BGP best-path selection algorithm dictates that you have absolute control over how traffic leaves your autonomous system. This is due to the fact that the attributes used to affect outbound traffic (weight and local-preference) are higher in the decision process than those used to affect inbound traffic (AS-path and MED). By setting either the weight or local-preference on a prefix, you can affect how traffic leaves your AS to get to that prefix. Under certain circumstances, this behavior may be undesirable. Therefore, BGP conditional advertisement offers an alternative way to affect how traffic enters your AS. By not advertising a prefix to a specific neighbor, it is forced to route through a peer that does have a route to it.

This feature is typically used when you have multiple connections of varying speeds to one or more providers. By controlling which prefixes get advertised to which neighbors, traffic can be forced to route in the appropriate link. In the case of a link failure, conditional advertisement will begin advertising the prefixes in question to the configured neighbor.

BGP conditional advertisement consists of two parts, the prefix or prefixes to watch, and the prefix or prefixes to advertise. It is applied by issuing the BGP process subcommand **neighbor [*neighbor*] advertise-map [*advertise-map*] non-exist-map [*non-exist-map*]**, where **advertise-map** is a route-map that matches the prefix that will be conditionally advertised, and **non-exist-map** is a route-map that matches the prefix to be watched.

Once the prefix in the **non-exist-map** leaves the BGP table, the prefix in the **advertise-map** is advertised to the configured neighbor. Note that both of these prefixes must exist in the BGP table before configuring conditional advertisement.

In the above task AS 300 wants all traffic for the prefix 136.1.29.0/24 to come in the HDLC link 136.1.23.0/24, unless it is down. To accomplish this, the

136.1.29.0/24 prefix should only be advertised from R2 to R5 if the HDLC link 136.1.23.0/24 is down. 136.1.29.0/24 will therefore be matched in the advertise-map, while 136.1.23.0/24 is matched in the non-exist-map.

Task 2.10 Verification

```
Rack1R2#show ip bgp neighbors 136.1.245.5 | include Condition
  Condition-map NON_EXIST, Advertise-map ADVERTISE, status: Withdraw
                                                      ↑
                                              Prefix Not Advertised

Rack1R2#show ip bgp neighbors 136.1.245.5 advertised-routes
BGP table version is 4, local router ID is 2.2.2.2
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
*> 136.1.23.0/24    0.0.0.0                  0         32768 i

Rack1R2(config)#interface Serial0/1
Rack1R2(config-if)#shutdown  ← Serial link fails

Rack1R2#show ip bgp 136.1.23.0
% Network not in table← Prefix no longer exists
```

```
Rack1R2#show ip bgp neighbors 136.1.245.5 | include Condition
  Condition-map NON_EXIST, Advertise-map ADVERTISE, status: Advertise
                                                    ↑
                                          Prefix Now Advertised

Rack1R2#show ip bgp neighbors 136.1.245.5 advertised-routes
BGP table version is 6, local router ID is 2.2.2.2
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
*> 136.1.29.0/24     0.0.0.0                  0         32768 i
        ↑
Prefix Now Advertised
```

Task 2.11

R3:

```
router bgp 100
 neighbor 136.1.23.2 allowas-in
```

SW3:

```
router bgp 100
 network 136.1.109.0 mask 255.255.255.0
 neighbor 136.1.29.2 allowas-in
```

Task 2.11 Breakdown

By default, if a BGP speaker receives an update with its own AS in the AS path, the update is dropped. This is used by BGP as a means of loop prevention and is the normal desired behavior.

To allow a router to accept a BGP update with its own AS in the AS path, the *allowas-in* option is used. Another option to solve this problem, although not permitted by this particular task, would be to configure R2 to use the AS override feature. When AS override option is used, if an update is to be sent to a BGP peer that already contains the remote BGP peer's AS in the AS path, the local BGP AS is replaced with the peer's AS. This will ensure the remote BGP peer will accept the BGP update, since its AS is not in the AS path anymore.

Task 2.11 Verification

```
Rack1R3#show ip bgp 136.1.109.0
BGP routing table entry for 136.1.109.0/24, version 40
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to update-groups:
     1
  300 100
    136.1.23.2 from 136.1.23.2 (150.1.2.2)
      Origin IGP, localpref 100, valid, external, best

Rack1SW3#show ip bgp
BGP table version is 19, local router ID is 150.1.9.9
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   136.1.29.2                             0 300 100 54 i
*> 28.119.17.0/24   136.1.29.2                             0 300 100 54 i
<output omitted>
*> 136.1.109.0/24   0.0.0.0                  0         32768 i
*> 205.90.31.0      136.1.29.2                             0 300 200 254 ?
*> 220.20.3.0       136.1.29.2                             0 300 200 254 ?
*> 222.22.2.0       136.1.29.2                             0 300 200 254 ?
```

Task 3.1

R2:

```
interface FastEthernet0/0
 ipv6 address 2001:CC1E:1:202::/64 eui-64
```

R4:

```
interface FastEthernet0/0
 ipv6 address 2001:CC1E:1:404::/64 eui-64
```

Task 3.1 Verification

Verify IPv6 addressing (note the EUI-64 host part):

```
Rack1R4#show ipv6 interface brief
Ethernet0/0                [up/up]
    FE80::230:94FF:FE7E:E581
    2001:CC1E:1:404:230:94FF:FE7E:E581
<output omitted>
```

Task 3.2

R2:

```
interface Tunnel0
 ipv6 address FEC0::2/64
 tunnel source 150.1.2.2
 tunnel destination 150.1.4.4
!
```



```
ipv6 route 2001:CC1E:1:404::/64 Tunnel0
```

R4:

```
interface Tunnel0
 ipv6 address FEC0::4/64
 tunnel source 150.1.4.4
 tunnel destination 150.1.2.2
!
ipv6 route 2001:CC1E:1:202::/64 Tunnel0
```

Task 3.1 – 3.2 Breakdown

Site-local IPv6 addresses are similar to private addresses defined in RFC 1918 for IPv4, as they are designed to be used as address space only routable within a private network. Site-local IPv6 addresses start with the bit pattern 1111111011, expressed as FEC0::/10 in IPv6 address format (not to be confused with the link-local FE80::/10 address space). By identifying the site-local address space with a unique prefix (FEC0::/10), ingress and egress filtering can be easily applied. Filtering can prevent traffic from/to hosts using this address space from moving between site boundaries at the network edge.
The above example illustrates how to tunnel IPv6 datagrams over the IPv4 network using GRE encapsulation. This configuration is similar to that of IPv6IP tunneling. One advantage with GRE is support for tunneling of multiple non-IP protocol stacks simultaneously (IPX, CLNS, etc).

