Task 1.1

```
SW1:
interface FastEthernet 0/13
 switchport trunk pruning vlan 2-7,9-1001

SW3:
interface FastEthernet 0/16
 switchport trunk pruning vlan 2-7,9-1001
```

Task 1.1 Breakdown

As previously mentioned, when a trunk is created all VLANs throughout the VTP domain may transit the trunk link. These VLANs are said to be in the *allowed vlan* list. In the same manner, when VTP pruning is enabled, all non-default VLANs can be pruned off of a trunk link (default VLANs such as 1 and 1002-1005 cannot be pruned). These VLANs are said to be *prune eligible*. In certain cases, such as when dealing with switches in transparent mode, it is not desirable to have a switch send pruning information out a specific trunk link. Since pruning can only be enabled or disabled globally, manually editing the prune eligible list is the only way to achieve the desired effect.

To edit the prune eligible list, use the interface level command **`switchport trunk pruning vlan [add | remove | none | except] [num]`**. To verify what is prune eligible on an interface, issue the **`show interface [int] switchport`** command. By default VLANs 2-1001 are prune eligible.

Task 1.1 Verification

```
Rack1SW1#show interface fa0/13 switchport | include Pruning
Pruning VLANs Enabled: 2-1001  ← Prune eligible list

Rack1SW1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1SW1(config)#interface fa0/13
Rack1SW1(config-if)#switchport trunk pruning vlan 2-7,9-1001
                                                      ↑
                        Delete vlan 8 from the prune eligible list
Rack1SW1(config)#^Z
Rack1SW1#show interface fa0/13 switchport | include Pruning
Pruning VLANs Enabled: 2-7,9-1001 ← VLAN 8 can no longer be pruned
```

Task 1.2

```
SW1:
spanning-tree vlan 258 root primary

SW3:
spanning-tree vlan 258 root secondary
```

```
SW2:
interface range Fa0/13 - 15
 spanning-tree vlan 258 cost 100
```

Task 1.2 Breakdown

As previously discussed, the two user defined variables that can be used to affect the spanning-tree root port selection are port-cost and port-priority. The above task specifies to “use the fewest number of commands to accomplish this task and do not alter SW1’s port-priority.” Since SW1 is the root of the spanning-tree, the appropriate value to change is the spanning tree cost for VLAN 258 on SW2 .

Note

To affect how the local switch elects its root port change the spanning-tree port-cost. Cost is cumulative throughout the STP domain.

To affect how a downstream switch elects its root port, change the spanning-tree port-priority. Port-priority is only locally significant between two directly connected bridges.

Task 1.2 Verification

```
Rack1SW2#show spanning-tree vlan 258

VLAN0258
  Spanning tree enabled protocol ieee
  Root ID    Priority    24834
             Address     000a.f4f3.e780
             Cost        10
             Port        18 (FastEthernet0/16) ← Root port
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

<output deleted>

Interface          Role Sts Cost      Prio.Nbr Type
----------------  ---- --- --------- -------- -----------------
Fa0/2            Desg FWD 19        128.4    P2p
Fa0/13           Altn BLK 100       128.15   P2p
Fa0/14           Altn BLK 100       128.16   P2p
Fa0/15           Altn BLK 100       128.17   P2p
Fa0/16           Root FWD 19        128.18   P2p
Fa0/17           Altn BLK 19        128.19   P2p
Fa0/18           Altn BLK 19        128.20   P2p
```

Task 1.3

SW1:
```
interface FastEthernet0/15
 udld port aggressive
 spanning-tree guard loop
```

SW2:
```
interface FastEthernet0/15
 udld port aggressive
```

Task 1.3 Breakdown

Unidirectional Link Detection (UDLD) is used to detect when the send channel of a cable is down, but not the receive channel, and vice versa. This situation typically can occur in a fiber optic cable when there is a break in one side of the cable run. When UDLD detects this situation, the interface is brought down and a log message is generated. This feature is useful to prevent against spanning-tree loops and traffic black holes due to unidirectional links. To enable UDLD on fiber optic interfaces, issue the *global* configuration command **`udld enable`** or **`udld aggressive`**. To enable UDLD on a copper interface, issue the *interface* command **`udld port aggressive`**.

In certain cases, a spanning-tree loop can occur when the send channel of a designated port is damaged. This will cause the bridge on the other side of the link to stop receiving STP BPDUs. When this occurs, the non-designated port assumes that it should become the designated port, and can eventually result in a loop in the topology. In order to prevent this case, spanning-tree loopguard will transition the non-designated port to loop-inconsistent state, and will not pass user traffic when this problem occurs. To enable loopguard, use the interface level command **`spanning-tree guard loop.`**

☠ **Pitfall**

The global command **`udld enable`** only applies to fiber interfaces. Ensure to use the *interface* command **`udld port aggressive`** for copper interfaces.

Task 1.3 Verification

Verify the UDLD configuration:

```
Rack1SW2#show udld fa0/15

Interface Fa0/15
---
Port enable administrative configuration setting: Enabled / in
aggressive mode
Port enable operational state: Enabled / in aggressive mode
Current bidirectional state: Bidirectional
Current operational state: Advertisement - Single neighbor detected
Message interval: 7
Time out interval: 5
<output omitted>
```

Verify loop guard on SW1 Fa0/15:

```
Rack1SW1#show spanning-tree interface fa0/15 detail
<output omitted>
   Link type is point-to-point by default
   Loop guard is enabled on the port
   BPDU: sent 1234, received 21
<output omitted>
```

Copyright © 2009 Internetwork Expert www.INE.com

Task 1.4

SW4:
```
interface FastEthernet0/21
 spanning-tree vlan 258 port-priority 16
```

Task 1.4 Breakdown

As mentioned earlier, the two common methods to affect the spanning-tree path to the root are cost and port-priority. The key to remembering which one to use where is to understand the direction these two options affect in regards to spanning tree. When going away from the root of the tree, use port-priority. When going towards the root of the tree, use cost.

Task 1.4 Verification

```
Rack1SW3#show spanning-tree vlan 258 | in Fa0/21
Fa0/21           Root FWD 19        128.21   P2p
```

Task 1.5

SW1:
```
interface FastEthernet0/1
 storm-control unicast level pps 250
```

Task 1.5 Breakdown

Storm control limits the amount of unicast, multicast, or broadcast traffic that is received on a layer 2 switchport. When the threshold of unicast or broadcast traffic is exceeded, traffic in excess of the threshold is dropped. When the multicast threshold is exceeded, all unicast, multicast, or broadcast traffic is dropped until the level falls below the threshold. To configure storm-control, issue the **`storm-control [unicast | broadcast | multicast] level [level]`** interface level command.

The 3550 and 3560 both support using the PPS (Packets Per Second) option with the **`storm-control`** command, but the 3550 does not support the BPS (Bits Per Second) option.

☠ Pitfall

Do not assume that the task title will directly indicate the solution. In this case, the title of the task is “Rate-Limiting”, but the solution used is storm-control.

Task 1.5 Verification

```
Rack1SW1#show storm-control unicast
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------
Fa0/1      Forwarding        250 pps      250 pps        0 pps
```

Task 1.6

```
SW2:
mls qos map ip-prec-dscp 0 0 0 0 32 40 0 0
```

Task 1.6 Breakdown

The switches use internal DSCP mappings for classification and marking of frames as they traverse the switch. Mappings are used to help determine how the switch will handle the frame. To alter the default mappings between the internal DSCP and the IP precedence, the **`mls qos map ip-prec-dscp <8 DSCP values>`** command is used.

