## Copyright Information

## *Disclaimer*

# Table of Contents

# IP Routing

**Note**

Load the *Full Layer 2* initial configurations prior to starting.

## 3.1 Routing to Multipoint Broadcast Interfaces

- Configure R1 with a static route for R4’s Loopback0 network pointing to the IP address of R4.
- Configure R1 with a static route for R6’s Loopback0 network pointing to R1’s Ethernet interface connecting to VLAN 146.
- Ensure that R1 can ping the Loopback0 interfaces of R4 and R6.
- Verify that R6 is running Proxy ARP by checking the ARP table of R1.
- Disable Proxy ARP on R6’s connections to VLAN 146.
- Modify R1’s ARP table so that it still has IP reachability to the Loopback0 interface of R6.

## 3.2 Routing to NBMA Interfaces

- Configure R1 and R2 with default routes pointing out their point-to-point Frame Relay interfaces connecting to R5.
- Configure R5 with a static route for R1’s Loopback0 network pointing to the IP address of R1.
- Configure R5 with a static route for R2’s Loopback0 network pointing out its main Frame Relay interface connecting to R2.
- Ensure that R1, R2, and R5 can all ping each other’s Loopback0 interfaces.

## 3.3 Longest Match Routing

- Configure R4 and R5 with routes to each other's Loopback0 interfaces via the point-to-point Serial interface between them.
- Configure R4 and R5 to route the rest of the 150.X.0.0/16 network via the Frame Relay connection between them.
- Ensure that traffic between R4 and R5's Loopback0 networks transits the point-to-point link between them, but is rerouted out the Frame Relay connection if the point-to-point link is down.

## 3.4 Floating Static Routes

- Remove the previous static routes on R4 and R5.
- Configure R4 and R5 with identical static routes for each others' Loopback0 networks out both their Frame Relay connections and the point-to-point connection between them.
- Using administrative distance ensure that traffic between R4 and R5's Loopback0 networks transits the point-to-point link between them, but is rerouted out the Frame Relay connection if the point-to-point link is down.

## 3.5 Backup Interface

- Modify the administrative distance of R4's static routes to R5's Loopback0 interface to be identical out the Frame Relay and point-to-point interfaces.
- Configure the backup interface feature on R4 to route traffic for R5's Loopback0 out the Frame Relay network unless the PVC to R5 is down.
- If the PVC to R5 is down this traffic should be rerouted out the point-to-point link.
- Ensure the backup link is activate 3 seconds after the main link fails, and deactivated once the main link is active for 60 seconds.

## 3.6 Reliable Static Routing with Enhanced Object Tracking

- Remove all previous routing configurations.
- Configure static routes on R5 so that it has reachability to the Loopback0 networks of R1 and R4 via the Frame Relay network.
- Configure R1 with a static route to R4's Loopback0 network via the Frame Relay connection to R5.
- Configure R4 with a static route to R1's Loopback0 network via the Ethernet connection to VLAN 146.
- Configure a second static route on R4 for R1's Loopback0 network via the Frame Relay connection to R5 with an administrative distance of 2.

- Configure an IP SLA instance on R4 to ping R1's Ethernet connection to VLAN 146 every 5 seconds.
- Create an enhanced object to track this SLA instance.
- Tie this enhanced object to the static route R4 has for R1's Loopback0 out the connection to VLAN 146.
- Ensure that R4 maintains IP reachability to R1's Loopback0 in the case that R1's Ethernet interface is down.

## 3.7 Policy Routing

- Remove all previous routing configuration.
- Configure default routes on R4 and R6 pointing to the VLAN 146 connection of R1.
- Configure a default route on R3 pointing to the point-to-point link connecting to R1.
- Configure a default route on R5 pointing to R1 out the Frame Relay network.
- Configure a static route on R3 for R5's Loopback0 network and a static route on R5 for R3's Loopback0 network out the Frame Relay network.
- Create two extended access-lists on R1, one named FROM_R4 and the other named FROM_R6.
- Access-list FROM_R4 should match all IP packets coming from R4, while access-list FROM_R6 should match all IP packets coming from R6.
- Configure policy-routing on R1 so that traffic from R4 is routed out the serial link between R1 and R3, and traffic from R6 is routed out the Frame Relay link to R5.
- Use traceroute on R4 and R6 for R3 and R5's Loopback0 networks to verify that this configuration is functional.

## 3.8 Reliable Policy Routing

- Remove the previous static and policy routing configurations.
- Remove the point-to-point Frame Relay subinterface on R1 and replace this with a static mapping to R5 on R1’s main Serial0/0 interface.
- Configure a default route on R4 pointing to the VLAN 146 connection of R1.
- Configure a default route on R3 pointing to the point-to-point link connecting to R1.
- Configure a default route on R5 pointing to R1 out the Frame Relay network.
- Configure a route for R4’s Loopback0 network on R5 via their point-to-point Serial link.
- Configure a route for R5’s Loopback0 network on R4 via their point-to-point Serial link.
- Configure R1 and R5 to run CDP over the Frame Relay network with each other.
- Configure an IP SLA instance on R1 that pings R4’s connection to VLAN 146 every five seconds.
- Configure policy routing on R1 so that traffic from R3 going to R4’s Loopback0 network is sent to R5 over the Frame Relay link, while traffic to R5’s Loopback0 network is sent to R4 over VLAN 146.
- Use traceroute to verify that this configuration is functional.
- Modify the R1’s policy routing so that if R1 loses R5 as a CDP neighbor traffic from R3 to R4’s Loopback0 network is rerouted directly to R4.
- Modify the R1’s policy routing so that if R1 loses ICMP reachability to R4 traffic from R3 to R5’s Loopback0 network is rerouted directly to R5.
- Use traceroute to verify that this configuration is functional.

## 3.9 Local Policy Routing

- Create two access-lists named TO_R4 and TO_R5 on R1.
- Access-list TO_R4 should match all IP packets going to the Loopback0 network of R4, while access-list TO_R5 should match all packets going to the Loopback0 network of R5.
- Configure local policy-routing on R1 so that locally generated traffic matched by the list TO_R5 is routed out the Ethernet link to R4, and traffic matched by the access-list TO_R4 is routed out the Frame Relay link to R5.
- Use traceroute on R1 for R4 and R5's Loopback0 networks to verify that this configuration is functional.

## 3.10 GRE Tunneling

- Remove all previous static and policy routing configurations.
- Enable RIPv2 on all links in the 150.X.0.0 and 155.X.0.0 address space on all internal devices and disable auto-summary.
- Create a GRE tunnel between SW3 and SW4 using the IP addresses 10.34.0.Y/24.
- The tunnel should be sourced from and destined to these devices Loopback0 networks.
- Create new Loopback1 interfaces on SW3 and SW4 with the IP addresses 172.16.0.Y/32.
- Configure static routes so that traffic from SW3 to SW4's new Loopback1 interface is routed over the tunnel, and vice-versa.
- Use traceroute on SW3 and SW4 to verify that this configuration is functional.

## 3.11 GRE Tunneling and Recursive Routing

- Enable RIPv2 on the tunnel interface between SW3 and SW4.
- Configure distribute-list filtering on SW3 and SW4 so that a recursive routing error does not occur for routing lookups out the tunnel interface between them.

## 3.12 Reliable Backup Interface with GRE

- Remove all previous routing configurations.
- Configure static routes on R5 for R4's Loopback0 interface via both the Frame Relay and point-to-point connections between them.
- Configure static routes on R4 for R5's Loopback0 interface via both the Frame Relay and point-to-point connections between them.
- The static routes on R4 and R5 via the Frame Relay circuit should have a higher administrative distance than those out the point-to-point link.
- Create a GRE tunnel between R4 and R5 using the addresses 10.0.0.Y/24 that is sourced from and destined to the Frame Relay connection between them, and enable GRE keepalives.
- Configure the backup interface feature on R5 such that if the tunnel between R4 and R5 goes down the point-to-point Serial link between them is activated.
- To verify this configuration ensure that traffic between the Loopback0 interfaces of R4 and R5 is routed out the Frame Relay link between them, but if R4's Frame Relay interface is shut down this traffic is rerouted out the point-to-point link.