Task 3.1 Verification

Verify the tunnel:

```
Rack1R4#show interfaces tunnel 0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 150.1.4.4, destination 150.1.2.2
  Tunnel protocol/transport GRE/IP
```

Verify addressing:

```
Rack1R4#show ipv6 interface brief
<output omitted>
Tunnel0                    [up/up]
    FE80::230:94FF:FE7E:E581
    FEC0::4
```

Verify L3 connectivity:

```
Rack1R4#ping fec0::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FEC0::2, timeout is 2 seconds:
!!!!!
```



```
Success rate is 100 percent (5/5), round-trip min/avg/max = 68/71/72 ms

Verify if static routing works (Note: You should have a different IPv6
EUI-64 host identifier):

Rack1R4#ping 2001:CC1E:1:202:204:27FF:FEB5:2FA0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:CC1E:1:202:204:27FF:FEB5:2FA0,
timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/72/76 ms
```

Task 4.1

R4:

```
!
! Interface prefix MUST be /32
!
interface Loopback0
 ip address 150.1.4.4 255.255.255.25
!
mpls ip
mpls label protocol tdp
mpls ldp router-id loopback 0 force
!
interface Serial 0/1/0
 mpls ip
 mpls ldp discovery transport-address interface
 mpls label protocol tdp
!
interface Serial 0/0/0
 mpls ip
 mpls ldp discovery transport-address interface
 mpls label protocol tdp
```

R5:

```
!
! Interface prefix MUST be /32
!
interface Loopback0
 ip address 150.1.5.5 255.255.255.25
!
mpls ip
mpls ldp router-id loopback 0 force
!
interface Serial 0/1/0
 mpls ip
 mpls ldp discovery transport-address interface
 mpls label protocol tdp
!
interface Serial 0/0/0.245
 mpls ip
 mpls ldp discovery transport-address interface
 mpls label protocol tdp
```


Task 4.1 Verification

```
Rack1R5#show mpls ldp neighbor
    Peer TDP Ident: 150.1.4.4:0; Local TDP Ident 150.1.5.5:0
        TCP connection: 136.1.245.4.711 - 136.1.245.5.40771
        State: Oper; PIEs sent/rcvd: 0/8; Downstream
        Up time: 00:03:53
        TDP discovery sources:
          Serial0/0/0.245, Src IP addr: 136.1.245.4
        Addresses bound to peer TDP Ident:
          136.1.4.4       136.1.44.4      136.1.245.4     136.1.45.4
          150.1.4.4

Rack1R5#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next
Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     136.1.4.0/24      0             Se0/0/0.245
136.1.245.4
17     Pop Label     136.1.44.0/24     0             Se0/0/0.245
136.1.245.4
18     Pop Label     136.1.45.4/32     0             Se0/1/0
point2point
19     Pop Label     136.1.245.4/32    0             Se0/0/0.245
136.1.245.4
20     Pop Label     150.1.4.4/32      0             Se0/0/0.245
136.1.245.4
21     Pop Label     136.1.245.2/32    0             Se0/0/0.245
136.1.245.2
22     No Label      150.1.2.0/24      0             Se0/0/0.245
136.1.245.2
23     No Label      136.1.23.3/32     0             Se0/0/0.245
136.1.245.2
24     No Label      150.1.3.0/24      0             Se0/0/0.245
136.1.245.2
25     No Label      136.1.136.0/24    0             Se0/0/0.15
point2point
       No Label      136.1.136.0/24    0             Se0/0/0.245
136.1.245.2
26     No Label      150.1.1.0/24      0             Se0/0/0.15
point2point

Rack1R4#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next
Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     136.1.45.5/32     0             Se0/1/0
point2point
17     Pop Label     150.1.5.5/32      0             Se0/0/0
136.1.245.5
18     Pop Label     192.10.1.0/24     0             Se0/0/0
136.1.245.5
19     Pop Label     136.1.245.5/32    0             Se0/0/0
136.1.245.5
```



```
20     21            136.1.245.2/32    0             Se0/0/0
136.1.245.5
21     22            150.1.2.0/24      0             Se0/0/0
136.1.245.5
22     23            136.1.23.3/32     0             Se0/0/0
136.1.245.5
23     24            150.1.3.0/24      0             Se0/0/0
136.1.245.5
24     25            136.1.136.0/24    0             Se0/0/0
136.1.245.5
25     No Label      136.1.23.0/24     0             Se0/0/0
136.1.245.5
26     Pop Label     136.1.15.0/24     0             Se0/0/0
136.1.245.5
27     26            150.1.1.0/24      0             Se0/0/0
136.1.245.5
```

Task 4.2

R4:

```
ip vrf VPN_AB
 rd 100:47
 route-target export 100:47
 route-target import 100:74
!
interface FastEthernet 0/1
 ip vrf forwarding VPN_AB
 ip address 136.1.44.4 255.255.255.0
!
router bgp 200
 address-family ipv4 unicast VRF VPN_AB
  redistribute connected
 !
 address-family vpnv4 unicast
  neighbor 150.1.5.5 activate
  neighbor 150.1.5.5 send-community both
```

R5:

```
ip vrf VPN_AB
 rd 100:47
 route-target export 100:74
 route-target import 100:47
!
interface FastEthernet 0/1
 ip vrf forwarding VPN_AB
 ip address 136.1.57.5 255.255.255.0
!
router bgp 200
 address-family ipv4 unicast VRF VPN_AB
  redistribute connected
 !
 address-family vpnv4 unicast
  neighbor 150.1.4.4 activate
  neighbor 150.1.4.4 send-community both
```