Task 1.6 Verification

```
Rack1SW2#show mls qos maps ip-prec-dscp
   IpPrecedence-dscp map:
     ipprec:   0  1  2  3  4  5  6  7
     --------------------------------
       dscp:   0  0  0  0 32 40  0  0
```

Task 1.7

```
SW2:
mls qos
interface FastEthernet0/2
 mls qos trust ip-precedence
```

Task 1.7 Breakdown

By configuring the interface to trust IP precedence, the IP precedence to DCSP map created in the previous task will be used to map ingress packet’s IP precedence values to the internal DSCP values. If you forget to enable QoS with the command **mls qos**, or your output may look like this:

```
Rack1SW2#show mls qos interface fa0/2
QoS is disabled. When QoS is enabled, following settings will be
applied
trust state: trust ip-precedence
trust mode: trust ip-precedence
trust enabled flag: ena
COS override: dis
default COS: 0
DSCP Mutation Map: Default DSCP Mutation Map
Trust device: none
qos mode: port-based
```

Task 1.7 Verification

```
Rack1SW2#show mls qos interface fa0/2 | include ip-precedence
trust state: trust ip-precedence
trust mode: trust ip-precedence
```

Task 2.1

```
R2:
router ospf 1
 area 1 nssa no-summary

R5:
router ospf 1
 area 1 nssa no-summary

SW2:
router ospf 1
 area 1 nssa
```

Task 2.1 Verification

Verify the area configuration and the translated prefixes:

```
Rack1R2#show ip ospf | begin Area 1
    Area 1
        Number of interfaces in this area is 1
        It is a NSSA area
        Perform type-7/type-5 LSA translation
<output omitted>

Rack1R2#show ip route ospf | include 150.1.8.0
O N2    150.1.8.0/24 [110/20] via 141.1.0.8, 00:23:07, FastEthernet0/0

Rack1R1#show ip route ospf | include 150.1.8.0
O E2    150.1.8.0/24 [110/20] via 141.1.123.2, 00:27:24, Serial0/0.1
```

Task 2.2

```
R2:
interface Tunnel0
 ip address 141.1.25.2 255.255.255.0
 tunnel source FastEthernet0/0
 tunnel destination 141.1.0.5
!
router ospf 1
 network 141.1.25.2 0.0.0.0 area 0

R3:
router ospf 1
 redistribute rip subnets
!
router rip
 redistribute ospf 1 metric 1
```

☞ **Quick Note**
A Virtual-Link cannot be created over a stub area

```
R4:
interface Serial0/0/0
 ip ospf network point-to-point
!
router ospf 1
 router-id 150.1.4.4
 network 141.1.45.4 0.0.0.0 area 2
 network 141.1.54.4 0.0.0.0 area 2
 network 141.1.145.4 0.0.0.0 area 2
 network 150.1.4.4 0.0.0.0 area 2

R5:
interface Serial0/0/0
 ip ospf network point-to-point
!
interface Tunnel0
 ip address 141.1.25.5 255.255.255.0
 tunnel source FastEthernet0/1
 tunnel destination 141.1.0.2
!
router ospf 1
 router-id 150.1.5.5
 network 141.1.25.5 0.0.0.0 area 0
 network 141.1.45.5 0.0.0.0 area 2
 network 141.1.54.5 0.0.0.0 area 2
 network 141.1.145.5 0.0.0.0 area 2
 network 150.1.5.5 0.0.0.0 area 2
```

☞ **Quick Note**
A Virtual-Link cannot be created over a stub area

Task 2.2 Breakdown

In order to properly compute the shortest path first (SPF) algorithm, routers within a link-state area must have a consistent view of the link state topology. For this reason, link-state protocols such as OSPF and IS-IS do not support the removal of a link state advertisement (LSA) from the link-state database on a per router basis. Instead, this must be done on a per link-state area basis. In OSPF, this is accomplished by the various stub area definitions.

By preventing certain types of LSAs from entering an area, the various stub area types can be used to reduce the amount of forwarding information required to be in both the OSPF database and the IP routing table. Such cases may be advantageous when there is only one exit point out of an area, or only one exit point out of the autonomous system. In such a design, it may be feasible to replace specific forwarding information with default information, hence reducing memory utilization and speeding up the routing table lookup process. There are four OSPF stub area definitions. These are stub, totally stubby, not-so-stubby (NSSA), and not-so-totally-stubby.

To understand why certain LSAs are removed from an area, you must first understand what each LSA type accomplishes. LSA types are defined as follows:

LSA	Name	Description
1	Router LSA	Generated by all routers in an area to describe their directly attached links (intra-area routes). Does not leave the area.
2	Network LSA	Generated by the DR of a broadcast or non-broadcast segment to describe the neighbors connected to that segment. Does not leave the area.
3	Summary LSA	Generated by the area border router (ABR) to describe a route to neighbors outside the area (inter-area route).
4	Summary LSA	Generated by the ABR to describe a route to an autonomous system border router (ASBR) to neighbors outside the area.
5	External LSA	Generated by the ASBR to describe routes redistributed into the area. These routes appear as E1 or E2 in the IP routing table. E2 (default) uses a static cost throughout the OSPF domain, as it only takes the cost into account that is reported at redistribution. E1 uses a cumulative cost of the cost reported into the OSPF domain at redistribution plus the local cost to the ASBR.
6	Multicast LSA	Used in multicast OSPF. Not supported by Cisco.
7	NSSA External LSA	Generated by an ASBR inside a not-so-stubby (NSSA) area to describe routes redistributed into the NSSA area. LSA 7 is translated into LSA 5 as it leaves the NSSA area. These routes appear as N1 or N2 in the IP routing table inside the NSSA area. Much like LSA 5, N2 is a static cost while N1 is a cumulative cost that includes the cost up to the ASBR.

A stub area blocks OSPF external routes (LSAs 4 & 5) from entering the area. The ABR of a stub area automatically generates a default route (LSA 3) into the stub area. A stub area is defined by issuing the **`area [area_id] stub`** routing process subcommand on all devices in the stub area.

A totally stubby area is a stub area that in addition to blocking OSPF external routes blocks OSPF inter-area routes (LSA 3). The ABR of a totally stubby area automatically generates a default route (LSA 3) into the stub area. Redistribution into stub and totally stubby areas is not permitted. A totally stubby area is defined by issuing the **`area [area_id] stub no-summary`** routing process subcommand on all ABRs of the stub area.

The not-so-stubby area (NSSA) overcomes the problem of not being able to redistribute into a stub area. Like a stub area, a not-so-stubby area blocks OSPF external routes (LSAs 4 & 5) from entering the area. However, redistribution is allowed into the NSSA area. These routes are redistributed as NSSA external (LSA 7) and are different than normal LSA 5 external routes. As these LSA 7 prefixes leave the NSSA area, the ABR translates them into LSA 5. In other words, routers outside the NSSA area do not know that these routes were redistributed into an NSSA area, but instead simply see them as LSA 5 external routes. A not-so-stubby area is defined by issuing the **`area [area_id] nssa`** routing process subcommand on all routers in the stub area.

Another difference between the stub area and the not-so-stubby area is that the ABR of the NSSA does not automatically originate a default route into the area. A default route may be originated into an NSSA by adding the **`default-originate`** keyword onto the **`area [area_id] nssa`** statement. This default is type 7 LSA.

The not-so-totally-stubby area combines the concept of the totally stubby area and the not-so-stubby area. The not-so-totally-stubby area blocks both OSPF external (LSA 5) and inter-area (LSA 3 & 4) routes from entering the area. The ABR of the not-so-totally-stubby area automatically generates a default route (LSA 3) into the not-so-totally-stubby area. Redistribution into the not-so-totally-stubby area is permitted. A not-so-totally-stubby area is defined by issuing the **`area [area_id] nssa no-summary`** routing process subcommand on all ABRs in the stub area.

Note

All routers in a stub or not-so-stub are must agree on the stub or NSSA flag. It is the ABR(s) of the stub area or NSSA area that determine if it is totally stubby or not-so-totally stubby by adding the **no-summary** keyword on to the appropriate stub command.

The stub area types can be summarized as follows:

Stub Type	Keyword	LSAs	Default Injected
Stub	area x stub	1,2,3	YES
Totally Stubby	area x stub no-summary	1,2, default of 3	YES
Not-So-Stub	area x nssa	1,2,3,7	NO
Not-So-Totally-Stubby	area x nssa no-summary	1,2, default of 3, 7	YES

☠ **Pitfall**

A stub area cannot be used as a transit area for a virtual-link. In the previous task, a GRE tunnel configured with OSPF area 0 was created between R2 and R5. This is due to the fact that OSPF area 2 is discontiguous from OSPF area 0. Typically this problem is fixed by creating a virtual-link back to connect area 0 with the discontiguous area. However, since in this case area 1 (the transit area) is stub, this method will not work. Therefore a virtual connection (GRE tunnel) is created between R2 and R5 to run OSPF area 0.