## 3.13 On-Demand Routing (ODR)

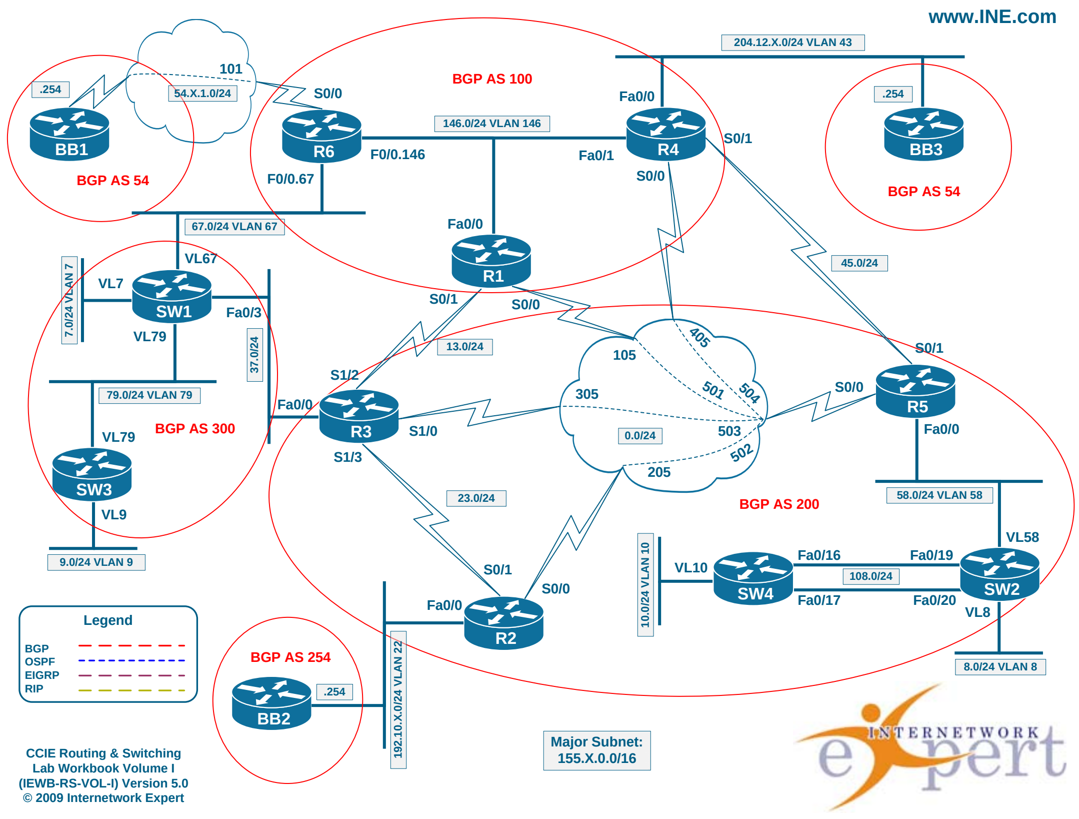
- Remove all previous routing configurations.
- Disable all physical interfaces on R1, R2, R3, R4, and R5 with the exception of their connections to the Frame Relay cloud.
- Ensure that CDP is enabled on the Frame Relay network between these devices.
- Enable ODR on R5.
- Ensure that all devices on this segment have IP reachability to each others' Loopback0 networks.

**Note**

Load the *OER* initial configurations prior to starting. The scenario presents AS 200 with 3 exits points to reach AS 100:

a) via the link between R4 and R5, with the link bandwidth of 128Kbps
b) via the link between R1 and R3, with the link bandwidth of 128Kbps
c) via the link between R3 and SW1, with the simulated bandwidth of 256Kbps (via traffic-shaping)

AS200 will be used as the network implementing Cisco OER to optimally utilize all exits points to AS 100 (two direct connection and one across AS 100).



## 3.14 OER Components Setup

- Configure R2 as the OER MC and R2, R3 and R5 as the Border Routers.
- Authenticate all OER communications using the password value of “CISCO”.
- Set up external and internal OER interfaces on all routers per the network topology presented.

- Enable OER events logging and change the keepalive interval to 5 seconds.

## 3.15 OER Profile Phase

- Configure the OER deployment to learn network prefixes based on maximum throughput and delay.
- The address information should be summarized based on the BGP table contents.
- Run periodic learning procedure every 5 minutes for the duration of 3 minutes and do not store more that 10 prefixes in the MTC.
- Only store the traffic flows corresponding to ICMP data exchange, WWW transactions and VoIP bearer traffic (UDP port range 16384-32767).
- Manually configure a prefix traffic class 112.0.0.0/24.

## 3.16 OER Measure Phase

- Use the combined monitoring mode and configured manual TCP connect probes for the IP addresses of R1, R4 and R6 Loopback0 interface.
- Configure the master controller to use active-monitoring only for the prefix 150.X.4.0/24.
- Configure the border routers so that any interface utilization changes are detected as fast as possible.

## 3.17 OER Apply Policy Phase

- Configure the network's performance policy to look for the best exit for the traffic classes.
- OER should generate an OOP event once any link utilization exceeds 80% or when the range between links utilization is above 10%.
- Any acceptable exit point should have relative loss of no higher that 3% and delay no higher than 200ms.
- Best exit should be selected based on the minimum external utilization, and then based on the lowest delay for the prefix-class.
- Reverse the above policy logic for the prefix 150.X.6.0/24 and change the acceptable delay to 100ms.
- All exit points with the performance metrics within 15% of the best exit should be considered equivalent.
- Ensure for periodic re-optimization occurring every minimal time interval.
- After an OOP event, the traffic class should stick to the new exit for no less than 10 minutes.

## 3.18 OER Control & Verify Phase

- Enable OER to control routes the network domain.
- Use the tag value of 1000 for injected static routes and BGP local preference value of 6000.
- Ensure that any injected static route will be redistributes into the IGP used by AS 200.

# IP Routing Solutions

## 3.1 Routing to Multipoint Broadcast Interfaces

- Configure R1 with a static route for R4's Loopback0 network pointing to the IP address of R4.
- Configure R1 with a static route for R6's Loopback0 network pointing to R1's Ethernet interface connecting to VLAN 146.
- Ensure that R1 can ping the Loopback0 interfaces of R4 and R6.
- Verify that R6 is running Proxy ARP by checking the ARP table of R1.
- Disable Proxy ARP on R6's connections to VLAN 146.
- Modify R1's ARP table so that it still has IP reachability to the Loopback0 interface of R6.

### *Configuration*

```
R1:
ip route 150.1.4.0 255.255.255.0 155.1.146.4
ip route 150.1.6.0 255.255.255.0 FastEthernet0/0
arp 150.1.6.6 0011.93da.bf40 arpa

R6:
interface FastEthernet0/0.146
 no ip proxy-arp
```

### *Verification*

**Note**

When routing to a next-hop value the router performs layer 2 to layer 3 resolution on the next-hop address. When routing to an interface the router performs layer 2 to layer 3 resolution on the final destination. In this particular case this means that R1 will use the MAC address of 155.1.146.4 to reach 150.1.4.4, but directly ARP for the address 150.1.6.6. Since R6 has Proxy ARP enabled by default on its interface connected to this segment it will respond to R1's ARP request with its own MAC address.

```
Rack1R1#show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  155.1.146.1             -   000e.83f8.0d00  ARPA   FastEthernet0/0
Internet  155.1.146.4             5   0011.20d2.4641  ARPA   FastEthernet0/0
Internet  155.1.146.6             5   0011.93da.bf40  ARPA   FastEthernet0/0
```

```
Rack1R1#ping 150.1.4.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.4.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Rack1R1#ping 150.1.6.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.6.6, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/2/4 ms

Rack1R1#show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  155.1.146.1             -   000e.83f8.0d00  ARPA   FastEthernet0/0
Internet  155.1.146.4             5   0011.20d2.4641  ARPA   FastEthernet0/0
Internet  155.1.146.6             5   0011.93da.bf40  ARPA   FastEthernet0/0
Internet  150.1.6.6               0   0011.93da.bf40  ARPA   FastEthernet0/0
```

Once Proxy ARP is disabled on R6, R1 cannot resolve the destination 150.1.6.6. This is seen from the *encapsulation failed* message R1 generates in the debug ip packet output. Encapsulation failed means that the router does not know the correct layer 2 address to use when building the layer 2 frame.

```
Rack1R1#clear arp

Rack1R1#debug arp
ARP packet debugging is on

Rack1R1#debug ip packet
IP packet debugging is on

Rack1R1#ping 150.1.6.6 repeat 1

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 150.1.6.6, timeout is 2 seconds:
.
Success rate is 0 percent (0/1)
IP: tableid=0, s=155.1.146.1 (local), d=150.1.6.6 (FastEthernet0/0),
routed via RIB
IP: s=155.1.146.1 (local), d=150.1.6.6 (FastEthernet0/0), len 100,
sending
IP ARP: creating incomplete entry for IP address: 150.1.6.6 interface
FastEthernet0/0
IP ARP: sent req src 155.1.146.1 000e.83f8.0d00,
                 dst 150.1.6.6 0000.0000.0000 FastEthernet0/0
IP: s=155.1.146.1 (local), d=150.1.6.6 (FastEthernet0/0), len 100,
encapsulation failed
```

There are two ways to resolve this problem, either change the IP routing on R1 so that it does not ARP for the final destination, or populate the ARP cache of R1 statically so that it knows which MAC address to use when it sends the packet to 150.1.6.6.

```
Rack1R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R1(config)#arp 150.1.6.6 0011.93da.bf40 arpa
Rack1R1(config)#end

Rack1R1#show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  155.1.146.1             -   000e.83f8.0d00  ARPA   FastEthernet0/0
Internet  155.1.146.4             4   0011.20d2.4641  ARPA   FastEthernet0/0
Internet  155.1.146.6             4   0011.93da.bf40  ARPA   FastEthernet0/0
Internet  150.1.6.6               -   0011.93da.bf40  ARPA

Rack1R1#ping 150.1.6.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.6.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms
```

The ideal solution for this problem design-wise is to never configure a static route to point out a multipoint interface.  Static routes should either point to the next-hop value of the neighbor on the multipoint interface, or point to and interface only if it is point-to-point, such as a GRE tunnel.

## 3.2 Routing to NBMA Interfaces

- Configure R1 and R2 with default routes pointing out their point-to-point Frame Relay interfaces connecting to R5.
- Configure R5 with a static route for R1’s Loopback0 network pointing to the IP address of R1.
- Configure R5 with a static route for R2’s Loopback0 network pointing out its main Frame Relay interface connecting to R2.
- Ensure that R1, R2, and R5 can all ping each other’s Loopback0 interfaces.