Task 4.2 Verification

```
Rack1R5#ping vrf VPN_AB 136.1.44.4 source fastEthernet 0/1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 136.1.44.4, timeout is 2 seconds:
Packet sent with a source address of 136.1.57.5
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms

Rack1R5#show ip cef vrf VPN_AB 136.1.44.4
136.1.44.0/24
  nexthop 136.1.245.4 Serial0/0/0.245 label 29

Rack1R5#show bgp vpnv4 unicast all 136.1.44.0
BGP routing table entry for 100:47:136.1.44.0/24, version 9
Paths: (1 available, best #1, table VPN_AB)
Flag: 0x820
  Not advertised to any peer
  Local
    150.1.4.4 (metric 12954) from 150.1.4.4 (150.1.4.4)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:47
      mpls labels in/out nolabel/29
```

Task 4.3

R4:

```
!
! To make sure the routers translate updates to type-3 LSA properly,
! OSPF domain IDs should match; by default domain-id is baased
! on the ospf process number.
!
router ospf 44 vrf VPN_AB
 network 0.0.0.0 255.255.255.255 area 44
 redistribute bgp 200 subnets
 domain-id 47.47.47.47
!
router bgp 200
 address-family ipv4 vrf VPN_AB
  redistribute ospf 44
```

R5:

```
router ospf 77 vrf VPN_AB
 network 0.0.0.0 255.255.255.255 area 77
 !
 ! Reflect SW1 settings for the NSSA area!
 !
 area 77 nssa
 redistribute bgp 200 subnets
 domain-id 47.47.47.47
!
router bgp 200
 address-family ipv4 vrf VPN_AB
  redistribute ospf 77
```


Task 4.3 Verification

```
Rack1SW2#show ip route ospf
     136.1.0.0/24 is subnetted, 3 subnets
O IA    136.1.7.0 [110/3] via 136.1.44.4, 00:02:38, Vlan44
O IA    136.1.57.0 [110/2] via 136.1.44.4, 00:02:53, Vlan44
     150.1.0.0/32 is subnetted, 1 subnets
O IA    150.1.7.7 [110/3] via 136.1.44.4, 00:02:38, Vlan44

Rack1SW1#show ip route ospf
     136.1.0.0/24 is subnetted, 3 subnets
O IA    136.1.44.0 [110/2] via 136.1.57.5, 00:05:00, Vlan57
     150.1.0.0/16 is variably subnetted, 2 subnets, 2 masks
O IA    150.1.8.8/32 [110/3] via 136.1.57.5, 00:00:14, Vlan57

Rack1SW1#ping 150.1.8.8 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.8.8, timeout is 2 seconds:
Packet sent with a source address of 150.1.7.7
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 58/62/67 ms

Rack1SW1#traceroute 150.1.8.8

Type escape sequence to abort.
Tracing the route to 150.1.8.8

  1 136.1.57.5 0 msec 9 msec 0 msec
  2 136.1.44.4 67 msec 67 msec 59 msec
  3 136.1.44.8 42 msec *  33 msec
```

Task 5.1

R2:

```
interface FastEthernet0/0
 ip igmp static-group 228.22.22.22
```

Task 5.1 Breakdown

The **ip igmp static-group** command is different than the **ip igmp join-group** command in that the static-group command does not cause the devices to process the multicast packets themselves. Instead the device will just forward the multicast packets out the interface.

With the **ip igmp static-group** command, a device will not respond to ICMP echo requests sent to the multicast group as with the **ip igmp join-group** command. If a particular multicast group needs to be sent out an interface, the static-group command would be preferred over the join-group command. The static-group command causes the group to be fast switched. The join-group command will cause the device to process switch the group. Neither command is needed to enable hosts on a segment to receive multicast traffic, as long as the host supports IGMP.

Task 5.1 Verification

Verify that the 228.22.22.22 group is statically configured:

```
Rack1R2#show ip igmp membership
Flags: A  - aggregate, T - tracked
       L  - Local, S - static, V - virtual, R - Reported through v3
       I - v3lite, U - Urd, M - SSM (S,G) channel
       1,2,3 - The version of IGMP the group is in
<output omitted>

 Channel/Group        Reporter        Uptime   Exp.  Flags  Interface
 *,224.0.1.39         136.1.245.5     00:05:37 02:44 2A     Se0/0
 *,224.0.1.40         136.1.29.2       00:06:31 02:34 2LA    Fa0/0
 *,228.22.22.22       0.0.0.0         00:00:08 stop  2SA    Fa0/0
```

Verify multicast routing (note that 228.22.22.22 has no RP, and is flooded in dense mode):

```
Rack1R5#ping 228.22.22.22

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 228.22.22.22, timeout is 2 seconds:
.

Rack1R2#show ip mroute
IP Multicast Routing Table
<output omitted>

(*, 228.22.22.22), 02:50:04/stopped, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 02:48:50/00:00:00
    Serial0/0, Forward/Sparse-Dense, 02:50:04/00:00:00

(136.1.245.5, 228.22.22.22), 00:00:09/00:02:53, flags: T
  Incoming interface: Serial0/0, RPF nbr 136.1.245.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:00:09/00:00:0
```


Task 5.2

R4:

```
interface FastEthernet0/0
 ip igmp access-group 50
!
access-list 50 permit 225.25.25.25
access-list 50 permit 226.26.26.26
```

Task 5.2 Breakdown

To limit which multicast groups clients on a segment can join, use the **ip igmp access-group** interface command. In this task, clients will only be allowed to join multicast groups 225.25.25.25 and 226.26.26.26.

Task 5.2 Verification

Verify that the filtering is configured:

```
Rack1R4#show ip igmp interface Fa0/0 | inc access
  Inbound IGMP access group is 50

Rack1R4#show ip access-lists 50
Standard IP access list 50
    10 permit 225.25.25.25
    20 permit 226.26.26.26
```

Task 5.3

R1:

```
interface FastEthernet0/0
 ip multicast ttl-threshold 12
```

Task 5.3 Breakdown

Whenever a multicast packet is sent by a server, the TTL of the packet is set. The TTL can be set anywhere from 0 to 255. A multicast packet sent with a TTL of 1, or technically 0, will not allow a multicast router to forward the packet. This is the mechanism that OSPF uses to ensure that its packets are never sent across a router.