Task 2.2 Verification

Verify that tunnel is up and running in OSPF area 0:

```
Rack1R5#show interfaces tu0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 141.1.25.5/24
<output omitted>

Rack1R5#show ip ospf interface tu0
Tunnel0 is up, line protocol is up
  Internet Address 141.1.25.5/24, Area 0
  Process ID 1, Router ID 150.1.5.5, Network Type POINT_TO_POINT, Cost:
11111
  Transmit Delay is 1 sec, State POINT_TO_POINT,
<output omitted>

Rack1R5#ping 141.1.25.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.25.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/10/36 ms
```

Next verify OSPF neighbors on the tunnel:

```
Rack1R5#show ip ospf neighbor tu0

Neighbor ID  Pri   State        Dead Time   Address         Interface
150.1.2.2      0   FULL/  -     00:00:38    141.1.25.2      Tunnel0
```

Finally verify that we are seeing R5 and R2 Loopback0 prefixes as inter-area summary prefixes on R1:

```
Rack1R1#show ip route ospf  | include (150.1.4.4|150.1.5.5)
O IA    150.1.5.5/32 [110/11176] via 141.1.123.2, 01:44:25, Serial0/0.1
O IA    150.1.4.4/32 [110/11186] via 141.1.123.2, 01:44:19, Serial0/0.1
```

Task 2.3

```
R6:
router rip
 network 150.1.0.0
 no passive-interface Serial0/0/0
 offset-list 1 in 9 Serial0/0/0
 network 54.0.0.0
!
access-list 1 permit 0.0.0.0 255.255.254.255
```

Task 3.1

R1:

```
ipv6 unicast-routing
!
interface Serial0/0.1 point-to-point
 ipv6 address 2001:141:1:12::1/64
 ipv6 address FE80::1 link-local
 ipv6 ospf 1 area 0
```

R2:

```
ipv6 unicast-routing
!
interface FastEthernet0/0
 ipv6 address 2001:141:1:25::2/64
 ipv6 ospf 1 area 1
!
interface Serial0/0
 ipv6 address 2001:141:1:12::2/64
 ipv6 address FE80::2 link-local
 ipv6 ospf network point-to-point
 ipv6 ospf 1 area 0
 frame-relay map ipv6 FE80::1 201
 frame-relay map ipv6 2001:141:1:12::1 201 broadcast
```

R5:

```
ipv6 unicast-routing
!
interface FastEthernet0/1
 ipv6 address 2001:141:1:25::5/64
 ipv6 ospf 1 area 1
```

SW2:

```
sdm prefer dual-ipv4-and-ipv6 default
!
ipv6 unicast-routing
!
interface VLAN258
 ipv6 address 2001:141:1:25::A/64
 ipv6 ospf 1 area 1
```

☞ **Quick Note**
A reload is required for the new SDM template to take effect. Until then, IPv6 routing is not supported on the 3560s

Task 3.1 Breakdown

Note

Output shown in this breakdown is general information regarding OSPFv3, and is not specific to this lab. For output specific to this lab, see the verification section.

The first step in enabling OSPF for IPv6 (OSPFv3) is to enable IPv6 routing globally with the **`ipv6 unicast-routing`** command, followed by the **`ipv6 router ospf 1 [process-id]`** command, where *process-id* is a locally significant number similar to OSPFv2 for IPv4. Unlike OSPFv2, OSPFv3 does not use the network statement to enable the process on an interface. Instead, the interface level command **`ipv6 ospf [process-id] area [area-id]`** command is used. Once OSPFv3 has been configured, issue the **`show ipv6 ospf neighbor`** command to verify adjacency status.

```
Rack1R1#show ipv6 ospf neighbor

Neighbor ID   Pri  State          Dead Time   Interface ID    Interface
222.22.2.1    1    FULL/DR        00:00:30    12
FastEthernet0/0
```

In the above output, we can see that R1 has formed an adjacency with an OSPFv3 router with the router-id 222.22.2.1. This router has a priority of 1 and has been elected the designated router. Although the above output relates specifically to IPv6 routing, the OSPFv3 router-id still uses the 32-bit dotted decimal format as used in OSPFv2. To get more detailed information about interfaces running OSPFv3 issue the **`show ipv6 ospf interface`** command.

```
Rack1R1#show ipv6 ospf interface
Ethernet0/0 is up, line protocol is up
  Link Local Address FE80::2D0:58FF:FE6E:B720, Interface ID 3
  Area 0, Process ID 1, Instance ID 0, Router ID 150.1.1.1
  Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 222.22.2.1, local address
FE80::205:5EFF:FE0F:B8E0
  Backup Designated router (ID) 150.1.1.1, local address
FE80::2D0:58FF:FE6E:B720
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
<snip>
```

From the above output, we can see more detailed information about the OSPF adjacency that has formed on interface FastEthernet0/0, such as the area, process-id, local router-id, network type, interface cost, local state (DR, BDR, or DROTHER), local priority, router-id of the DR and BDR, link-local address of the DR and BDR, and interface timers.

```
Rack1R1#show ipv6 route ospf
IPv6 Routing Table - 7 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
    O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
OE2  2001:51:51:51::/64 [110/20]
     via FE80::205:5EFF:FE0F:B8E0, FastEthernet0/0
```

Note that when the routing table is examined, the next-hop address for OSPFv3 learned routes is the remote link-local address of the advertising router. Since the FastEthernet interface used for this adjacency is a broadcast media, ICMPv6 neighbor discovery will automatically resolve the remote link-local IPv6 address to the remote layer 2 (MAC) address. This can be verified by issuing the **show ipv6 neighbors** command in global configuration mode.

```
Rack1R1#show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
FE80::205:5EFF:FE0F:B8E0                   17 0005.5e0f.b8e0  STALE Et0/0
2001:192:10:1::254                         17 0005.5e0f.b8e0  STALE Et0/0
```

Had the interface used for adjacency been a multipoint non-broadcast interface, such as the main interface in Frame Relay or ATM, an explicit layer 3 to layer 2 resolution statement would have been required for the remote link-local IPv6 address.

Task 3.1 Verification

```
Rack1R2#show ipv6 ospf neighbor

Neighbor ID Pri  State         Dead Time  Interface ID  Interface
150.1.1.1    1   FULL/  -      00:00:38   8             Serial0/0
150.1.8.8    1   FULL/DROTHER  00:00:39   2326          FastEthernet0/0
150.1.5.5    1   FULL/DR       00:00:38   6             FastEthernet0/0
```

Task 3.2

```
R2:
ipv6 router ospf 1
 area 1 range 2001:150:1::/60

R5:
interface Loopback100
 ipv6 address 2001:150:1:5::5/64
 ipv6 ospf 1 area 1

SW2:
interface Loopback100
 ipv6 address 2001:150:1:8::8/64
 ipv6 ospf 1 area 1
```

Task 3.2 Breakdown

Similar to OSPFv2, OSPFv3 supports both internal and external summarization. Internal summarization occurs as LSAs are moving between areas, and is configured with the interface level command **`area [area-id] range [summary]`**, where *area-id* is the area you are summarizing from and *summary* is the summary prefix. For external summaries, the process level command **`summary-prefix`** is used.

Task 3.2 Verification

Verify that the summary is generated:

```
Rack1R1#show ipv6 route ospf
<snip>
OI  2001:141:1:25::/64 [110/65]
     via FE80::2, Serial0/0.1
OI  2001:150:1::/60 [110/65]
     via FE80::2, Serial0/0.1
```

Task 4.1

```
R4:
router rip
 version 2
 no auto-summary
 address-family ipv4 vrf VPN_A
 network 204.12.1.0
 no passive-interface FastEthernet 0/1


R6:
router rip
 version 2
 no auto-summary
 address-family ipv4 vrf VPN_A
 network 54.0.0.0
 no passive-interface Serial 0/0/0

!
access-list 1 permit 0.0.0.0 255.255.254.255
```

Task 4.1 Verification

```
Rack1R4#show ip route vrf VPN_A

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

Gateway of last resort is not set

C    204.12.1.0/24 is directly connected, FastEthernet0/1
     31.0.0.0/16 is subnetted, 4 subnets
R       31.3.0.0 [120/1] via 204.12.1.254, 00:00:18, FastEthernet0/1
R       31.2.0.0 [120/1] via 204.12.1.254, 00:00:18, FastEthernet0/1
R       31.1.0.0 [120/1] via 204.12.1.254, 00:00:18, FastEthernet0/1
R       31.0.0.0 [120/1] via 204.12.1.254, 00:00:18, FastEthernet0/1
     30.0.0.0/16 is subnetted, 4 subnets
R       30.2.0.0 [120/1] via 204.12.1.254, 00:00:18, FastEthernet0/1
R       30.3.0.0 [120/1] via 204.12.1.254, 00:00:18, FastEthernet0/1
R       30.0.0.0 [120/1] via 204.12.1.254, 00:00:18, FastEthernet0/1
R       30.1.0.0 [120/1] via 204.12.1.254, 00:00:18, FastEthernet0/1

Rack1R6#show ip route vrf VPN_A

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

Gateway of last resort is not set

     54.0.0.0/24 is subnetted, 1 subnets
C       54.1.1.0 is directly connected, Serial0/0/0
R    212.18.1.0/24 [120/1] via 54.1.1.254, 00:00:17, Serial0/0/0
R    212.18.0.0/24 [120/10] via 54.1.1.254, 00:00:17, Serial0/0/0
R    212.18.3.0/24 [120/1] via 54.1.1.254, 00:00:17, Serial0/0/0
R    212.18.2.0/24 [120/10] via 54.1.1.254, 00:00:17, Serial0/0/0
```