### *Configuration*

```
R1:
ip route 0.0.0.0 0.0.0.0 Serial0/0.1

R2:
ip route 0.0.0.0 0.0.0.0 Serial0/0.1

R5:
interface Serial0/0
 frame-relay map ip 150.1.2.2 502
!
ip route 150.1.1.0 255.255.255.0 155.1.0.1
ip route 150.1.2.0 255.255.255.0 Serial0/0
```

## *Verification*

**Note**

Similar to the previous routing example over Ethernet, when a static route points at an interface, layer 3 to layer 2 resolution is performed for the final destination, but if the route points at the intermediary next-hop, layer 3 to layer 2 resolution is performed for the next-hop. Since point-to-point interfaces do not require layer 3 to layer 2 resolution there is no functional difference between using the interface versus the next-hop value.

In this particular case R1 and R2 point routes directly out their point-to-point Frame Relay interfaces. Since these links do not require address resolution there is no problem with encapsulation. On R5 the route to R1's Loopback uses the next-hop value of R1, while the route to R2's Loopback uses the connected interface. This means that when R5 sends traffic to 150.1.1.1 it will use the DLCI that is associated with 155.1.0.1 (DLCI 501), but when it sends traffic to 150.1.2.2 it will look for the DLCI associated directly with 150.1.2.2. Since there is a Frame Relay mapping for R1's next-hop, but not R2's Loopback, only transport to R1's Loopback is successful.

```
Rack1R1#ping 150.1.5.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.5.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/64/88 ms

Rack1R2#ping 150.1.5.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.5.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/59/60 ms

Rack1R5#ping 150.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms
```

```
Rack1R5#debug ip packet
IP packet debugging is on

Rack1R5#debug frame-relay packet
Frame Relay packet debugging is on

Rack1R5#ping 150.1.2.2 repeat 1

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 150.1.2.2, timeout is 2 seconds:

IP: tableid=0, s=155.1.0.5 (local), d=150.1.2.2 (Serial0/0), routed via
RIB
IP: s=155.1.0.5 (local), d=150.1.2.2 (Serial0/0), len 100, sending
Serial0/0:Encaps failed--no map entry link 7(IP)
IP: s=155.1.0.5 (local), d=150.1.2.2 (Serial0/0), len 100,
encapsulation failed.
Success rate is 0 percent (0/1)
```

The output *no map entry link* means that R5 does not know which PVC to use to get to 150.1.2.2 because there is no associated **`frame-relay map`** statement. Once the Frame Relay mapping table includes this resolution, encapsulation is successful.

```
Rack1R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/0
Rack1R5(config-if)#frame-relay map ip 150.1.2.2 502
Rack1R5(config-if)#end

Rack1R5#ping 150.1.2.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/61 ms
```

This is another design example as to why static routes should not point out multipoint interfaces.  If R5 were to configure a default route pointing directly to Serial0/0 it would mean that there would need to be a matching **`frame-relay map`** statement for every possible destination.  On the other hand if the default route pointed towards R1's next-hop value, R5 would only need a **`frame-relay map`** statement for R1's next-hop.

## 3.3 Longest Match Routing

- Configure R4 and R5 with routes to each other's Loopback0 interfaces via the point-to-point Serial interface between them.
- Configure R4 and R5 to route the rest of the 150.X.0.0/16 network via the Frame Relay connection between them.
- Ensure that traffic between R4 and R5's Loopback0 networks transits the point-to-point link between them, but is rerouted out the Frame Relay connection if the point-to-point link is down.

### *Configuration*

```
R4:
ip route 150.1.5.0 255.255.255.0 Serial0/1
ip route 150.1.0.0 255.255.0.0 155.1.0.5

R5:
ip route 150.1.4.0 255.255.255.0 Serial0/1
ip route 150.1.0.0 255.255.0.0 155.1.0.4
```

### *Verification*

**Note**

IPv4 routing logic uses *longest match* routing to determine which entry to use from the routing table for forwarding.  This principle can be used both to achieve redundancy and traffic engineering.

In this particular example traffic engineering is accomplished for traffic routed between the Loopback0 networks of R4 and R5.  When R5 does a lookup on the final destination 150.1.4.4, the longest match is 150.1.4.0/24 out Serial0/1.  While technically there are multiple routes to this destination, 150.1.4.0/24 and 150.1.0.0/16, the /24 route wins.  For any other 150.1.0.0 destinations, such as 150.1.3.3, the longest match will be 150.1.0.0 via 155.1.0.4.  This allows traffic for a portion of the IP address space to be routed one way, and traffic for a different portion of the IP address space to be routed another way.

```
Rack1R5#show ip route 150.1.4.4
Routing entry for 150.1.4.0/24
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks:
  * directly connected, via Serial0/1
      Route metric is 0, traffic share count is 1
```

```
Rack1R5#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.45.4 16 msec *  12 msec
```

Redundancy is accomplished in this example when the point-to-point link between R4 and R5 fails. As long as this link is up and installed in the routing table, the route 150.1.4.0/24 can be installed in the routing table. However if the Serial0/1 link is up/down or down/down it cannot be installed in the routing table, and any routes that recurse to Serial0/1 cannot be installed in the routing table. The result of this is that when the link is down traffic between the two Loopbacks of R4 and R5 is rerouted out the Frame Relay link using the prefix 150.1.0.0/16 as the longest match.

```
Rack1R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/1
Rack1R5(config-if)#shutdown
Rack1R5(config-if)#end
Rack1R5#

Rack1R5#show ip route 150.1.4.4
Routing entry for 150.1.0.0/16
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 155.1.0.4
      Route metric is 0, traffic share count is 1

Rack1R5#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.0.4 32 msec *  28 msec
```

## 3.4 Floating Static Routes

- Remove the previous static routes on R4 and R5.
- Configure R4 and R5 with identical static routes for each others' Loopback0 networks out both their Frame Relay connections and the point-to-point connection between them.
- Using administrative distance ensure that traffic between R4 and R5's Loopback0 networks transits the point-to-point link between them, but is rerouted out the Frame Relay connection if the point-to-point link is down.

### *Configuration*

```
R4:
ip route 150.1.5.0 255.255.255.0 Serial0/1 10
ip route 150.1.5.0 255.255.255.0 155.1.0.5 20

R5:
ip route 150.1.4.0 255.255.255.0 Serial0/1 10
ip route 150.1.4.0 255.255.255.0 155.1.0.4 20
```

### *Verification*

**Note**

When the router does a lookup for a destination and there are multiple routes with the same longest match, the administrative distance is compared. The route with the lower administrative distance is installed in the routing table. With static routing this principle can be used for simple redundancy by configuring a backup route with a higher administrative distance than the primary route.

In this example R5 installs the route to 150.1.4.0/24 via Serial0/1 with an administrative distance of 10. When the link Serial0/1 is down the route with the next lowest administrative distance, 150.1.4.0/24 via 155.1.0.4 with a distance of 20, is installed. The result is that traffic is routed out the point-to-point link unless it is down, in which case traffic is rerouted out the Frame Relay link.

```
Rack1R5#show ip route 150.1.4.4
Routing entry for 150.1.4.0/24
  Known via "static", distance 10, metric 0 (connected)
  Routing Descriptor Blocks:
  * directly connected, via Serial0/1
      Route metric is 0, traffic share count is 1

Rack1R5#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.45.4 12 msec *  12 msec

Rack1R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/1
Rack1R5(config-if)#shutdown
Rack1R5(config-if)#end
Rack1R5#show ip route 150.1.4.4
Routing entry for 150.1.4.0/24
  Known via "static", distance 20, metric 0
  Routing Descriptor Blocks:
  * 155.1.0.4
      Route metric is 0, traffic share count is 1

Rack1R5#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.0.4 28 msec *  28 msec
```

## 3.5 Backup Interface

- Modify the administrative distance of R4’s static routes to R5’s Loopback0 interface to be identical out the Frame Relay and point-to-point interfaces.
- Configure the backup interface feature on R4 to route traffic for R5’s Loopback0 out the Frame Relay network unless the PVC to R5 is down.
- If the PVC to R5 is down this traffic should be rerouted out the point-to-point link.
- Ensure the backup link is activate 3 seconds after the main link fails, and deactivated once the main link is active for 60 seconds.