By using TTL, a multicast implementation can limit where the multicast traffic can be sent in the network or stop the multicast traffic from being sent out of a network. This is commonly referred to as *multicast scoping*. Multicast scoping can be difficult to implement reliably in a medium or large sized network. Issues like alternate paths in the network or tunnels used to carry multicast traffic can make scoping nearly impossible to properly implement.

The **ip multicast ttl-threshold** interface command stops the forwarding of multicast packets that do not have a TTL value greater than what is defined by the command.

Task 6.1

R6:

```
interface Serial0/0/0
 ip access-group IN_ACL in
 ip access-group OUT_ACL out
!
ip access-list extended IN_ACL
 permit icmp any any time-exceeded
 permit icmp any any port-unreachable
 permit udp any any eq rip
 permit tcp any any eq bgp
 permit tcp any eq bgp any
 evaluate MY_REFLECT
!
ip access-list extended OUT_ACL
 permit tcp any any reflect MY_REFLECT
 permit udp any any reflect MY_REFLECT
 permit icmp any any reflect MY_REFLECT
```

Task 6.1 Breakdown

A reflexive access-list is used to meet the requirements of this section. Remember that by default, packets generated by the router itself will not be reflected. This is why BGP is explicitly permitted inbound. Instead of using a reflexive access list, CBAC could also be used to meet the section requirements.

To permit traceroute, we first need to fully understand how traceroute works. First off, it is important to think of traceroute as a technique and not necessarily an application. The goal of traceroute is the find the path the packets take between the source and destination. To do this, traceroute needs to accomplish two tasks. The first task is to discover any routers along the path. This is done by manipulating the time-to-live (TTL). The second task is to have the destination respond, so that traceroute knows that the destination has been reached. Depending on the particular implementation of traceroute, this response could be an ICMP echo reply, ICMP port unreachable, or in the case of TCP based traceroute, a ‘SYN, ACK’.

TTL is used by routers to ensure a packet will not remain on the network indefinitely due to issues like a routing loop. When a packet is received by a router, the router must decrement the TTL by at least 1 before forwarding it. Technically, a router must decrement the TTL for each second that the router holds the packet. Of course, holding a packet for more than a few milliseconds, much less a few seconds, would be rare in today's networks. When the TTL has been decremented to 0, the router must discard the packet. When a packet is discarded due to the TTL expiring, an ICMP time-exceeded message is sent by the router that discarded the packet to the source. This packet notifies the source that the packet expired in transit. This is the particular message that a traceroute application is looking to receive back from the routers in the path between the source and destination. When a router generates the ICMP time-exceeded, the traceroute application discovers that router and the traceroute application now knows that the packet should transit that particular router.

Now that the routers in the path have been discovered, the source needs to know when the destination has been reached. Traceroute needs to send a packet that will entice the destination to reply. Technically, this could be any type of packet that would cause the destination to send a return packet to the source. The packet type will depend on the particular implementation of the traceroute application.

Now that we know how traceroute works and what traceroute is trying to do, we can look at how traceroute is implemented. There are implementations of traceroute that use ICMP, UDP and even TCP. The Cisco IOS uses UDP, Microsoft uses ICMP, and new versions of Linux use ICMP or UDP. There are also various TCP traceroute applications available for different operating systems.

Here is an example of TCP traceroute to www.cisco.com port 80.

```
[root@Homer bdennis]# tcptraceroute -n www.cisco.com 80
Selected device eth0, address 172.16.1.250, port 32795 for outgoing
packets
Tracing the path to www.cisco.com (198.133.219.25) on TCP port 80, 30
hops max
 1  64.172.154.254 (64.172.154.254)  38.485 ms  13.425 ms  39.358 ms
 2  63.234.16.33 (63.234.16.33)  13.793 ms  15.680 ms  26.380 ms
 3  63.234.16.8 (63.234.16.8)  15.852 ms  28.993 ms  13.263 ms
 4  151.164.181.73 (151.164.181.73)  34.957 ms  36.938 ms  31.576 ms
 5  151.164.240.134 (151.164.240.134)  31.983 ms  34.405 ms  31.756 ms
 6  144.228.44.49 (144.228.44.49)  33.196 ms  34.641 ms  31.970 ms
 7  144.232.0.225 (144.232.0.225)  31.998 ms  36.637 ms  31.794 ms
 8  144.232.3.138 (144.232.3.138)  31.458 ms  34.687 ms  31.498 ms
 9  144.228.44.14 (144.228.44.14)  32.125 ms  34.090 ms  31.759 ms
10  128.107.239.89 (128.107.239.89)  31.990 ms  34.416 ms  31.970 ms
11  128.107.239.98 (128.107.239.98)  34.473 ms  36.888 ms  31.753 ms
12  198.133.219.25 (198.133.219.25) [open]  31.808 ms  39.538 ms
37.105 ms
[root@Homer bdennis]#
```

Note

One of the reasons why many traceroute applications do not use ICMP echos is in accordance with RFC 792 (Internet Control Message Protocol). RFC 792 states that since ICMP is typically used to report errors, ICMP messages should not be send about other ICMP messages.

Remember that traceroute is trying to discover the routers along the path by manipulating the TTL. The TTL field is in the header of the IP packet, so this means that ICMP, UDP and TCP packets can be used. Any of these protocols can have a packet sent with a TTL of 1. When the first router in the path receives the packet and decrements the TTL to 0, an ICMP time-exceeded message is generated and the packet is discarded. When the traceroute application receives the ICMP time-exceeded, the application will then generate another packet with a TTL of 2. This cycle will continue till the final destination is reached.