Task 4.2

R4:

```
interface Tunnel 46
 ip address 141.1.46.4 255.255.255.0
 tunnel source Loopback0
 tunnel destination 150.1.6.6
 mpls ip
!
ip route 150.1.66.66 255.255.255.255 Tunnel 46
!
interface Loopback100
 ip address 150.1.44.44 255.255.255.255
!
router bgp 400
 neighbor 150.1.66.66 remote-as 100
 neighbor 150.1.66.66 update-source Loopback100
 neighbor 150.1.66.66 ebgp-multihop
!
address-family ipv4 vrf VPN_A
 redistribute rip
address-family ipv4
 no neighbor 150.1.66.66 activate
address-family vpnv4 unicast
 neighbor 150.1.66.66 activate
 neighbor 150.1.66.66 send-community both
!
router rip
 address-family ipv4 vrf VPN_A
  redistribute bgp 400 metric transparent
```

R6:

```
interface Tunnel 46
 ip address 141.1.46.6 255.255.255.0
 tunnel source Loopback0
 tunnel destination 150.1.4.4
 mpls ip
!
ip route 150.1.44.44 255.255.255.255 Tunnel 46
!
```

```
interface Loopback100
 ip address 150.1.66.66 255.255.255.255
!
router bgp 100
 neighbor 150.1.44.44 remote-as 400
 neighbor 150.1.44.44 update-source Loopback100
 neighbor 150.1.44.44 ebgp-multihop
!
address-family ipv4 vrf VPN_A
 redistribute rip
address-family ipv4
 no neighbor 150.1.44.44 activate
address-family vpnv4 unicast
 neighbor 150.1.44.44 activate
 neighbor 150.1.44.44 send-community both
!
router rip
 address-family ipv4 vrf VPN_A
  redistribute bgp 100 metric transparent
```

Task 4.2 Verification

```
Rack1R6#show mpls ldp neighbor
    Peer LDP Ident: 150.1.44.44:0; Local LDP Ident 150.1.66.66:0
        TCP connection: 150.1.44.44.646 - 150.1.66.66.55164
        State: Oper; Msgs sent/rcvd: 38/39; Downstream
        Up time: 00:09:44
        LDP discovery sources:
          Tunnel46, Src IP addr: 141.1.46.4
        Addresses bound to peer LDP Ident:
          141.1.145.4     141.1.54.4      141.1.45.4      150.1.4.4
          150.1.44.44     141.1.46.4

Rack1R6#show bgp vpnv4 unicast vrf VPN_A
BGP table version is 34, 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
Route Distinguisher: 100:1 (default for vrf VPN_A)
*> 30.0.0.0/16      150.1.44.44              1             0 400 ?
*> 30.1.0.0/16      150.1.44.44              1             0 400 ?
*> 30.2.0.0/16      150.1.44.44              1             0 400 ?
*> 30.3.0.0/16      150.1.44.44              1             0 400 ?
*> 31.0.0.0/16      150.1.44.44              1             0 400 ?
*> 31.1.0.0/16      150.1.44.44              1             0 400 ?
*> 31.2.0.0/16      150.1.44.44              1             0 400 ?
*> 31.3.0.0/16      150.1.44.44              1             0 400 ?
*> 54.1.1.0/24      0.0.0.0                  0         32768 ?
*> 204.12.1.0       150.1.44.44              0             0 400 ?
*> 212.18.0.0       54.1.1.254              10         32768 ?
*> 212.18.1.0       54.1.1.254               1         32768 ?
*> 212.18.2.0       54.1.1.254              10         32768 ?
*> 212.18.3.0       54.1.1.254               1         32768 ?
```

Check that the routes learned via VPN connection are being advertised. Notice that split-horizon is disabled by default with RIP on Frame-Relay interfaces, thus R6 re-adverise the routes learned from BB1. Pay attention to the metric values, learned from BGP MED attribute for RIP prefixes.

```
Rack1R6#debug ip rip
RIP protocol debugging is on
RIP: sending v2 update to 224.0.0.9 via Serial0/0/0 (54.1.1.6)
RIP: build update entrie
        30.0.0.0/16 via 0.0.0.0, metric 2, tag 0
        30.1.0.0/16 via 0.0.0.0, metric 2, tag 0
        30.2.0.0/16 via 0.0.0.0, metric 2, tag 0
        30.3.0.0/16 via 0.0.0.0, metric 2, tag 0
        31.0.0.0/16 via 0.0.0.0, metric 2, tag 0
        31.1.0.0/16 via 0.0.0.0, metric 2, tag 0
        31.2.0.0/16 via 0.0.0.0, metric 2, tag 0
        31.3.0.0/16 via 0.0.0.0, metric 2, tag 0
        204.12.1.0/24 via 0.0.0.0, metric 1, tag 0
        212.18.0.0/24 via 54.1.1.254, metric 11, tag 0
        212.18.1.0/24 via 54.1.1.254, metric 2, tag 0
        212.18.2.0/24 via 54.1.1.254, metric 11, tag 0

Rack1R4#show bgp vpnv4 unicast vrf VPN_A
BGP table version is 38, 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
Route Distinguisher: 100:1 (default for vrf VPN_A)
*> 30.0.0.0/16      204.12.1.254             1         32768 ?
*> 30.1.0.0/16      204.12.1.254             1         32768 ?
*> 30.2.0.0/16      204.12.1.254             1         32768 ?
*> 30.3.0.0/16      204.12.1.254             1         32768 ?
*> 31.0.0.0/16      204.12.1.254             1         32768 ?
*> 31.1.0.0/16      204.12.1.254             1         32768 ?
*> 31.2.0.0/16      204.12.1.254             1         32768 ?
*> 31.3.0.0/16      204.12.1.254             1         32768 ?
*> 54.1.1.0/24      150.1.66.66              0             0 100 ?
*> 204.12.1.0       0.0.0.0                  0         32768 ?
*> 212.18.0.0       150.1.66.66             10             0 100 ?
*> 212.18.1.0       150.1.66.66              1             0 100 ?
*> 212.18.2.0       150.1.66.66             10             0 100 ?
*> 212.18.3.0       150.1.66.66              1             0 100 ?

Rack1R4#debug ip rip
RIP protocol debugging is on
RIP: sending v2 update to 224.0.0.9 via FastEthernet0/1 (204.12.1.4)
RIP: build update entries
        54.1.1.0/24 via 0.0.0.0, metric 1, tag 0
        212.18.0.0/24 via 0.0.0.0, metric 11, tag 0
        212.18.1.0/24 via 0.0.0.0, metric 2, tag 0
        212.18.2.0/24 via 0.0.0.0, metric 11, tag 0
        212.18.3.0/24 via 0.0.0.0, metric 2, tag 0
```

Notice that in both cases metrics were carried as BGP MEDs and preserved across the VPN connection.

Task 4.3

```
R4:
router bgp 400
 address-family ipv4 vrf VPN_A
  neighbor 204.12.1.254 remote-as 54
!
! Needed to override the last AS with local AS
!
  neighbor 204.12.1.254 as-override
!
! No-prepend is needed to make R6 learn prefixes from AS 54
!
  neighbor 204.12.1.254 local-as 100 no-prepend

R6:
router bgp 100
 address-family ipv4 vrf VPN_A
  neighbor 54.1.1.254 remote-as 54
  neighbor 54.1.1.254 as-override
```