### *Configuration*

```
R4:
ip route 150.1.5.0 255.255.255.0 Serial0/1
ip route 150.1.5.0 255.255.255.0 155.1.0.5
!
interface Serial0/0.1 point-to-point
 backup delay 3 60
 backup interface Serial0/1
```

### *Verification*

**Note**

In this example R4 uses the backup interface feature along with duplicate routing information to perform both traffic engineering and redundancy. With the backup interface configured on R4’s point-to-point subinterface to R5, R4 waits for the line protocol of interface Serial0/0.1 to go down before interface Serial0/1 is activated. Since the line protocol status of Serial0/0.1 is based on the PVC status of DLCI 405, any failure in the end-to-end LMI path will notify R4 of a problem, and activate the backup link. Once the backup link is up, the route to 150.1.5.0/24 can be installed via Serial0/1.

```
Rack1R4#show ip interface brief
Interface           IP-Address      OK? Method Status                Protocol
FastEthernet0/0     204.12.1.4      YES manual up                    up
Serial0/0           unassigned      YES unset  up                    up
Serial0/0.1         155.1.0.4       YES manual up                    up
FastEthernet0/1     155.1.146.4     YES manual up                    up
Serial0/1           155.1.45.4      YES manual standby mode          down
Loopback0           150.1.4.4       YES manual up                    up

Rack1R4#show backup
Primary Interface   Secondary Interface   Status
-----------------   -------------------   ------
Serial0/0.1         Serial0/1             normal operation
```

```
Rack1R4#show ip route 150.1.5.5
Routing entry for 150.1.5.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 155.1.0.5
      Route metric is 0, traffic share count is 1

Rack1R4#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.0.5 28 msec *  28 msec
```

Once the circuit status for PVC 405 on R4 changes to INACTIVE, the line protocol state of Serial0/0.1 is down. Three seconds later the backup interface is taken out of standby mode, and the new routing information is installed in the routing table.

```
Rack1R4#debug backup
Backup events debugging is on

Rack1R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/0
Rack1R5(config-if)#shutdown
Rack1R5(config-if)#

Rack1R4#
Jun 30 11:59:20: BACKUP(Serial0/0.1): event = primary interface went down
Jun 30 11:59:20: BACKUP(Serial0/0.1): changed state to "waiting to backup"
Jun 30 11:59:23: BACKUP(Serial0/0.1): event = timer expired on primary
Jun 30 11:59:23: BACKUP(Serial0/0.1): secondary interface (Serial0/1) made
active
Jun 30 11:59:23: BACKUP(Serial0/0.1): changed state to "backup mode"
%LINK-3-UPDOWN: Interface Serial0/1, changed state to up
Jun 30 11:59:25: BACKUP(Serial0/1): event = secondary interface came up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed state to up
Jun 30 11:59:26: BACKUP(Serial0/1): event = secondary interface came up

Rack1R4#show ip route 150.1.5.5
Routing entry for 150.1.5.0/24
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks:
  * directly connected, via Serial0/1
      Route metric is 0, traffic share count is 1

Rack1R4#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.45.5 16 msec *  12 msec
```

Once the circuit status for PVC 405 on R4 changes to ACTIVE, the line protocol state of Serial0/0.1 returns to up. Sixty seconds later the backup interface is sent back into standby state. If dynamic routing were configured in this topology the backup delay would allow time for the routing domain to reconverge before failing back over to the primary link.

```
Rack1R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/0
Rack1R5(config-if)#no shutdown
Rack1R5(config-if)#

Rack1R4#
Jun 30 12:00:10: BACKUP(Serial0/0.1): event = primary interface came up
Jun 30 12:00:10: BACKUP(Serial0/0.1): changed state to "waiting to revert"
Jun 30 12:01:10: BACKUP(Serial0/0.1): event = timer expired on primary
Jun 30 12:01:10: BACKUP(Serial0/0.1): secondary interface (Serial0/1) moved to
standby
Jun 30 12:01:10: BACKUP(Serial0/0.1): changed state to "normal operation"
%LINK-5-CHANGED: Interface Serial0/1, changed state to standby mode
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed state to
down
Jun 30 12:01:13: BACKUP(Serial0/1): event = secondary interface went down

Rack1R4#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.0.5 32 msec *  28 msec
```

## 3.6 Reliable Static Routing with Enhanced Object Tracking

- Remove all previous routing configurations.
- Configure static routes on R5 so that it has reachability to the Loopback0 networks of R1 and R4 via the Frame Relay network.
- Configure R1 with a static route to R4’s Loopback0 network via the Frame Relay connection to R5.
- Configure R4 with a static route to R1’s Loopback0 network via the Ethernet connection to VLAN 146.
- Configure a second static route on R4 for R1’s Loopback0 network via the Frame Relay connection to R5 with an administrative distance of 2.
- Configure an IP SLA instance on R4 to ping R1’s Ethernet connection to VLAN 146 every 5 seconds.
- Create an enhanced object to track this SLA instance.
- Tie this enhanced object to the static route R4 has for R1’s Loopback0 out the connection to VLAN 146.
- Ensure that R4 maintains IP reachability to R1’s Loopback0 in the case that R1’s Ethernet interface is down.

### *Configuration*

```
R1:
ip route 150.1.4.0 255.255.255.0 155.1.0.4

R4:
ip sla monitor 1
 type echo protocol ipIcmpEcho 155.1.146.1 source-interface
FastEthernet0/1
 timeout 2000
 frequency 5
ip sla monitor schedule 1 start-time now
!
track 1 rtr 1
!
ip route 150.1.1.0 255.255.255.0 155.1.146.1 track 1
ip route 150.1.1.0 255.255.255.0 155.1.0.5 2

R5:
ip route 150.1.1.0 255.255.255.0 155.1.0.1
ip route 150.1.4.0 255.255.255.0 155.1.0.4
```

## *Verification*

**Note**

Although R1 and R4 are on the same layer 2 segment for VLAN 146, their physical Ethernet interfaces are not on the same layer 1 network. This means that the link status of R1's connection to VLAN 146 is independent of R4's connection. From a redundancy design point of view the possible problem with this is that R4 has no way to know whether or not R1's link is actually up. In point-to-point designs, like the previous examples with R4 and R5's directly connected Serial link, this is not an issue because the line protocol status of one side of the link dictates the line protocols status of the other side.

In the below example we can see that even though R1's link to VLAN 146 is down, R4 still installs the route to 150.1.1.0/24 via 155.1.146.1, since R4 has no way to know if 155.1.146.1 is available. The result of this problem is that if R1's link to VLAN 146 goes down traffic between the Loopbacks of R1 and R4 is dropped.

```
Rack1R1#ping 150.1.4.4 source 150.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.4.4, timeout is 2 seconds:
Packet sent with a source address of 150.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms

Rack1SW1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1SW1(config)#interface Fa0/1
Rack1SW1(config-if)#shutdown
Rack1SW1(config-if)#

Rack1R4#show ip route 150.1.1.1
Routing entry for 150.1.1.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 155.1.146.1
      Route metric is 0, traffic share count is 1

Rack1R1#ping 150.1.4.4 source 150.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.4.4, timeout is 2 seconds:
Packet sent with a source address of 150.1.1.1
.....
```

In order to fix this design the IP Service Level Agreement (SLA) and Enhanced Object Tracking features are introduced. First, R4 creates an SLA instance that sends an ICMP ping to R1 via VLAN 146 every five seconds. If a response is not received within 2000 milliseconds the SLA instance reports its status as down.

```
Rack1R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R4(config)#ip sla monitor 1
Rack1R4(config-sla-monitor)#type echo protocol ipIcmpEcho 155.1.146.1
source-interface Fa0/1
Rack1R4(config-sla-monitor-echo)#frequency 5
Rack1R4(config-sla-monitor-echo)#timeout 2000
Rack1R4(config-sla-monitor-echo)#exit
Rack1R4(config)#ip sla monitor schedule 1 start now
Rack1R4(config)#end

Rack1R4#show ip sla monitor statistics
Round trip time (RTT)   Index 1
        Latest RTT: 1 ms
Latest operation start time: 12:38:33.673 UTC Mon Jun 30 2008
Latest operation return code: OK
Number of successes: 201
Number of failures: 10
Operation time to live: 2546 sec
```

Next an Enhanced Object Tracking is created that watches the IP SLA instance. This tracking says that if SLA instance 1 is down, tracking instance 1 is down, and the static route to R1's Loopback cannot be installed in the routing table.

```
Rack1R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R4(config)#track 1 rtr 1
Rack1R4(config-track)#exit
Rack1R4(config)#no ip route 150.1.1.0 255.255.255.0 155.1.146.1
Rack1R4(config)#ip route 150.1.1.0 255.255.255.0 155.1.146.1 track 1
Rack1R4(config)#end
Rack1R4#show track 1
Track 1
  Response Time Reporter 1 state
  State is Up
    3 changes, last change 00:15:06
  Latest operation return code: OK
  Latest RTT (millisecs) 1
  Tracked by:
    STATIC-IP-ROUTING 0
```

The same failure scenario is re-introduced, however with the reliable static route R4 can now reroute around this network failure.

```
Rack1R4#traceroute 150.1.1.1 source 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.1.1

  1 155.1.146.1 36 msec *  36 msec

Rack1R4#debug track

Rack1R4#debug ip routing
IP routing debugging is on

Rack1SW1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1SW1(config)#interface Fa0/1
Rack1SW1(config-if)#shutdown
Rack1SW1(config-if)#

Rack1R4#
Track: 1 Change #2 rtr 1, state Up->Down
RT: del 150.1.1.0/24 via 155.1.146.1, static metric [1/0]
RT: delete subnet route to 150.1.1.0/24
RT: NET-RED 150.1.1.0/24
RT: SET_LAST_RDB for 150.1.1.0/24
  NEW rdb: via 155.1.0.5

RT: add 150.1.1.0/24 via 155.1.0.5, static metric [2/0]
RT: NET-RED 150.1.1.0/24

Rack1R4#traceroute 150.1.1.1 source 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.1.1

  1 155.1.0.5 28 msec 28 msec 28 msec
  2 155.1.0.1 56 msec *  56 msec

Rack1R4#show ip route 150.1.1.1
Routing entry for 150.1.1.0/24
  Known via "static", distance 2, metric 0
  Routing Descriptor Blocks:
  * 155.1.0.5
      Route metric is 0, traffic share count is 1
```