After the final destination has been reached, the reply from the destination will depend on the packet type used by the traceroute application. If the packet generated by the traceroute application is an UDP packet (as in the case of a Cisco router), the destination of the packet will normally be sent to a UDP port that is not being used by the destination. When the UDP packet arrives at the destination device, sent to the unused UDP port, the destination device will generate an ICMP port unreachable message. When that ICMP port unreachable is received by the source's traceroute application, the source will then know it has reached the destination. Below is the output from traceroute on a Cisco router.

```
IE#traceroute
Protocol [ip]:
Target IP address: 198.133.219.25
Source address:
Numeric display [n]: y
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 198.133.219.25

  1 63.115.78.1 4 msec 4 msec 4 msec
  2 157.130.213.37 12 msec 12 msec 12 msec
  3 65.208.80.242 16 msec 16 msec 16 msec
  4 128.107.239.5 16 msec 16 msec 16 msec
  5 128.107.239.106 16 msec 12 msec 12 msec
  6 198.133.219.25 16 msec 16 msec 16 msec
IE#
```

As we can see, the router did not perform DNS lookups on the replies. This is desirable in a lab environment were DNS lookups can not be resolved. It generated 3 packets per TTL as indicated by the 'probe count'. The first packet was sent with a TTL of 1 and a destination UDP port of 33434.

If the packet generated by the traceroute application is an ICMP echo request packet, as in the case of Microsoft Windows, the destination will reply with an ICMP echo reply. The same process as described before happens in regards to sending the first packet with a TTL of 1, second packet with a TTL of 2, etc. The traceroute application will know that the destination has been reached when an ICMP echo-reply is received.

Now, we can look at our configuration and understand why we explicitly permitted ICMP time-exceeded and ICMP port-unreachable messages. The time-exceeded messages are needed when a router in the path discards a packet due to the TTL expiring. The port unreachable messages are needed by UDP based traceroute (i.e. Cisco IOS). If ICMP echo based traceroute was used, we would not need to permit the ICMP port-unreachables as the reflective access-list will detect the ICMP echo request and dynamically allow the ICMP echo reply.

Configuring an access list on the interface blocking ICMP inbound will prevent successful pings sourced from R6. The section states to only allow ICMP traffic inbound if it originated from behind R6, but does not state anything about ICMP traffic originating from R6. When in doubt, make sure to get clarification from the proctor. This will affect the output of ping tests run from R6, so you may want to make a note to yourself that the failures are expected behavior.

Task 6.1 Verification

Verify that the BGP peering is not disrupted and RIP updates are still being received:

```
Rack1R6#show ip bgp summary | include ^54|Nei
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
54.1.3.254      4    54     364     374       40    0    0 05:36:26       10

Rack1R6#show ip route rip | inc 54.1.3.254
R    212.18.1.0/24 [120/1] via 54.1.3.254, 00:00:23, Serial0/0/0
R    212.18.0.0/24 [120/1] via 54.1.3.254, 00:00:23, Serial0/0/0
R    212.18.3.0/24 [120/1] via 54.1.3.254, 00:00:23, Serial0/0/0
R    212.18.2.0/24 [120/1] via 54.1.3.254, 00:00:23, Serial0/0/0
```

Verify that reflective ACL works. Initiate TCP traffic from R3 to BB1 and check reflective ACL:

```
Rack1R3#telnet 54.1.3.254
Trying 54.1.3.254 ... Open

BB1>

Rack1R6#show ip access-lists  MY_REFLECT
Reflexive IP access list MY_REFLECT
     permit tcp host 54.1.3.254 eq telnet host 136.1.136.3 eq 16410 (34
matches) (time left 287)
```


Verify traceroute:

```
Rack1R3#traceroute 212.18.2.1

Type escape sequence to abort.
Tracing the route to 212.18.2.1

  1 136.1.136.6 4 msec 0 msec 0 msec
  2 54.1.3.254 36 msec *  36 msec
```


Task 6.2

R4:

```
ip tcp intercept list 125
ip tcp intercept watch-timeout 15
ip tcp intercept mode watch
!
access-list 125 permit tcp any host 136.1.4.100
```

Task 6.2 Breakdown

TCP intercept is used to help prevent a TCP SYN flood DoS attack. In a SYN flood DoS attack, a source (or in most cases many sources), sends a flood of TCP SYN packets usually containing bogus (i.e. fake) source IP addresses.

The SYN packet is the first part of the TCP 3-way handshake. When a server receives the TCP SYN (synchronization) packet from a client, the server replies with a 'SYN ACK' (synchronization acknowledgement). The server will then wait for the client to complete the handshake process. For the process to be completed, the client will send an ACK in response to the server's 'SYN, ACK'. At this point the TCP session will be established. If the ACK is not received from the client, the session will timeout and in turn will be torn down. Once the session is torn down, the server's resources will be released.

A TCP SYN flood DoS attack uses this 3-way handshake process to cause the server to allocate resources for sessions that will never become established. The source or sources of the attack in most cases send thousands of TCP SYN packets per second to the server using bogus source IP addresses. The server, which does not know that the source IP addresses are bogus, receives the SYN packets, and replies with the 'SYN ACK'. The server then begins to allocate resources for the anticipated TCP session. After 10's of thousands of these SYN packets have been received by the server within a few seconds, the server will run out of resources to allocate for additional TCP sessions. The server now has thousands of half open TCP sessions that will eventually timeout after failing to receive the ACK from the client. Since most, if not all, of the server's resources are tied up replying to the SYN packets generated by attackers, legitimate users will not be able to establish a TCP session with the server.

TCP intercept can be used to enable the router to intercept the TCP SYN packets. The router will proxy for the server, and send the SYN ACK to the client. If the router receives an ACK from the client, the router knows that the session is valid and connects the session with the server. In theory, TCP intercept is a good solution in that the attack is offloaded from the server. In reality, it is not a good long term solution, as the burden of the attack is now taken on by the router. In a real network, it is normally easier to just install more (or faster) servers to deal with the burden of a DoS attack.

There are two modes of TCP intercept. The first is intercept mode and the second is watch mode. With intercept mode, the router will actively intercept the TCP sessions. In watch mode, the router will not intercept the TCP sessions but will monitor the TCP sessions. If a session does not reach the established state within 30 seconds (default time), the router will send a RST to the server so the server can release the resources allocated for that particular session. This is the intercept mode used by this section. In this section, a TCP RST packet will be sent to the server for TCP sessions that do not become established within 15 seconds. An access-list is additionally used to restrict which hosts are being 'watched'.