Task 4.3 Verification

Check the BGP prefixes learned by the backbone routers. Notice that in the real exam you may have not the ability to access the backbone routers.

```
BB1>show ip bgp regexp 100
BGP table version is 411, local router ID is 212.18.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 30.0.0.0/16      54.1.1.6                               0 100 400 ?
*> 30.1.0.0/16      54.1.1.6                               0 100 400 ?
*> 30.2.0.0/16      54.1.1.6                               0 100 400 ?
*> 30.3.0.0/16      54.1.1.6                               0 100 400 ?
*> 31.0.0.0/16      54.1.1.6                               0 100 400 ?
*> 31.1.0.0/16      54.1.1.6                               0 100 400 ?
*> 31.2.0.0/16      54.1.1.6                               0 100 400 ?
*> 31.3.0.0/16      54.1.1.6                               0 100 400 ?
r> 54.1.1.0/24      54.1.1.6                 0             0 100 ?
*> 204.12.1.0       54.1.1.6                               0 100 400 ?
* i                 172.16.4.3               0    100      0 100 400 ?
r> 212.18.0.0       54.1.1.6                10             0 100 ?
r> 212.18.1.0       54.1.1.6                 1             0 100 ?
r> 212.18.2.0       54.1.1.6                10             0 100 ?
r> 212.18.3.0       54.1.1.6                 1             0 100 ?

BB3>show ip bgp regexp 100
```

```
BGP table version is 361, local router ID is 31.3.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
r>i30.0.0.0/16      172.16.4.1               0    100      0 100 400 ?
r                   204.12.1.4               1             0 100 400 ?
r>i30.1.0.0/16      172.16.4.1               0    100      0 100 400 ?
r                   204.12.1.4               1             0 100 400 ?
r>i30.2.0.0/16      172.16.4.1               0    100      0 100 400 ?
r                   204.12.1.4               1             0 100 400 ?
r>i30.3.0.0/16      172.16.4.1               0    100      0 100 400 ?
r                   204.12.1.4               1             0 100 400 ?
r>i31.0.0.0/16      172.16.4.1               0    100      0 100 400 ?
r                   204.12.1.4               1             0 100 400 ?
r>i31.1.0.0/16      172.16.4.1               0    100      0 100 400 ?
r                   204.12.1.4               1             0 100 400 ?
r>i31.2.0.0/16      172.16.4.1               0    100      0 100 400 ?
r                   204.12.1.4               1             0 100 400 ?
r>i31.3.0.0/16      172.16.4.1               0    100      0 100 400 ?
r                   204.12.1.4               1             0 100 400 ?
r>i54.1.1.0/24      172.16.4.1               0    100      0 100 ?
   Network          Next Hop            Metric LocPrf Weight Path
r                   204.12.1.4                             0 100 400
100 ?
r i204.12.1.0       172.16.4.1               0    100      0 100 400 ?
r>                  204.12.1.4               0             0 100 400 ?
r>i212.18.0.0       172.16.4.1              10    100      0 100 ?
r                   204.12.1.4                             0 100 400
100 ?
r>i212.18.1.0       172.16.4.1               1    100      0 100 ?
r                   204.12.1.4                             0 100 400
100 ?
r>i212.18.2.0       172.16.4.1              10    100      0 100 ?
r                   204.12.1.4                             0 100 400
100 ?
r>i212.18.3.0       172.16.4.1               1    100      0 100 ?
r                   204.12.1.4                             0 100 400
100 ?
```

Task 5.2

R1, R2, R3, R4, R5, SW1, and SW2:
```
ip pim autorp listener
```

R2, R5, and SW2:
```
interface Loopback0
 ip pim sparse-mode
```

R2:
```
ip pim send-rp-announce Loopback0 scope 16 group-list 50
!
access-list 50 permit 225.0.0.0 0.255.255.255
```

R5:
```
ip pim send-rp-announce Loopback0 scope 16 group-list 50
!
access-list 50 permit 239.0.0.0 0.255.255.255
```

SW2:
```
ip pim send-rp-discovery Loopback0 scope 16
```

Task 5.2 Breakdown

Since PIM sparse mode was required in task 6.1 and this section asks for Auto-RP, the **`ip pim autorp listener`** command will need to be used on all multicast devices to enable the 224.0.1.39 and 224.0.1.40 groups to be distributed in dense mode. For more information concerning the **`ip pim autorp listener`** command, refer to lab 3, section 6.1 breakdown.

Task 5.2 Verification

Verify the group to RP mappings on each multicast enabled router. For example on SW2:

```
Rack1SW2#show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP-mapping agent (Loopback0)

Group(s) 225.0.0.0/8
  RP 150.1.2.2 (?), v2v1
    Info source: 150.1.2.2 (?), elected via Auto-RP
         Uptime: 00:00:58, expires: 00:01:57
Group(s) 239.0.0.0/8
  RP 150.1.5.5 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:00:11, expires: 00:02:49
```

Task 5.3

R2:
```
interface Serial0/0
 ip pim nbma-mode
!
interface Tunnel0
 ip pim sparse-mode
```

R3:
```
interface FastEthernet0/0
 ip igmp join-group 225.25.25.25
!
interface FastEthernet0/1
 ip igmp join-group 239.39.39.39
```

R5:
```
interface Tunnel0
 ip pim sparse-mode
```

Task 5.3 Breakdown

The problem encountered with this task is that R2 will not forward multicast packets received on an interface S0/0 back out interface S0/0. This will cause the multicast pings from R1 to not reach R3. To overcome this issue, PIM NBMA mode can be configured.

Before the NBMA mode is configured:

```
Rack1R1#ping 225.25.25.25

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 225.25.25.25, timeout is 2 seconds:
.
Rack1R1#ping 239.39.39.39

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 239.39.39.39, timeout is 2 seconds:
.
Rack1R1#
```

After the NBMA mode is configured:

```
Rack1R1#ping 225.25.25.25

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 225.25.25.25, timeout is 2 seconds:

Reply to request 0 from 141.1.13.3, 36 ms

Rack1R1#ping 239.39.39.39

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 239.39.39.39, timeout is 2 seconds:

Reply to request 0 from 141.1.13.3, 36 ms
Rack1R1#
```

Copyright © 2009 Internetwork Expert www.INE.com

Task 5.3 Verification

Simulate multicast packets from R1 to group 225.25.25.25:

```
Rack1R1#ping
Protocol [ip]:
Target IP address: 225.25.25.25
Repeat count [1]: 100
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Interface [All]: Serial0/0.1
Time to live [255]:
Source address: 192.10.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 225.25.25.25, timeout is 2 seconds:
Packet sent with a source address of 192.10.1.1

Reply to request 0 from 141.1.123.3, 92 ms
Reply to request 1 from 141.1.123.3, 92 ms
Reply to request 2 from 141.1.123.3, 92 ms
```

Verify the multicast routing table on R2:

```
Rack1R2#show ip mroute
IP Multicast Routing Table
<snip>

(*, 225.25.25.25), 00:10:49/stopped, RP 150.1.2.2, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, 141.1.123.3, Forward/Sparse, 00:10:49/00:02:35

<output omitted>

(192.10.1.1, 225.25.25.25), 00:01:43/00:03:22, flags: T
  Incoming interface: Serial0/0, RPF nbr 141.1.123.1
  Outgoing interface list:
    Serial0/0, 141.1.123.3, Forward/Sparse, 00:01:43/00:02:46
```

Finally, simulate multicast packets from R4 to group 239.39.39.39:

```
Rack1R4#ping
Protocol [ip]:
Target IP address: 239.39.39.39
Repeat count [1]: 100
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Interface [All]: FastEthernet0/0
Time to live [255]:
```

```
Source address: 204.12.1.4
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.39.39.39, timeout is 2 seconds:
Packet sent with a source address of 204.12.1.4

Reply to request 0 from 141.1.123.3, 40 ms
Reply to request 1 from 141.1.123.3, 36 ms
Reply to request 2 from 141.1.123.3, 32 ms
```

Verify the multicast table on R2 again (note Tunnel0 is used as RPF interface):

```
Rack1R2#show ip mroute 239.39.39.39
<snip>

(*, 239.39.39.39), 01:52:21/stopped, RP 150.1.5.5, flags: S
  Incoming interface: Tunnel0, RPF nbr 141.1.25.5
  Outgoing interface list:
    Serial0/0, 141.1.123.3, Forward/Sparse, 01:52:21/00:03:00

(204.12.1.4, 239.39.39.39), 00:00:06/00:03:23, flags: T
  Incoming interface: Tunnel0, RPF nbr 141.1.25.5
  Outgoing interface list:
    Serial0/0, 141.1.123.3, Forward/Sparse, 00:00:06/00:03:23
```

Task 5.4

```
SW1:
interface FastEthernet0/3
 ip multicast rate-limit out 1000
```

Task 5.4 Breakdown

Although standard QoS methods could be used to meet the requirements for this section, the **`ip multicast rate-limit`** command is the simplest solution. Other options for this command can be configured to limit the amount of multicast traffic inbound, from a specific source address, and/or multicast traffic sent to a specific multicast group. If a value of 0 is used, all multicast traffic will be dropped.