Once R1’s connection to VLAN 146 comes back, the SLA instance reports itself as back up, the tracking instance reports itself as back up, and the static route with the lower administrative distance is reinstalled in the routing table.

```
Rack1SW1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1SW1(config)#interface Fa0/1
Rack1SW1(config-if)#no shutdown
Rack1SW1(config-if)#

Rack1R4#
Track: 1 Change #3 rtr 1, state Down->Up
RT: closer admin distance for 150.1.1.0, flushing 1 routes
RT: SET_LAST_RDB for 150.1.1.0/24
  NEW rdb: via 155.1.146.1

RT: add 150.1.1.0/24 via 155.1.146.1, static metric [1/0]
RT: NET-RED 150.1.1.0/24

Rack1R4#show ip route 150.1.1.1
Routing entry for 150.1.1.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 155.1.146.1
      Route metric is 0, traffic share count is 1

Rack1R4#traceroute 150.1.1.1 source 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.1.1

  1 155.1.146.1 40 msec *  36 msec
```

## 3.7 Policy Routing

- Remove all previous routing configuration.
- Configure default routes on R4 and R6 pointing to the VLAN 146 connection of R1.
- Configure a default route on R3 pointing to the point-to-point link connecting to R1.
- Configure a default route on R5 pointing to R1 out the Frame Relay network.
- Configure a static route on R3 for R5's Loopback0 network and a static route on R5 for R3's Loopback0 network out the Frame Relay network.
- Create two extended access-lists on R1, one named FROM_R4 and the other named FROM_R6.
- Access-list FROM_R4 should match all IP packets coming from R4, while access-list FROM_R6 should match all IP packets coming from R6.
- Configure policy-routing on R1 so that traffic from R4 is routed out the serial link between R1 and R3, and traffic from R6 is routed out the Frame Relay link to R5.
- Use traceroute on R4 and R6 for R3 and R5's Loopback0 networks to verify that this configuration is functional.

### *Configuration*

```
R1:
interface FastEthernet0/0
 ip policy route-map POLICY_ROUTING
!
ip access-list extended FROM_R4
 permit ip host 155.1.146.4 any
ip access-list extended FROM_R6
 permit ip host 155.1.146.6 any
!
route-map POLICY_ROUTING permit 10
 match ip address FROM_R4
 set ip next-hop 155.1.13.3
!
route-map POLICY_ROUTING permit 20
 match ip address FROM_R6
 set ip next-hop 155.1.0.5

R3:
ip route 0.0.0.0 0.0.0.0 155.1.13.1
ip route 150.1.5.0 255.255.255.0 155.1.0.5

R4:
ip route 0.0.0.0 0.0.0.0 155.1.146.1
```

```
R5:
ip route 0.0.0.0 0.0.0.0 155.1.0.1
ip route 150.1.3.0 255.255.255.0 155.1.0.3

R6:
ip route 0.0.0.0 0.0.0.0 155.1.146.1
```

### *Verification*

**Note**

Policy routing allows the router to forward traffic based on user-defined criteria before the normal IP routing table is consulted. In this example we can see that R1 does not have routing information for either of the Loopbacks of R3 and R5, so it cannot send traffic there when it is locally generated.

```
Rack1R1#show ip route 150.1.3.3
% Subnet not in table

Rack1R1#show ip route 150.1.5.5
% Subnet not in table

Rack1R1#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 0 percent (0/5)

Rack1R1#ping 150.1.5.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.5.5, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

If traffic comes in from R4 or R6’s IP addresses attached on VLAN 146 it is routed according to the route-map attached to the incoming Fa0/0 interface.

```
Rack1R1#debug ip policy
Policy routing debugging is on

Rack1R4#traceroute 150.1.3.3

Type escape sequence to abort.
Tracing the route to 150.1.3.3

  1 155.1.146.1 4 msec 4 msec 0 msec
  2 155.1.13.3 16 msec *  16 msec

Rack1R1#
IP: s=155.1.146.4 (FastEthernet0/0), d=150.1.3.3, len 28, FIB policy match
IP: s=155.1.146.4 (FastEthernet0/0), d=150.1.3.3, g=155.1.13.3, len 28, FIB
policy routed

Rack1R4#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.146.1 0 msec 0 msec 4 msec
  2 155.1.13.3 16 msec 20 msec 16 msec
  3 155.1.0.5 36 msec *  36 msec

Rack1R1#
IP: s=155.1.146.4 (FastEthernet0/0), d=150.1.5.5, len 28, FIB policy match
IP: s=155.1.146.4 (FastEthernet0/0), d=150.1.5.5, g=155.1.13.3, len 28, FIB
policy routed

Rack1R6#traceroute 150.1.3.3

Type escape sequence to abort.
Tracing the route to 150.1.3.3

  1 155.1.146.1 0 msec 4 msec 0 msec
  2 155.1.0.5 28 msec 32 msec 33 msec
  3 155.1.0.3 32 msec *  32 msec

Rack1R1#
IP: s=155.1.146.6 (FastEthernet0/0), d=150.1.3.3, len 28, FIB policy match
IP: s=155.1.146.6 (FastEthernet0/0), d=150.1.3.3, g=155.1.0.5, len 28, FIB
policy routed

Rack1R6#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.146.1 0 msec 4 msec 0 msec
  2 155.1.0.5 28 msec *  28 msec

Rack1R1#
IP: s=155.1.146.6 (FastEthernet0/0), d=150.1.5.5, len 28, FIB policy match
IP: s=155.1.146.6 (FastEthernet0/0), d=150.1.5.5, g=155.1.0.5, len 28, FIB
policy routed
```

R1's route-map used for policy routing does not match traffic sourced from R6's Loopback0 interface, so this traffic is dropped when it is received in the link to VLAN 146.

```
Rack1R6#ping 150.1.5.5 source 150.1.6.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.5.5, timeout is 2 seconds:
Packet sent with a source address of 150.1.6.6
.....
Success rate is 0 percent (0/5)

Rack1R1#
IP: s=150.1.6.6 (FastEthernet0/0), d=150.1.5.5, len 28, FIB policy rejected(no
match) - normal forwarding
IP: s=150.1.6.6 (FastEthernet0/0), d=150.1.5.5, len 28, policy rejected --
normal forwarding
```

## 3.8 Reliable Policy Routing

- Remove the previous static and policy routing configurations.
- Remove the point-to-point Frame Relay subinterface on R1 and replace this with a static mapping to R5 on R1's main Serial0/0 interface.
- Configure a default route on R4 pointing to the VLAN 146 connection of R1.
- Configure a default route on R3 pointing to the point-to-point link connecting to R1.
- Configure a default route on R5 pointing to R1 out the Frame Relay network.
- Configure a route for R4's Loopback0 network on R5 via their point-to-point Serial link.
- Configure a route for R5's Loopback0 network on R4 via their point-to-point Serial link.
- Configure R1 and R5 to run CDP over the Frame Relay network with each other.
- Configure an IP SLA instance on R1 that pings R4's connection to VLAN 146 every five seconds.
- Configure policy routing on R1 so that traffic from R3 going to R4's Loopback0 network is sent to R5 over the Frame Relay link, while traffic to R5's Loopback0 network is sent to R4 over VLAN 146.
- Use traceroute to verify that this configuration is functional.
- Modify the R1's policy routing so that if R1 loses R5 as a CDP neighbor traffic from R3 to R4's Loopback0 network is rerouted directly to R4.
- Modify the R1's policy routing so that if R1 loses ICMP reachability to R4 traffic from R3 to R5's Loopback0 network is rerouted directly to R5.
- Use traceroute to verify that this configuration is functional.

## *Configuration*

```
R1:
ip sla monitor 1
 type echo protocol ipIcmpEcho 155.1.146.4 source-interface
FastEthernet0/0
 timeout 2000
 frequency 5
!
ip sla monitor schedule 1 life forever start-time now
!
track 1 rtr 1
!
interface Serial0/0
 ip address 155.1.0.1 255.255.255.0
 encapsulation frame-relay
 cdp enable
 frame-relay map ip 155.1.0.5 105 broadcast
 no frame-relay inverse-arp
!
interface Serial0/1
 ip policy route-map RELIABLE_POLICY_ROUTING
!
ip access-list extended FROM_R3_TO_R4_LOOPBACK
 permit ip host 155.1.13.3 host 150.1.4.4
!
ip access-list extended FROM_R3_TO_R5_LOOPBACK
 permit ip host 155.1.13.3 host 150.1.5.5
!
route-map RELIABLE_POLICY_ROUTING permit 10
 match ip address FROM_R3_TO_R4_LOOPBACK
 set ip next-hop 155.1.0.5
 set ip next-hop verify-availability
 set ip default next-hop 155.1.146.4
!
route-map RELIABLE_POLICY_ROUTING permit 20
 match ip address FROM_R3_TO_R5_LOOPBACK
 set ip next-hop verify-availability 155.1.146.4 1 track 1
 set ip default next-hop 155.1.0.5

R3:
ip route 0.0.0.0 0.0.0.0 155.1.13.1

R4:
ip route 0.0.0.0 0.0.0.0 155.1.146.1
ip route 150.1.5.0 255.255.255.0 155.1.45.5

R5:
interface Serial0/0
 cdp enable
!
ip route 0.0.0.0 0.0.0.0 155.1.0.1
ip route 150.1.5.0 255.255.255.0 155.1.45.4
```

## *Verification*

**Note**

In this example of policy routing R1 matches traffic as it comes in Serial0/1 from R3, and routes the traffic based on both the source and the destination. The traceroute output on R3 indicates that the traffic is policy routed correctly.