Task 6.2 Verification

Verify that TCP Intercept is working:

```
Rack1R5#telnet 136.1.4.100
Trying 136.1.4.100 ...

Rack1R4#show tcp intercept statistics
Watching new connections using access-list 125
1 incomplete, 0 established connections (total 1)
0 connection requests per minute

Rack1R4#show tcp intercept connections
Incomplete:
Client            Server              State    Create   Timeout  Mode
136.1.245.5:59676 136.1.4.100:23      SYNSENT  00:00:11 00:00:03 W

Established:
Client            Server              State    Create   Timeout  Mode
```


Task 7.1

R4:

```
username WEB secret CISCO
!
ip http server
ip http port 8080
ip http access-class 75
ip http authentication local
!
access-list 75 permit 136.1.2.0 0.0.0.255
```

Task 7.1 Breakdown

Although commonly not used, the IOS supports management and configuration through a web browser. In this section, the router has been configured to listen to HTTP requests on TCP port 8080. An access-list has been additionally defined to permit devices from the 136.1.2.0/24 subnet to access the router via HTTP. This is similar to applying an access-class inbound under the VTY lines.

Newer IOS versions support HTTP configuration using Secure Socket Layer (SSL).

Task 7.1 Verification

Verify the HTTP server configuration:

```
Rack1R4#show ip http server status
HTTP server status: Enabled
HTTP server port: 8080
HTTP server authentication method: local
HTTP server access class: 75
<output omitted>
```

Check to see if password for user WEB is encrypted with md5 hash:

```
Rack1R4#show running-config | inc username WEB
username WEB secret 5 $1$L09G$fX0brRRcfgoQygNfWTc0Q1
```

Task 7.2

R3:

```
tftp-server flash:c2600-iuo-mz.122-13.bin alias cisco2-C2600
```

Task 7.2 Breakdown

The key to this section is the *alias* portion of the **tftp-server** command. When a router starts to boot up, it will look in its global configuration for any boot commands. If there are not any boot commands specified, the router will fall over to using the first image in flash. If an image is not found in flash, the router will then try to boot a default image via TFTP. The default IOS image name for

the 2600 used in this lab is 'cisco2-C2600'. The image name is hardware dependent in that a 3800 will attempt to boot a different default IOS image.

To determine which IOS image a router will attempt to boot, reload the router and then send *control-break* to get into ROMMON mode. Once in ROMMON mode, type **confreg**, assuming you are using a 2600 series or higher router. You will then be able to see the default IOS image.

```
Rack1R1#reload

Proceed with reload? [confirm]

%SYS-5-RELOAD: Reload requested by console.
System Bootstrap, Version 11.3(2)XA4, RELEASE SOFTWARE (fc1)
Copyright (c) 1999 by cisco Systems, Inc.
TAC:Home:SW:IOS:Specials for info
C2600 platform with 65536 Kbytes of main memory

<BREAK SEQUENCE SENT>

PC = 0xfff0a530, Vector = 0x500, SP = 0x83fff8b0

monitor: command "boot" aborted due to user interrupt
rommon 1 > confreg

    Configuration Summary
enabled are:
load rom after netboot fails
console baud: 9600
boot: image specified by the boot system commands
      or default to: cisco2-C2600

do you wish to change the configuration? y/n  [n]:
```

Task 7.3

R2:

```
interface FastEthernet0/0
 ip directed-broadcast
```

R5:

```
interface Serial0/0/0.555 point-to-point
 ip address 136.1.5.1 255.255.255.252
 ip helper-address 136.1.29.255
 frame-relay interface-dlci 555 protocol ip 136.1.5.2
```


Task 7.3 Breakdown

The solution for this section is auto-install over Frame Relay. As mentioned previously, a router is a BOOTP server by default. In this case R5 will give the new router the IP address of 136.1.5.2 via BOOTP.

One issue with this section is that the IP address of the TFTP was not given. The only information given is that the TFTP server is located in VLAN 29. The solution is to configure the **ip helper-address** command to point to the directed broadcast for the subnet. To allow for successful transmission, ensure that the last hop interface supports directed-broadcast, which is disabled by default.

Task 7.3 Verification

Verify that the helper-address is configured:

```
Rack1R5#show ip helper-address
Interface           Helper-Address  VPN VRG Name             VRG State
Serial0/0/0.555       136.1.29.255     0   None                  None

Verify if DLCI is mapped:

Rack1R5#show frame-relay map  | beg 0/0\.555
Serial0/0/0.555 (down): point-to-point dlci, dlci 555(0x22B,0x88B0),
broadcast
 status deleted

PVC with DLCI 555 is not yet provisioned.
```

Task 7.4

R2:

```
enable secret level 1 CISCO
!
privilege exec level 0 traceroute
privilege exec level 0 ping
!
line vty 0 4
 privilege level 0
```

Task 7.4 Breakdown

Privilege levels are used to restrict user access to certain commands. There are 16 privilege levels available on the router (0-15). The privilege levels that (by default) have commands assigned to them are 0, 1, and 15. Privilege level 1 is commonly referred to as user mode, and privilege level 15 is commonly referred to as enable mode.

When first logging into a router, the default privilege level assigned to all lines (VTY, console, etc) is privilege level 1.

Note

When in privilege level 0 or 1, the router's prompt will be '>'. Any level above level 1 will have a prompt of '#'.

To change the default privilege level for a line, the **privilege level** line command is used. If the privilege level is set to 15 for a particular line the user will automatically be placed into enable mode (privilege level 15).

```
Rack1R1#show run | include (vty)|(privilege)
line vty 0 4
 privilege level 15
Rack1R1#
Rack1R1#telnet 150.1.1.1
Trying 150.1.1.1 ... Open


User Access Verification

Password:
Rack1R1#show privilege
Current privilege level is 15
Rack1R1#
```

Privilege level 0 is the lowest level on the router. There are only a few commands available to a user in privilege level 0.

```
Rack1R1>?
Exec commands:
  <1-99>   Session number to resume
  disable  Turn off privileged commands
  enable   Turn on privileged commands
  exit     Exit from the EXEC
  help     Description of the interactive help system
  logout   Exit from the EXEC
  voice    Voice Commands

Rack1R1>
```

Normally when privilege level 0 is used, additional commands are moved down to privilege level 0 from privilege level 1 or 15. To move a command from one privilege level to another, the **privilege** global configuration command is used. Commands in lower privilege levels are automatically available to users in higher privilege levels.