Task 6.1

```
R6:
!
! Hosts opened for Telnet Access
!
ip access-list extended ACL_OUTSIDE_TO_INSIDE_TELNET
 permit tcp any 141.1.7.0 0.0.0.255
 permit tcp any 141.1.77.0 0.0.0.255

!
! Matching telnet protocol in addition to the ACL
!
class-map type inspect CMAP_OUTSIDE_TO_INSIDE_TELNET
 match access-group name ACL_OUTSIDE_TO_INSIDE_TELNET
 match protocol telnet

!
! The host opened for HTTP/HTTPs access
!
ip access-list extended ACL_OUTSIDE_TO_INSIDE_HTTP
 permit tcp any host 141.1.88.100

!
! Combine two protocol in “OR” pair
!
class-map type inspect match-any CMAP_HTTP_HTTPS
 match protocol http
 match protocol https

!
! Class map that matches HTTP/HTTPs traffic to the server
!
class-map type inspect CMAP_OUTSIDE_TO_INSIDE_HTTP
 match access-group name ACL_OUTSIDE_TO_INSIDE_HTTP
 match class-map CMAP_HTTP_HTTPS

!
! Other protocols allowed across the firewall (DNS/ICMP)
!
class-map type inspect match-any CMAP_OTHER_PROTOCOLS
 match protocol dns
 match protocol icmp

policy-map type inspect PMAP_OUTSIDE_TO_INSIDE
 class CMAP_OUTSIDE_TO_INSIDE_TELNET
  inspect
 class CMAP_OUTSIDE_TO_INSIDE_HTTP
  inspect
!
! Rate-Limit DNS and ICMP
!
 class CMAP_OTHER_PROTOCOLS
  inspect
  police rate 128000 burst 8000
!
class-map type inspect match-any CMAP_INSIDE_TO_OUTSIDE
```

```
 match protocol udp
 match protocol tcp
 match protocol icmp
!
policy-map type inspect PMAP_INSIDE_TO_OUTSIDE
 class CMAP_INSIDE_TO_OUTSIDE
  inspect
!
zone security INSIDE
zone security OUTSIDE
!
zone-pair security ZP_INSIDE_TO_OUTSIDE source INSIDE destination
OUTSIDE
service-policy type inspect PMAP_INSIDE_TO_OUTSIDE
!
zone-pair security ZP_OUTSIDE_TO_INSIDE source OUTSIDE destination
INSIDE
service-policy type inspect PMAP_OUTSIDE_TO_INSIDE

!
! Assign zones to the interfaces
!
interface FastEthernet 0/1
 zone-member security OUTSIDE
!
interface FastEthernet 0/0
 zone-member security INSIDE
```

Task 6.1 Verification

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

policy exists on zp ZP_INSIDE_TO_OUTSIDE
 Zone-pair: ZP_INSIDE_TO_OUTSIDE

  Service-policy inspect : PMAP_INSIDE_TO_OUTSIDE

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

   Inspect
        Session creations since subsystem startup or last reset 0
        Current session counts (estab/half-open/terminating) [0:0:0]
        Maxever session counts (estab/half-open/terminating) [0:0:0]
        Last session created never
        Last statistic reset never
        Last session creation rate 0
        Maxever session creation rate 0
        Last half-open session total 0

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

policy exists on zp ZP_OUTSIDE_TO_INSIDE
 Zone-pair: ZP_OUTSIDE_TO_INSIDE

  Service-policy inspect : PMAP_OUTSIDE_TO_INSIDE

    Class-map: CMAP_OUTSIDE_TO_INSIDE_TELNET (match-all)
      Match: access-group name ACL_OUTSIDE_TO_INSIDE_TELNET
      Match: protocol telnet

   Inspect
        Session creations since subsystem startup or last reset 0
        Current session counts (estab/half-open/terminating) [0:0:0]
        Maxever session counts (estab/half-open/terminating) [0:0:0]
        Last session created never
        Last statistic reset never
        Last session creation rate 0
        Maxever session creation rate 0
        Last half-open session total 0

    Class-map: CMAP_OUTSIDE_TO_INSIDE_HTTP (match-all)
      Match: access-group name ACL_OUTSIDE_TO_INSIDE_HTTP
      Match: class-map match-any CMAP_HTTP_HTTPS
        Match: protocol http
          0 packets, 0 bytes
          30 second rate 0 bps
        Match: protocol https
          0 packets, 0 bytes
          30 second rate 0 bps

   Inspect
        Session creations since subsystem startup or last reset 0
        Current session counts (estab/half-open/terminating) [0:0:0]
        Maxever session counts (estab/half-open/terminating) [0:0:0]
        Last session created never
        Last statistic reset never
        Last session creation rate 0
        Maxever session creation rate 0
        Last half-open session total 0

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

   Inspect
        Session creations since subsystem startup or last reset 0
        Current session counts (estab/half-open/terminating) [0:0:0]
        Maxever session counts (estab/half-open/terminating) [0:0:0]
        Last session created never
        Last statistic reset never
        Last session creation rate 0
        Maxever session creation rate 0
        Last half-open session total 0
       Police
        rate 128000 bps,8000 limit
        conformed 0 packets, 0 bytes; actions: transmit
        exceeded 0 packets, 0 bytes; actions: drop
        conformed 0 bps, exceed 0 bps

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

Task 6.2

R4:

```
ip cef
!
interface FastEthernet0/1
 ip verify unicast reverse-path
```

☞ **Legacy command**

OR

```
ip cef
!
interface FastEthernet0/1
 ip verify unicast source reachable-via any
```

Task 6.2 Breakdown

Unicast reverse path forwarding was developed to help solve issues concerning DoS attacks. With DoS attacks, the source IP address of the packet is usually spoofed. Basically, unicast reverse path forwarding checks the source IP address of all packets received inbound on an interface to see if that particular interface is the interface the router would use to route a packet to that source IP address. Theoretically, this is a good idea and in some situations can be helpful, but in a real network this can cause problems with asymmetrical routing. Let's look at an example of strict mode:

A packet with a source IP address of 141.X.37.100 is received inbound on R4 interface Fa0/1. With unicast RPF enabled, R4 would assume the source IP address must be spoofed as R4 will normally route via R5 to reach that particular subnet. This will cause R4 to drop this packet. Although this might be the action we want, what if R3 is routing through BB1 & BB3 (AS 54) to reach the networks behind R4, while R4 is routing through R5 & R2 to reach R3. In this situation, unicast RPF would cause packets to be incorrectly dropped.

Another example would be for a remote site with one connection to the rest of the network. Normally, to prevent spoofed IP addresses from being sent by the remote site, an access-list is created to only permit packets with the remote site's network as the source IP address. This access-list is applied inbound on the main site's router. Whenever a new network is added to the remote site, an additional route is added in the hub router and the access-list will need to be updated. For an enterprise network that does not make a lot of changes, this might not mean a lot of additional work, but for a large ISP this could possibly mean a lot of additional work. A simpler solution would be to use unicast RPF as the access-list is not needed.

For reasons we can see from the examples, unicast RPF is normally implemented on the edge of the network where symmetrical routing is more likely to occur. If unicast RPF is used in the core of a network, ensure that asymmetric routing does not occur.

In addition to strict mode, there is also the possibility of loose mode. In strict mode, the route to the source address for the received packet must be via the same interface. For loose mode, there just needs to be a matching route in the routing table, and it can be via any interface.

The 28.119 networks learned via BGP are learned from BB1, but the networks are originated on BB3. A ping to these networks will pass out via BB1, and will return to your topology via R4, since BB3 is learning routes via R4. Because of the asymmetric routing, strict mode would cause these pings to fail, whereas loose mode will allow the return traffic to enter R4's interface.

Note

Unicast RPF requires CEF to be enabled globally.