```
Rack1R3#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.13.1 16 msec 16 msec 12 msec
  2 155.1.0.5 40 msec 44 msec 40 msec
  3 155.1.45.4 32 msec *  32 msec

Rack1R3#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.13.1 16 msec 16 msec 16 msec
  2 155.1.146.4 16 msec 20 msec 16 msec
  3 155.1.45.5 40 msec *  36 msec
```

Since R1's policy routing configuration is only locally significant, network failures do not automatically update the routing policy of R1. As seen below if either R5's connection to the Frame Relay network goes down or R4's connection to VLAN 146 goes down, traffic will be blackholed. This is based on the fact that the next-hop values of 155.1.0.5 and 155.1.146.4 can still be recursed in the routing table, because the interfaces R1 uses to reach them do not go down when the other side of the link fails.

```
Rack1R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/0
Rack1R5(config-if)#shutdown
Rack1R5(config-if)#

Rack1R3#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.13.1 16 msec 12 msec 12 msec
  2  *  *  *
```

```
Rack1R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/0
Rack1R5(config-if)#no shutdown
Rack1R5(config-if)#

Rack1R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R4(config)#interface FastEthernet0/1
Rack1R4(config-if)#shutdown
Rack1R4(config-if)#

Rack1R3#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.13.1 16 msec 16 msec 16 msec
  2  *  *  *
```

In order to resolve this design problem R1 needs some way to track end-to-end reachability on these links used for the outbound forwarding through policy routing. The two ways illustrated in this example are through the IP SLA and Enhanced Object Tracking features, and through CDP.

With IP SLA configured R1 tracks the end-to-end circuit status of VLAN 146 through ICMP ping. When R4's connection to VLAN 146 goes down, R1's SLA instance reports its status down, which in turn causes the tracked object to do down. The tracked object is called from the route-map syntax **`set ip next-hop verify-availability 155.1.146.4 1 track 1`**. This means that if tracked object 1 goes down, do not use the next-hop 155.1.146.4. Instead this route-map sequence fails over to the "default" next-hop of 155.1.0.5. The result of this is seen through the below traceroute of R3 and debug ip policy output of R1.

```
Rack1R1#debug track
Rack1R1#debug ip policy
Policy routing debugging is on

Rack1R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R4(config)#interface FastEthernet0/1
Rack1R4(config-if)#shutdown
Rack1R4(config-if)#

Rack1R1#
Track: 1 Change #2 rtr 1, state Up->Down
```

```
Rack1R3#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.13.1 16 msec 16 msec 16 msec
  2 155.1.0.5 44 msec *  40 msec

Rack1R1#
IP: s=155.1.13.3 (Serial0/1), d=150.1.5.5, len 28, FIB policy match
IP: s=155.1.13.3 (Serial0/1), d=150.1.5.5, g=155.1.0.5, len 28, FIB
policy routed
```

With CDP tracking for policy routing R1 looks into the CDP table to see if there is a neighbor installed with the IP address that matches the next-hop value being set in the route-map. In this case the syntax **`set ip next-hop 155.1.0.5`** , **`set ip next-hop verify-availability`** and **`set ip default next-hop 155.1.146.4`** means if there is no CDP neighbor with the IP address 155.1.0.5 traffic that matches this sequence will be routed to 155.1.146.4. This can be seen from the below output when R5's Frame Relay interface fails.

```
Rack1R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R4(config)#interface FastEthernet0/1
Rack1R4(config-if)#no shutdown
Rack1R4(config-if)#

Rack1R1#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Rack1SW1         Fas 0/0            155          S I      WS-C3560- Fas 0/1
Rack1R3          Ser 0/1            152         R S I     2611XM    Ser 1/2
Rack1R5          Ser 0/0            102         R S I     2611XM    Ser 0/0

Rack1R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/0
Rack1R5(config-if)#shutdown
Rack1R5(config-if)#

Rack1R1#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Rack1SW1         Fas 0/0            137          S I      WS-C3560- Fas 0/1
Rack1R3          Ser 0/1            134         R S I     2611XM    Ser ½
```

```
Rack1R3#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.13.1 16 msec 12 msec 12 msec
  2 155.1.146.4 20 msec *  16 msec

Rack1R1#
IP: s=155.1.13.3 (Serial0/1), d=150.1.4.4, len 28, FIB policy match
IP: s=155.1.13.3 (Serial0/1), d=150.1.4.4, g=155.1.146.4, len 28, FIB
policy routed
```

## 3.9 Local Policy Routing

- Create two access-lists named TO_R4 and TO_R5 on R1.
- Access-list TO_R4 should match all IP packets going to the Loopback0 network of R4, while access-list TO_R5 should match all packets going to the Loopback0 network of R5.
- Configure local policy-routing on R1 so that locally generated traffic matched by the list TO_R5 is routed out the Ethernet link to R4, and traffic matched by the access-list TO_R4 is routed out the Frame Relay link to R5.
- Use traceroute on R1 for R4 and R5's Loopback0 networks to verify that this configuration is functional.

### *Configuration*

```
R1:
ip local policy route-map LOCAL_POLICY
!
ip access-list extended TO_R4
 permit ip any host 150.1.4.4
!
ip access-list extended TO_R5
 permit ip any host 150.1.5.5
!
route-map LOCAL_POLICY permit 10
 match ip address TO_R4
 set ip next-hop 155.1.0.5
!
route-map LOCAL_POLICY permit 20
 match ip address TO_R5
 set ip next-hop 155.1.146.4
```

### *Verification*

**Note**

Local policy routing is similar in operation to normal policy routing, except it affects locally generated traffic from the router instead of traffic received inbound on an interface. In the below output we can see that R1 does not have a normal route to either of the destinations 150.1.4.4 or 150.1.5.5, however traffic is successfully routed due to the locally configured policy.

```
Rack1R1#show ip route 150.1.4.4
% Subnet not in table
```

```
Rack1R1#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.0.5 32 msec 28 msec 32 msec
  2 155.1.45.4 16 msec *  16 msec

Rack1R1#show ip route 150.1.5.5
% Subnet not in table

Rack1R1#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.146.4 4 msec 4 msec 0 msec
  2 155.1.45.5 28 msec *  24 msec
```

## ☠ Pitfall

Note that when the remote devices receive traffic from R1 it is sourced from the Loopback0 interface of R1. Normally the router uses the IP address of the outgoing interface in the routing table as the source IP address in its own packets. However since the routing table is not consulted for the lookup you may see inconsistencies in what the source address of the local traffic is. This behavior could have a negative impact on protocols such as BGP which need to agree on what the source and destination IP addresses are for a peering.

```
Rack1R5#debug ip icmp
ICMP packet debugginet debugging is on

Rack1R1#ping 150.1.5.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.5.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 44/44/45 ms

Rack1R5#
Jun 30 16:59:40.118: ICMP: echo reply sent, src 150.1.5.5, dst 150.1.1.1
Jun 30 16:59:40.166: ICMP: echo reply sent, src 150.1.5.5, dst 150.1.1.1
Jun 30 16:59:40.210: ICMP: echo reply sent, src 150.1.5.5, dst 150.1.1.1
Jun 30 16:59:40.254: ICMP: echo reply sent, src 150.1.5.5, dst 150.1.1.1
Jun 30 16:59:40.302: ICMP: echo reply sent, src 150.1.5.5, dst 150.1.1.1
```

## 3.10 GRE Tunneling

- Remove all previous static and policy routing configurations.
- Enable RIPv2 on all links in the 150.X.0.0 and 155.X.0.0 address space on all internal devices and disable auto-summary.
- Create a GRE tunnel between SW3 and SW4 using the IP addresses 10.34.0.Y/24.
- The tunnel should be sourced from and destined to these devices Loopback0 networks.
- Create new Loopback1 interfaces on SW3 and SW4 with the IP addresses 172.16.0.Y/32.
- Configure static routes so that traffic from SW3 to SW4's new Loopback1 interface is routed over the tunnel, and vice-versa.
- Use traceroute on SW3 and SW4 to verify that this configuration is functional.

### *Configuration*

```
R1 – R6:
router rip
 version 2
 no auto-summary
 network 150.1.0.0
 network 155.1.0.0

SW1 & SW2:
ip routing
!
router rip
 version 2
 no auto-summary
 network 150.1.0.0
 network 155.1.0.0
```

```
SW3:
ip routing
!
interface Tunnel0
 ip address 10.34.0.9 255.255.255.0
 tunnel source Loopback0
 tunnel destination 150.1.10.10
!
interface Loopback1
 ip address 172.16.0.9 255.255.255.255
!
ip route 172.16.0.10 255.255.255.255 Tunnel0
!
router rip
 version 2
 no auto-summary
 network 150.1.0.0
 network 155.1.0.0

SW4:
ip routing
!
interface Tunnel0
 ip address 10.34.0.10 255.255.255.0
 tunnel source Loopback0
 tunnel destination 150.1.9.9
!
interface Loopback1
 ip address 172.16.0.10 255.255.255.255
!
ip route 172.16.0.9 255.255.255.255 Tunnel0
!
router rip
 version 2
 no auto-summary
 network 150.1.0.0
 network 155.1.0.0
```

## *Verification*

**Note**

Generic Routing Encapsulation (GRE) tunneling is used to take another protocol payload, such as IPv4, IPv6, IPX, etc., and tunnel it over an IPv4 transit network. In this case GRE is used to tunnel IPv4 packets between the 172.16.0.9/32 and 172.16.0.10/32 segments.

We can verify that traffic between these segments is going out the tunnel because the traceroute output shows only one hop between the networks, which is the tunnel.