To switch between privilege levels, use the **enable** command. The default option on the **enable** command is '15'.

```
Rack2R3#enable ?
```



```
  <0-15>  Enable level
  <cr>

Rack2R3#enable
```

When switching from a higher privilege level to a lower privilege level, a password is not required. Only when switching from a lower level to a higher level is a password required. To configure a password for particular privilege level, use the **enable secret level** or the **enable password level** commands.

Task 7.4 Verification

Telnet to R2 and verify the privilege level 0 commands:

```
Rack1R2>ping 150.1.3.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms

Rack1R2>traceroute 150.1.6.6

Type escape sequence to abort.
Tracing the route to 150.1.6.6

  1 136.1.23.3 16 msec 16 msec 16 msec
  2 136.1.136.6 16 msec *  16 msec

Rack1R2>?
Exec commands:
  <output omitted>
  ping        Send echo messages
  traceroute  Trace route to destination
  <output omitted>
```

Task 7.5

R5:

```
privilege exec level 1 debug ip rip
privilege exec level 1 undebug ip rip
privilege exec level 1 terminal monitor
```

Task 7.5 Breakdown

☠ Pitfall

In a real network, when allowing users access to debugging command, ensure that the users are also given access to the **undebug** command.

Task 7.5 Verification

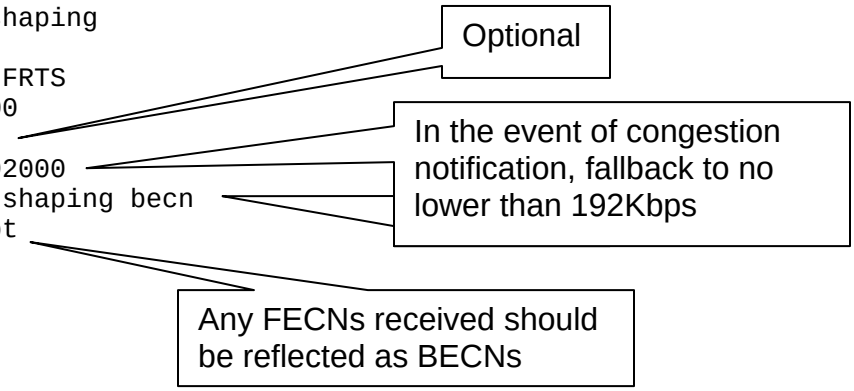
Enter privilege level 1 and verify the debug commands:

```
Rack1R5#enable 1
Rack1R5>
Rack1R5>debug ip ?
  rip  RIP protocol transactions

Rack1R5>debug ip rip
RIP protocol debugging is on

Rack1R5>undebug ip rip
```

Task 8.1

R1, R2, R4, and R5:

```
interface Serial0/0
 frame-relay class FRTS
 frame-relay traffic-shaping
!
map-class frame-relay FRTS
 frame-relay cir 256000
 frame-relay bc 32000
 frame-relay mincir 192000
 frame-relay adaptive-shaping becn
 frame-relay fecn-adapt
```



Task 8.1 Breakdown

Forward Explicit Congestion Notification (FECN) is used by the Frame Relay switch to notify a router that the remote router is causing congestion in the network. Backward Explicit Congestion Notification (BECN) is use to notify a router that it is the source of the congestion. OSI and DECnet Phase V are the only protocols that will automatically map the FECN bit to their own congestion experienced bit. This allows the devices to decrease their window size and in turn will theoretically decrease network utilization. This method of slowing down by using windowing is similar to TCP decreasing the window size when a packet is lost.

It is important to note that the BECN and FECN bits are set in normal data frames, and are not explicit frames generated by the Frame Relay switch. This makes it theoretically possible that a router will never receive a frame with the BECN bit set if the remote router never sends data to it. A realistic example would be where a DLCI is used exclusively to send multicast traffic. In this case, the vast majority of frames will be in one direction, source toward the receivers. If congestion occurs the Frame Relay switch will start marking frames. The majority of the frames will of course be marked in the opposite direction of the congestion as most traffic will flow from the source toward the receivers. The **frame-relay fecn-adapt** map-class command is useful in this type of situation as the receiving router can generate a frame with the BECN bit set upon receipt of a frame with the FECN bit set. This will allow the source of the congestion to throttle its sending rate down if the **frame-relay adaptive-shaping becn** map-class command is configured on the router causing the congestion.

Task 8.1 Verification

Verify the FRTS configuration in details:

```
Rack1R5#show frame-relay pvc 502

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

DLCI = 502, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0.245

<output omitted>
  Shaping adapts to BECN
  pvc create time 05:11:19, last time pvc status changed 03:33:07
  cir 256000    bc 32000     be 0        byte limit 4000   interval 125
  mincir 192000    byte increment 4000  Adaptive Shaping BECN
  pkts 5         bytes 442       pkts delayed 0         bytes delayed 0
<output omitted>
```

Task 8.2

R4:

```
class-map match-all HTTP_CLASS
 match access-group 150
!
policy-map HTTP_POLICY
  class HTTP_CLASS
   police cir 256000
!
interface FastEthernet0/1
 service-policy output HTTP_POLICY
!
access-list 150 permit tcp any eq www any time-range HTTP_TIMERANGE
!
time-range HTTP_TIMERANGE
 periodic weekdays 8:00 to 17:00
```


Task 8.2 Breakdown

In this task, a common mistake made is to configure the access-list incorrectly. Make sure that you closely read the wording of tasks in regards to the direction of the traffic flow when configuring access-lists. This task mentioned that HTTP responses send out R4's interface Fa0/1 be limited. This means that the below access-list would be incorrect, as it would match HTTP packets destined to a web server in VLAN 44.

```
access-list 150 permit tcp any any eq www time-range HTTP_TIMERANGE
```

The access-list needs to match the HTTP server's responses. This is why the access-list is configured as shown below.

```
access-list 150 permit tcp any eq www any time-range HTTP_TIMERANGE
```

The **time-range** option of the extended access-list allows for selective filtering based on the clock of the local router. Time based access-lists are created by first defining the **time-range** in global configuration mode. The keywords **absolute** and **periodic** determine whether the event will occur at one specific (absolute) time, or will recur at a certain (periodic) interval.