Task 6.2 Verification

Verify that uRPF is configured on the interface:

```
Rack1R4#show ip interface Fa0/1 | include verif
  IP verify source reachable-via ANY
  0 verification drops
  0 suppressed verification drops
```

Task 6.3

R6:

```
ip access-list extended ELIGIBLE_TRAFFIC
 permit   tcp any any eq bgp
 permit   tcp any eq bgp any
 permit   tcp any any eq telnet
 permit   tcp any any eq 22

!
class-map match-all ELIGIBLE_TRAFFIC
 match access-group name ELIGIBLE_TRAFFIC
!
policy-map CONTROL_PLAN_POLICING
 class ELIGIBLE_TRAFFIC
 class class-default
  police rate 1000 pps
!
control-plane
 service-policy input CONTROL_PLAN_POLICING
```

Task 6.3 Verification

```
Rack1R6#show policy-map control-plane input
Control Plane

  Service-policy input: CONTROL_PLAN_POLICING

    Class-map: ELIGIBLE_TRAFFIC (match-all)
      3 packets, 574 bytes
      5 minute offered rate 0 bps
      Match: access-group name ELIGIBLE_TRAFFIC

    Class-map: class-default (match-any)
      22 packets, 1397 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      police:
          rate 1000 pps, burst 244 packets
        conformed 22 packets; actions:

          transmit
        exceeded 0 packets; actions:
          drop
        conformed 1 pps, exceed 0 pps
```

Task 7.1

```
R3 and R6:
snmp-server community CISCO_RO RO 2
snmp-server community CISCO_RW RW 2
snmp-server enable traps hsrp
snmp-server host 141.1.7.100 CISCO tty
snmp-server host 141.1.77.100 version 2c CISCO hsrp
!
access-list 2 permit 141.1.77.100
access-list 2 permit 141.1.7.100
```

Task 7.1 Breakdown

This configuration is similar to SNMP configurations from earlier labs with the exception of the version option. The default SNMP version is version 1. Also, note that SNMP traps relating to HSRP will only be sent to host 141.1.77.100.

Task 7.1 Verification

Verify the ACL and basic SNMP configuration:

```
Rack1R3#show access-lists 2
Standard IP access list 2
    10 permit 141.1.77.100
    20 permit 141.1.7.100

Rack1R3#show snmp
Chassis: 10578766
<output omitted>
SNMP logging: enabled
    Logging to 141.1.7.100.162, 0/10, 0 sent, 0 dropped.
    Logging to 141.1.77.100.162, 0/10, 0 sent, 0 dropped.
```

And finally verify that the configuration commands were accepted:

```
Rack1R3#show running-config | include snmp
snmp-server community CISCO_RO RO 2
snmp-server community CISCO_RW RW 2
snmp-server enable traps hsrp
snmp-server host 141.1.7.100 CISCO  tty
snmp-server host 141.1.77.100 version 2c CISCO  hsrp
```

Task 7.2

```
R2:
username NOC password 0 CISCO
username NOC autocommand menu NOC
!
menu NOC title # Menu for NOC users #
menu NOC prompt # Choose your selection: #
menu NOC text 1. Ping R5
menu NOC command 1. ping 150.1.5.5
menu NOC options 1. pause
menu NOC text 2. Ping R6
menu NOC command 2. ping 150.1.6.6
menu NOC options 2. pause
menu NOC text 3. Traceroute to R5
menu NOC command 3. trace 150.1.5.5
menu NOC options 3. pause
menu NOC text 4. Traceroute to R6
menu NOC command 4. trace 150.1.6.6
menu NOC options 4. pause
menu NOC text 5. Exit
menu NOC command 5. exit
menu NOC clear-screen
!
line vty 0 4
 login local
```

Task 7.2 Breakdown

This is a standard menu configuration. Ensure that **`login local`** is configured under the VTY lines to prompt for the username along with the password. Depending on the IOS version and hardware platform, there are usually more that 5 VTY lines. You may want to ask if something should be applied to ALL VTY lines. Here, we are just applying to the first five. Also, the autocommand is needed to activate the menu once the NOC user logs in. Technically, the autocommand could have been configured under the VTY line itself to meet the requirements of this section. However, binding the autocommand to the username allows more flexibility as to which users must use the menu.

☠ Pitfall

When using menus, ensure that the user is given an option to exit the menu.

Task 7.2 Verification

Telnet to R2 to verify the menu:

```
Rack1R3#telnet 150.1.2.2
Trying 150.1.2.2 ... Open

User Access Verification

Username: NOC
Password:
Menu for NOC users

    1.          Ping R5

    2.          Ping R6

    3.          Traceroute to R5

    4.          Traceroute to R6

    5.          Exit
```

Task 7.3

```
R2:
ip alias 141.1.0.22 23
```

Task 7.3 Breakdown

The **ip alias** command will allow the router to answer for the aliased IP address. Although this command is not commonly used, it could be useful in a situation where users need to access a particular async line by telneting to the DNS name along with the standard TCP port of 23.

```
Rack1AS#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1AS(config)#ip alias 172.16.2.122 2008
Rack1AS(config)#ip host AS-Line8 172.16.2.122
Rack1AS(config)#^Z
Rack1AS#telnet AS-Line8
Trying AS-Line8 (172.16.2.122)... Open

Rack1SW2#

Rack1AS#show sessions
Conn Host                Address             Byte  Idle Conn Name
*  8 AS-Line8            172.16.2.122           0     0 AS-Line8

Rack1AS#
```

Task 7.3 Verification

Verify that the alias is configured:

```
Rack1R2#show ip aliases
Address Type             IP Address      Port
Interface                141.1.0.2
Interface                141.1.25.2
Interface                150.1.2.2
Alias                    141.1.0.22     23
Interface                141.1.123.2
```

Verify the ARP entry for the aliased IP:

```
Rack1R2#show ip arp 141.1.0.2
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  141.1.0.2               -   0004.27b5.2fa0  ARPA
FastEthernet0/0
```

Finally telnet from R5 to 141.1.0.2:

```
Rack1R5#telnet 141.1.0.2
Trying 141.1.0.2 ... Open


User Access Verification

Username: NOC
Password:
Menu for NOC users
<output omitted>
```

Task 7.4

R3:
```
interface FastEthernet0/1
 standby 1 ip 141.1.36.1
 standby 1 priority 150
 standby 1 preempt
 standby 2 ip 141.1.36.2
```

R6:
```
interface FastEthernet0/0
 standby 1 ip 141.1.36.1
 standby 2 ip 141.1.36.2
 standby 2 priority 150
 standby 2 preempt
```

Task 7.4 Breakdown

This section will require HSRP with R3 active for one IP address (141.1.36.1) and R6 active for another IP address (141.1.36.2). Therefore, the appropriate DHCP server for this segment can assign one address as the default gateway to a certain pool of clients, while assigning the other address as the default gateway for another pool of clients. Note that these addresses have been arbitrarily chosen. If R3 becomes unavailable, R6 will take over for 141.1.36.1 and vice versa. The HSRP priority is set above the default of 100 along with the preempt option, so that R3 and R6 can take back over for their HSRP group after becoming available again.

Note

Without the HSRP preempt option, a router will not take over for an HSRP group even if its own priority is higher than the current active router.

Task 7.4 Verification

Verify the HSRP configuration:

```
Rack1R6#show standby
FastEthernet0/0 - Group 1
  State is Standby
    1 state change, last state change 00:00:13
  Virtual IP address is 141.1.36.1
  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 1.368 secs
  Preemption disabled
  Active router is 141.1.36.3, priority 150 (expires in 9.108 sec)
  Standby router is local
  Priority 100 (default 100)
  IP redundancy name is "hsrp-Fa0/0-1" (default)
FastEthernet0/0 - Group 2
  State is Active
    1 state change, last state change 00:00:30
  Virtual IP address is 141.1.36.2
  Active virtual MAC address is 0000.0c07.ac02
    Local virtual MAC address is 0000.0c07.ac02 (v1 default)
  Hello time 3 sec, hold time 10 sec
    Next hello sent in 2.112 secs
  Preemption enabled
  Active router is local
  Standby router is 141.1.36.3, priority 100 (expires in 7.112 sec)
  Priority 150 (configured 150)
  IP redundancy name is "hsrp-Fa0/0-2" (default)
```

Task 7.5

```
R3:
ip host R4 150.1.4.4
!
busy-message R4 “Connection Unsuccessful”
```

Task 7.5 Breakdown

The “host failed” message (aka busy-message) is used to display a custom message with a telnet connection fails to a particular host. The key to the **`busy-message`** is that the **`busy-message`** is configured on the host attempting to make the connection, and not the host that actually refused the connection. To have the router that refused the connection display a “line in use” message, use the **`refuse-message`** under the remote router’s VTY line.

The **`busy-message`** only takes a hostname and will not take an IP address. This means the **`ip host`** command will need to be configured in conjunction with the **`busy-message`** command assuming a DNS server has not been defined.