```
Rack1SW4#show ip route 172.16.0.9
Routing entry for 172.16.0.9/32
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks:
  * directly connected, via Tunnel0
      Route metric is 0, traffic share count is 1

Rack1SW4#traceroute 172.16.0.9

Type escape sequence to abort.
Tracing the route to 172.16.0.9

  1 10.34.0.9 40 msec *  40 msec
```

The second verification is in the transit path. From the output of **`debug ip packet detail`** we can see that R5 is forwarding IP protocol number 47 between SW3 and SW4, which is GRE.

```
Rack1R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/0
Rack1R5(config-if)#no ip route-cache
Rack1R5(config-if)#interface FastEthernet0/0
Rack1R5(config-if)#no ip route-cache
Rack1R5(config-if)#end
Rack1R5#debug ip packet detail
IP packet debugging is on (detailed)
Rack1R5#

Rack1SW4#ping 172.16.0.9

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.9, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 76/76/80 ms

Rack1R5#
IP: s=150.1.10.10 (FastEthernet0/0), d=150.1.9.9 (Serial0/0),
g=155.1.0.3, len 124, forward, proto=47
IP: tableid=0, s=150.1.9.9 (Serial0/0), d=150.1.10.10
(FastEthernet0/0), routed via FIB
IP: s=150.1.9.9 (Serial0/0), d=150.1.10.10 (FastEthernet0/0),
g=155.1.58.8, len 124, forward, proto=47
IP: tableid=0, s=150.1.10.10 (FastEthernet0/0), d=150.1.9.9
(Serial0/0), routed via FIB
```

## ☠ Pitfall

Note to debug this traffic on R5 the **`no ip route-cache`** command is needed on the transit interfaces in order to force the traffic to be process switched. Be very careful using this command in production networks as it means that all previously CEF switched traffic must use the CPU for forwarding on a per packet basis.

## 3.11 GRE Tunneling and Recursive Routing

- Enable RIPv2 on the tunnel interface between SW3 and SW4.
- Configure distribute-list filtering on SW3 and SW4 so that a recursive routing error does not occur for routing lookups out the tunnel interface between them.

### *Configuration*

```
SW3:
router rip
 network 10.0.0.0
 distribute-list prefix STOP_RECURSIVE_ERROR out Tunnel0
!
ip prefix-list STOP_RECURSIVE_ERROR seq 5 deny 150.1.9.0/24
ip prefix-list STOP_RECURSIVE_ERROR seq 10 permit 0.0.0.0/0 le 32

SW4:
router rip
 network 10.0.0.0
 distribute-list prefix STOP_RECURSIVE_ERROR out Tunnel0
!
ip prefix-list STOP_RECURSIVE_ERROR seq 5 deny 150.1.10.0/24
ip prefix-list STOP_RECURSIVE_ERROR seq 10 permit 0.0.0.0/0 le 32
```

### *Verification*

**Note**

A recursive routing error is when a lookup for prefix X points at prefix Y, and a lookup for prefix Y points at prefix X. For tunnel interfaces this occurs when the tunnel destination is dynamically learned in the tunnel interface itself. This problem can be easily identified by the IOS log message %TUN-5-RECURDOWN: Tunnel0 temporarily disabled due to recursive routing. The step-by-step process by which this error occurs is as follows.

First, SW4 learns the route to the tunnel destination 150.1.9.9 via SW2.

```
Rack1SW4#debug ip routing
IP routing debugging is on

Rack1SW4#clear ip route *
Rack1SW4#
<output omitted>
RT: SET_LAST_RDB for 150.1.9.0/24
  NEW rdb: via 155.1.108.8

RT: add 150.1.9.0/24 via 155.1.108.8, rip metric [120/5]
<output omitted>
```

Now that there is a valid path for the tunnel control traffic, the tunnel interface comes up. With RIP enabled on the 10.0.0.0 network and no filtering, SW4 learns an alternate route for the tunnel destination.

```
Rack1SW4#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state
to up
<output omitted>
RT: del 150.1.9.0/24 via 155.1.108.8, rip metric [120/5]
RT: SET_LAST_RDB for 150.1.9.0/24
  OLD rdb: via 13.11.13.11

RT: SET_LAST_RDB for 150.1.9.0/24
  NEW rdb: via 10.34.0.9

RT: add 150.1.9.0/24 via 10.34.0.9, rip metric [120/1]
<output omitted>
```

With the reception of a new RIP route for 150.1.9.0/24 with a metric of 1, the old route with a metric of 5 is flushed out. The recursive error now occurs because 150.1.9.0/24 is reachable via 10.34.0.9, but 10.34.0.9 is the tunnel reachable via 150.1.9.9. The router detects this error, disables the tunnel, and all routes learned via the tunnel are flushed.

```
Rack1SW4#
%TUN-5-RECURDOWN: Tunnel0 temporarily disabled due to recursive routing
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state
to down
<output omitted>
RT: del 150.1.9.0/24 via 10.34.0.9, rip metric [120/1]
RT: delete subnet route to 150.1.9.0/24
<output omitted>
```

To solve this problem a distribute-list filter is put in place to never advertise the tunnel control networks 150.1.9.0/24 and 150.1.10.0/24 out the tunnel interface itself. This problem could also be solved by configuring static routes to these destinations with a lower administrative distance than the dynamically learned routes. Once complete the tunnel can be used for any destinations other than the Loopbacks of SW3 and SW4.

```
Rack1SW4#traceroute 155.1.9.9

Type escape sequence to abort.
Tracing the route to 155.1.9.9

  1 10.34.0.9 44 msec *  44 msec
```

## 3.12 Reliable Backup Interface with GRE

- Remove all previous routing configurations.
- Configure static routes on R5 for R4’s Loopback0 interface via both the Frame Relay and point-to-point connections between them.
- Configure static routes on R4 for R5’s Loopback0 interface via both the Frame Relay and point-to-point connections between them.
- The static routes on R4 and R5 via the Frame Relay circuit should have a higher administrative distance than those out the point-to-point link.
- Create a GRE tunnel between R4 and R5 using the addresses 10.0.0.Y/24 that is sourced from and destined to the Frame Relay connection between them, and enable GRE keepalives.
- Configure the backup interface feature on R5 such that if the tunnel between R4 and R5 goes down the point-to-point Serial link between them is activated.
- To verify this configuration ensure that traffic between the Loopback0 interfaces of R4 and R5 is routed out the Frame Relay link between them, but if R4’s Frame Relay interface is shut down this traffic is rerouted out the point-to-point link.

### *Configuration*

```
R4:
ip route 150.1.5.0 255.255.255.0 155.1.0.5 20
ip route 150.1.5.0 255.255.255.0 155.1.45.5 10
!
interface Tunnel0
 ip address 10.0.0.4 255.255.255.0
 tunnel source 155.1.0.4
 tunnel destination 155.1.0.5
 keepalive 1 3

R5:
ip route 150.1.4.0 255.255.255.0 155.1.0.4 20
ip route 150.1.4.0 255.255.255.0 155.1.45.4 10
!
interface Tunnel0
 ip address 10.0.0.5 255.255.255.0
 tunnel source 155.1.0.5
 tunnel destination 155.1.0.4
 keepalive 1 3
 backup interface Serial0/1
```

## *Verification*

**Note**

In the previous backup design, R4 configured the **`backup interface`** command on its point-to-point Frame Relay subinterface. Since the backup interface feature tracks the line protocol status of the link, and the point-to-point subinterface line protocol status is based on the Frame Relay PVC status, if there is a failure anywhere in the end-to-end LMI path R4 will be able to detect this and switch over to the backup link.

In this design case the backup interface command is configured on R5's main Serial0/0 interface. Since the line protocol status of this interface is based on the LMI keepalive status from the local Frame Relay switch, not the end-to-end PVC status, R5 cannot trigger the backup link when there is a failure of the circuit to R4. While the Frame Relay circuit is up we can see that R5 routes across this link to reach R4's Loopback0 network. This is due to the fact that the route with the lower administrative distance via Serial0/1 cannot be installed because this link is in standby state.

```
Rack1R5#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial0/0
Rack1R5(config-if)#backup interface Serial0/1
Rack1R5(config-if)#end
%LINK-5-CHANGED: Interface Serial0/1, changed state to standby mode
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed
state to down

Rack1R5#show ip interface brief
Interface          IP-Address      OK? Method Status                Protocol
FastEthernet0/0    155.1.58.5      YES manual up                    up
Serial0/0          155.1.0.5       YES manual up                    up
FastEthernet0/1    155.1.5.5       YES manual up                    up
Serial0/1          155.1.45.5      YES manual standby mode          down
Loopback0          150.1.5.5       YES manual up                    up

Rack1R5#show ip route 150.1.4.4
Routing entry for 150.1.4.0/24
  Known via "static", distance 20, metric 0
  Routing Descriptor Blocks:
  * 155.1.0.4
      Route metric is 0, traffic share count is 1

Rack1R5#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.0.4 28 msec *  28 msec
```

If there is a failure in the Frame Relay network that is not on the local link between R5 and the Frame Relay switch, traffic becomes black holed.

```
Rack1R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R4(config)#interface Serial0/0
Rack1R4(config-if)#shutdown
Rack1R4(config-if)#
```

Although R5 updates DLCI 504's circuit status to INACTIVE, this does not cause the line protocol of Serial0/0 to go down.