Once the local time is within the specified time-range, the access-list entry or entries which reference the time-range are active. When the time range is inactive, it is as if the access-list entries do not exist.

There are various methods that can be used to limit traffic, including policing, shaping, and even legacy CAR using the rate-limit command on the interface. If there are no section restrictions, then any method that limits the traffic could be used.

Task 8.2 Verification

```
Rack1R4#clock set 00:00:00 1 Jan 2004
Rack1R4#show clock
00:00:01.322 UTC Thu Jan 1 2004
Rack1R4#show access-lists
Extended IP access list 150
    10 permit tcp any eq www any time-range HTTP_TIMERANGE (inactive)
                                                          ↗
              time-range is inactive on a Thursday at 12am

Rack1R4#clock set 10:00:00 1 Jan 2004
Rack1R4#show clock
10:00:03.069 UTC Thu Jan 1 2004
R4#show access-lists
Extended IP access list 150
    10 permit tcp any eq www any time-range HTTP_TIMERANGE (active)
                                                          ↗
                time-range is active on a Thursday at 10am

Rack1R4#clock set 10:00:00 3 Jan 2004
Rack1R4#show clock
10:00:02.480 UTC Sat Jan 3 2004
Rack1R4#show access-lists
Extended IP access list 150
    10 permit tcp any eq www any time-range HTTP_TIMERANGE (inactive)
                                                          ↗
          time-range is inactive at the same time on a Saturday
```

Task 8.3

R4 and R5:

```
map-class frame-relay FRTS
 frame-relay fair-queue
```

R4:

```
interface Serial0/0/0
 ip rsvp bandwidth 128 64
```

R5:

```
interface Serial0/0/0
ip rsvp bandwidth 128 64
!
interface Serial0/0.245 multipoint
 ip rsvp bandwidth 128 64
```

Task 8.3 Breakdown

Resource Reservation Protocol (RSVP) is used to dynamically request specific QoS from the network for a particular data flow. A data flow is defined as sequence of packets that have the same QoS requirements and have the same source and destination. Note that the destination could possibly be more than one host in the case of IP multicast. RSVP requests will normally result in resources being reserved by each router along the path between the source and destination.

There are three possible requests that R4 can make for its VoIP traffic. The first is best effort. Best effort does just what is says, supplies a best effort QoS policy for particular data flow. Best effort is commonly used with general application traffic. The second possibility is controlled load. Controlled load is used for rate sensitive traffic. Rate sensitive traffic will be guaranteed bandwidth (rate) through RSVP. The third possibility is guaranteed delay. Guaranteed delay is used to help ensure a minimum amount of jitter. Delay is normally one of the more important QoS requirements in relation to VoIP.

Although this is a basic configuration in regards to RSVP, it is important to note that when using subinterfaces, the **ip rsvp bandwidth** command will need to be applied to the physical interface along with the subinterface. If more than one subinterface is using RSVP, the physical interface's **ip rsvp bandwidth command** will be the sum of all the subinterface's **ip rsvp bandwidth** commands.

Task 8.3 Verification

Verify per-VC queueing (note the WFQ and the reserved conversations):

```
Rack1R5#show frame-relay pvc 504

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

DLCI = 504, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0.245
<output omitted>

  Queueing strategy: weighted fair
  Current fair queue configuration:
   Discard     Dynamic      Reserved
   threshold   queue count  queue count
    64          16           4
  Output queue size 0/max total 600/drops 0
```

Check RSVP resources:

```
Rack1R5#show ip rsvp interface s0/0
interface    allocated  i/f max  flow max sub max
Se0/0        0          128K     64K      0

Rack1R5#show ip rsvp interface s0/0.245
interface    allocated  i/f max  flow max sub max
Se0/0.245    0          128K     64K      0
```

To finish with RSVP verification, simulate RSVP sender and reservation hosts:

R5:

```
ip rsvp reservation-host 150.1.5.5 150.1.4.4 UDP 5000 4000 FF RATE 32 4
```

R4:

```
ip rsvp sender-host 150.1.5.5 150.1.4.4 UDP 5000 4000 32 4
```

Verify that the RSVP reservation is installed:

```
Rack1R4#show ip rsvp reservation
To         From       Pro DPort Sport Next Hop     I/F      Fi Serv BPS
150.1.5.5  150.1.4.4  UDP 5000  4000  136.1.245.5  Se0/0    FF RATE 32K

Rack1R4#show ip rsvp installed
RSVP: Serial0/0/0
BPS    To        From         Protoc DPort  Sport  Weight Conversation
32K    150.1.5.5 150.1.4.4    UDP    5000   4000   6      25

Rack1R4#show ip rsvp interface
interface    allocated  i/f max  flow max sub max
Se0/0        32K        128K     64K      0
```


☠ Pitfall

Weighted Fair Queuing (WFQ) needs to be enabled for RSVP. WFQ is normally enabled by default on Serial interfaces (2.048 Mbps and below). Once Frame Relay Traffic Shaping is enabled, WFQ is disabled.

```
Rack1R1#show run interface s1/0
Building configuration...

Current configuration : 113 bytes
!
interface Serial1/0
 no ip address
 encapsulation frame-relay
end

Rack1R1#show queueing interface s1/0
Interface Serial1/0 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/1/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 96 kilobits/sec

Rack1R1(config)#int s1/0
Rack1R1(config-if)#frame-relay traffic-shaping
Rack1R1(config-if)#^Z
Rack1R1#show queueing interface s1/0
Interface Serial1/0 queueing strategy: none
Rack1R1#show run interface s1/0
Building configuration...

Current configuration : 113 bytes
!
interface Serial1/0
 no ip address
 encapsulation frame-relay
 no fair-queue
 frame-relay traffic-shaping
end

Rack1R1#
```