Task 7.5 Verification

Shutdown Serial 1/0 on R3, and try connecting to R4 by name:

```
Rack1R3(config)#interface s1/0
Rack1R3(config-if)#shutdown

Rack1R3#R4
Translating "R4"
 Connection Unsuccessful
```

Task 8.1

R1:
```
interface FastEthernet0/0
 random-detect
 random-detect precedence 0 15 40 10
```

Task 8.1 Breakdown

The section will require Weighted Random Early Detection (WRED) to be configured. Random early detection is designed to avoid tail drop by randomly dropping packets prior to the output queue of an interface actually becoming full. When the output queue becomes full, all packets that attempt to enter the tail (end) of the queue will have to be dropped, hence the term tail drop. If these dropped packets were all TCP packets, then theoretically all TCP sessions that had dropped packets will back off and perform TCP slow start. With TCP slow start, a TCP session will start with a small TCP window size and gradually increase the window size until congestion or delay is experienced. If all TCP sessions back off and then start up again in synchronization with each other, the same situation will occur once the interface's queue to become full again. This results in periods of high congestion followed by periods of low utilization. This type of behavior is called global synchronization, and is what random early detection is designed to prevent.

Random detection is enabled by issuing the interface level command **`random-detect`**. Once enabled, random detect applies various low and high threshold values based on the IP precedence value of the packet trying to exit the interface. To change these threshold values, use the interface level command **`random-detect precedence [precedence] [minimum threshold] [maximum threshold] [mark probability demonitator].`**

Note

Weighted RED is the process of using different queues and thresholds for packets with different IP precedence values.

The *precedence* option references the IP precedence value for which the thresholds will be modified. The *minimum threshold* is the threshold value at which RED will become active. The *maximum threshold* is the threshold value at which all packets above this threshold will be dropped. The *mark probability denominator* is used to determine how aggressively packets will be dropped. The lower the number, the more aggressively packets will be dropped. When the number of packets in the queue matches the maximum threshold, 1/*(mark probability denominator)* will be dropped.

If the *mark probability denominator* is 10 as in this section, when the average queue size equals 40 IP precedence 0 packets, 1 of every 10 packets will be dropped. With an average queue size between 15 and 39 packets, less than 1 of every 10 packets will be dropped. The closer the average queue size gets to 40, the closer the number of dropped packets will equal 1 of every 10. Once the average queue exceeds 40, all packets above 40 will be dropped.

As seen from the above output, the default values of WRED dictate that the higher the IP Precedence value the higher the minimum threshold. This means that a larger volume of packets with a high precedence value must be stuck in the output queue before they start to get dropped. Assuming a relatively equal distribution of packet precedence, these values imply that packets with a higher precedence value are less likely to get dropped than packets with a lower precedence value.

Task 8.1 Verification

```
Rack1R1#show queueing interface FastEthernet0/0
Interface FastEthernet0/0 queueing strategy: random early detection
(WRED)
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0

  class          Random drop      Tail drop    Minimum Maximum  Mark
                  pkts/bytes       pkts/bytes  thresh  thresh   prob
      0           0/0              0/0            15      40    1/10
      1           0/0              0/0            22      40    1/10
      2           0/0              0/0            24      40    1/10
      3           0/0              0/0            26      40    1/10
      4           0/0              0/0            28      40    1/10
      5           0/0              0/0            31      40    1/10
      6           0/0              0/0            33      40    1/10
      7           0/0              0/0            35      40    1/10
   rsvp           0/0              0/0            37      40    1/10
```

Task 8.2

R5:
```
ip cef
!
class-map match-all SMTP
 match protocol smtp
!
policy-map QoS
 class SMTP
  bandwidth 1500
!
interface FastEthernet0/1
 service-policy output QoS
```

Task 8.2 Breakdown

This is a standard MQC configuration, and dictates that SMTP traffic will be guaranteed 1500Kbps of the output queue of the FastEthernet0/1 interface in the case of congestion. SMTP may use more than 1500Kbps, however only 1500Kbps is guaranteed in the case of congestion.

Note

When matching a protocol in a class-map, CEF should be enabled globally. CEF is on by default in newer IOS versions, but it is possible that it was disabled in a pre-loaded initial configuration file. Make sure to verify that CEF is enabled.

Task 8.2 Verification

Verify that the policy-map is attached and the class-map is configured:

```
Rack1R5#show policy-map interface Fa0/1
 FastEthernet0/1

  Service-policy output: QoS

    Class-map: SMTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol smtp
      Queueing
        Output Queue: Conversation 265
        Bandwidth 1500 (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)
      23 packets, 2310 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Task 8.3

```
R5:
class-map match-all ABOVE_1250_BYTES
 match packet length min 1251
!
policy-map QoS
 class ABOVE_1250_BYTES
  police cir 2500000
```

Task 8.3 Breakdown

Policing packets based on the packet’s size can be useful in situations where framentation is not possible (i.e. an HDLC link). Large packets can cause excessive delay on slow speed links due to serization delay. In this section, packets larger than 1250 bytes will be limited to 2.5 Mbps.

☠ Pitfall

The IOS is not consistent in how QoS parameters are entered. Some commands take values in as bits, some in as bytes, and some in as kilobytes. When in the CCIE lab, use the ‘?’ along with the command to view how the parameter is accepted by the IOS for that particular command. Do not expect the values given in the lab itself to be the same as what will need to be entered in the configuration. Entering a value in as bytes when the command takes bits will cause you to lose points for that section.

Task 8.3 Verification

Verify that the policy-map is configured and applied:

```
Rack1R5#show policy-map interface Fa0/1
 FastEthernet0/1

  Service-policy output: QoS

  <output omitted>

    Class-map: ABOVE_1250_BYTES (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: packet length min 1251
      police:
          cir 2500000 bps, bc 78125 bytes
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          drop
        conformed 0 bps, exceed 0 bps

    Class-map: class-default (match-any)
      220 packets, 21377 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Simulate a packet stream from R4 and verify the statistics:

```
Rack1R4#ping 141.1.0.8 size 1250 repeat 10000 timeout 1

Type escape sequence to abort.
Sending 10000, 1250-byte ICMP Echos to 141.1.0.8, timeout is 1 seconds:
!!!!!!!!!!!!!

Rack1R5#show policy-map interface Fa0/1
 FastEthernet0/1

  Service-policy output: QoS

<output omitted>

    Class-map: ABOVE_1250_BYTES (match-all)
      27 packets, 34128 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: packet length min 1251
      police:
          cir 2500000 bps, bc 78125 bytes
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          drop
        conformed 0 bps, exceed 0 bps
<output omitted>
```

```
Rack1R4#ping 141.1.0.8 size 1251 repeat 10000 timeout 1

Type escape sequence to abort.
Sending 10000, 1251-byte ICMP Echos to 141.1.0.8, timeout is 1 seconds:
!!!!!!!!!!!!!

Rack1R5#show policy-map interface Fa0/1
 FastEthernet0/1

  Service-policy output: QoS

<output omitted>

    Class-map: ABOVE_1250_BYTES (match-all)
      27 packets, 34128 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: packet length min 1250
      police:
          cir 2500000 bps, bc 78125 bytes
        conformed 27 packets, 34128 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          drop
        conformed 0 bps, exceed 0 bps
<output omitted>
```

Task 8.4

```
R4 and R5:
interface Serial0/1/0
 compress predictor
```

Task 8.4 Breakdown

There are two common types of compression used with PPP, stacker and predictor. Stacker is more CPU intensive but more forgiving on memory utilization. Predictor is less CPU intensive but utilizes more memory. The key to determining which to use here is based on the word “guessing” that is used in the task. Predictor will try to predict the next sequence of characters in the data stream. Although “predicting” is not the same as “guessing”, the use of the word leads us to interrupt the task to want predictor to be used over stac.

Task 8.4 Verification

```
Rack1R4#show compress
 Serial0/1/0
         Software compression enabled
         uncompressed bytes xmt/rcv 327/332
         compressed bytes   xmt/rcv 0/0
         Compressed bytes sent:         0 bytes   0 Kbits/sec
         Compressed bytes recv:         0 bytes   0 Kbits/sec
         1  min avg ratio xmt/rcv 0.709/0.922
         5  min avg ratio xmt/rcv 0.709/0.922
         10 min avg ratio xmt/rcv 0.709/0.922
         no bufs xmt 0 no bufs rcv 0
         resyncs 0

Rack1R4#ping 141.1.45.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.45.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/16/16 ms

Rack1R4#show compress
 Serial0/1/0
         Software compression enabled
         uncompressed bytes xmt/rcv 837/842
         compressed bytes   xmt/rcv 170/168
         Compressed bytes sent:  170 bytes   0 Kbits/sec  ratio: 4.923
         Compressed bytes recv:  168 bytes   0 Kbits/sec  ratio: 5.011
         1  min avg ratio xmt/rcv 1.190/1.482
         5  min avg ratio xmt/rcv 1.190/1.482
         10 min avg ratio xmt/rcv 1.190/1.482
         no bufs xmt 0 no bufs rcv 0
         resyncs 0
```