```
Rack1R5#show frame-relay pvc | include DLCI
DLCI = 501, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0
DLCI = 502, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0
DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0
DLCI = 504, DLCI USAGE = LOCAL, PVC STATUS = INACTIVE, INTERFACE = Serial0/0
DLCI = 513, DLCI USAGE = UNUSED, PVC STATUS = INACTIVE, INTERFACE = Serial0/0

Rack1R5#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        155.1.58.5      YES manual up                    up
Serial0/0              155.1.0.5       YES manual up                    up
FastEthernet0/1        155.1.5.5       YES manual up                    up
Serial0/1              155.1.45.5      YES manual standby mode          down
Loopback0              150.1.5.5       YES manual up                    up
```

Since Serial0/0 is still installed in the routing table, the route to 150.1.4.0/24 via 155.1.0.4 is still installed in the routing table, and traffic to the final destination is dropped inside of the Frame Relay network.

```
Rack1R5#show ip route 150.1.4.4
Routing entry for 150.1.4.0/24
  Known via "static", distance 10, metric 0
  Routing Descriptor Blocks:
  * 155.1.0.4
      Route metric is 0, traffic share count is 1

Rack1R5#ping 150.1.4.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.4.4, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Similar to using the IP SLA feature to add reliability to static routing, the backup interface feature can become reliable by using a GRE tunnel and tunnel keepalives. By configuring a tunnel with keepalives between R4 and R5 sourced from the Frame Relay interface, these routers will be able to detect if end-to-end IP transport is lost between these interfaces. Furthermore if the backup interface command is configured on the tunnel interface, instead of the main Serial0/0 interface, R5 will be able to switch to the backup link if there is a failure anywhere in the end-to-end transit of the Frame Relay PVC.

The following output demonstrates the same failure scenario as before, but not R5 is configured with the backup interface command on the tunnel with keepalives instead of the Serial0/0 interace.

```
Rack1R5#debug tunnel keepalive
Tunnel keepalive debugging is on
Rack1R5#debug backup
Backup events debugging is on
Tunnel0: sending keepalive, 155.1.0.4->155.1.0.5 (len=24 ttl=255),
counter=1
Tunnel0: keepalive received, 155.1.0.4->155.1.0.5 (len=24 ttl=253),
resetting counter

Rack1R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R4(config)#interface Serial0/0
Rack1R4(config-if)#shutdown
Rack1R4(config-if)#
```

With R4’s Frame Relay interface down it cannot reply to the tunnel keeplives. R5 detects this, drops the line protocol status of the tunnel, and activates the backup interface.

```
Rack1R5#
Tunnel0: sending keepalive, 155.1.0.4->155.1.0.5 (len=24 ttl=255), counter=1
Tunnel0: sending keepalive, 155.1.0.4->155.1.0.5 (len=24 ttl=255), counter=2
Tunnel0: sending keepalive, 155.1.0.4->155.1.0.5 (len=24 ttl=255), counter=3
Tunnel0: sending keepalive, 155.1.0.4->155.1.0.5 (len=24 ttl=255), counter=4
Tunnel0: sending keepalive, 155.1.0.4->155.1.0.5 (len=24 ttl=255), counter=5
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to down
BACKUP(Tunnel0): event = primary interface went down
BACKUP(Tunnel0): changed state to "waiting to backup"
BACKUP(Tunnel0): event = timer expired on primary
BACKUP(Tunnel0): secondary interface (Serial0/1) made active
BACKUP(Tunnel0): changed state to "backup mode"
%LINK-3-UPDOWN: Interface Serial0/1, changed state to up
BACKUP(Serial0/1): event = secondary interface came up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed state to up
BACKUP(Serial0/1): event = secondary interface came up
```

Once Serial0/1 is installed in the routing table, traffic from R5 to R4's Loopback is rerouted via this link.

```
Rack1R5#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.45.4 12 msec *  12 msec
```

When transport over the Frame Relay link comes back, R5's tunnel interfaces comes back up, and Serial0/1 is placed into standby state once again.

```
Rack1R4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R4(config)#interface Serial0/0
Rack1R4(config-if)#no shutdown
Rack1R4(config-if)#

Rack1R5#
Tunnel0: sending keepalive, 155.1.0.4->155.1.0.5 (len=24 ttl=255), counter=149
Tunnel0: keepalive received, 155.1.0.4->155.1.0.5 (len=24 ttl=253), resetting
counter
Tunnel0: sending keepalive, 155.1.0.4->155.1.0.5 (len=24 ttl=255), counter=1
Tunnel0: keepalive received, 155.1.0.4->155.1.0.5 (len=24 ttl=253), resetting
counter
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to up
BACKUP(Tunnel0): event = primary interface came up
BACKUP(Tunnel0): changed state to "waiting to revert"
BACKUP(Tunnel0): event = timer expired on primary
BACKUP(Tunnel0): secondary interface (Serial0/1) moved to standby
BACKUP(Tunnel0): changed state to "normal operation"
%LINK-5-CHANGED: Interface Serial0/1, changed state to standby mode
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed state to
down
BACKUP(Serial0/1): event = secondary interface went down

Rack1R5#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 155.1.0.4 32 msec *  28 msec
```

## 3.13 On-Demand Routing (ODR)

- Remove all previous routing configurations.
- Disable all physical interfaces on R1, R2, R3, R4, and R5 with the exception of their connections to the Frame Relay cloud.
- Ensure that CDP is enabled on the Frame Relay network between these devices.
- Enable ODR on R5.
- Ensure that all devices on this segment have IP reachability to each others' Loopback0 networks.

### *Configuration*

```
R5:
interface Serial0/0
 cdp enable
!
router odr
```

### *Verification*

**Note**

On-Demand Routing (ODR) uses Cisco Discovery Protocol (CDP) to advertise connected IP routes of a stub router to the hub. The hub router then advertises a default IP route back to the spokes via CDP. Assuming that the hub and spokes are already running CDP, the configuration of this feature requires only one command on the hub, **`router odr`**.

In this design CDP is disabled on the main Serial0/0 interface of R5, but it is enabled on the point-to-point subinterfaces of the spokes by default. Therefore the additional command **`cdp enable`** is required on R5.

```
Rack1R1#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Rack1R5          Ser 0/0.1          169         R S I      2611XM    Ser 0/0

Rack1R1#show ip route
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 155.1.0.5 to network 0.0.0.0

     155.1.0.0/24 is subnetted, 1 subnets
C       155.1.0.0 is directly connected, Serial0/0.1
     150.1.0.0/24 is subnetted, 1 subnets
C       150.1.1.0 is directly connected, Loopback0
o*   0.0.0.0/0 [160/1] via 155.1.0.5, 00:00:14, Serial0/0.1

Rack1R2#show cdp neighbor
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Rack1R5          Ser 0/0.1          135         R S I      2611XM    Ser 0/0

Rack1R2#show ip route
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 155.1.0.5 to network 0.0.0.0

     155.1.0.0/24 is subnetted, 1 subnets
C       155.1.0.0 is directly connected, Serial0/0.1
     150.1.0.0/24 is subnetted, 1 subnets
C       150.1.2.0 is directly connected, Loopback0
o*   0.0.0.0/0 [160/1] via 155.1.0.5, 00:00:44, Serial0/0.1

Rack1R3#show cdp neighbor
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Rack1R5          Ser 1/0.1          131         R S I      2611XM    Ser 0/0
```

```
Rack1R3#show ip route
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 155.1.0.5 to network 0.0.0.0

     155.1.0.0/24 is subnetted, 1 subnets
C       155.1.0.0 is directly connected, Serial1/0.1
     150.1.0.0/24 is subnetted, 1 subnets
C       150.1.3.0 is directly connected, Loopback0
o*   0.0.0.0/0 [160/1] via 155.1.0.5, 00:00:49, Serial1/0.1

Rack1R4#show cdp neighbor
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Rack1R5          Ser 0/0.1          127         R S I      2611XM    Ser 0/0

Rack1R4#show ip route
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 155.1.0.5 to network 0.0.0.0

     155.1.0.0/24 is subnetted, 1 subnets
C       155.1.0.0 is directly connected, Serial0/0.1
     150.1.0.0/24 is subnetted, 1 subnets
C       150.1.4.0 is directly connected, Loopback0
o*   0.0.0.0/0 [160/1] via 155.1.0.5, 00:00:52, Serial0/0.1

Rack1R5#show cdp neighbor
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Rack1R1          Ser 0/0            161         R S I      2610XM    Ser 0/0.1
Rack1R2          Ser 0/0            160         R S I      2610XM    Ser 0/0.1
Rack1R3          Ser 0/0            163         R S I      2611XM    Ser 1/0.1
Rack1R4          Ser 0/0            171         R S I      2611XM    Ser 0/0.1
```

```
Rack1R5#show ip route
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

     155.1.0.0/24 is subnetted, 1 subnets
C       155.1.0.0 is directly connected, Serial0/0
     150.1.0.0/24 is subnetted, 5 subnets
C       150.1.5.0 is directly connected, Loopback0
o       150.1.4.0 [160/1] via 155.1.0.4, 00:00:10, Serial0/0
o       150.1.3.0 [160/1] via 155.1.0.3, 00:00:17, Serial0/0
o       150.1.2.0 [160/1] via 155.1.0.2, 00:00:21, Serial0/0
o       150.1.1.0 [160/1] via 155.1.0.1, 00:00:19, Serial0/0


Rack1R1#ping 150.1.2.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 112/116/124 ms

Rack1R1#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 = 112/116/120 ms

Rack1R1#ping 150.1.4.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.4.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 113/117/124 ms

Rack1R1#ping 150.1.5.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.5.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/60 ms
```

## 3.14 OER Components Setup

- Configure R2 as the OER Master Controller and R2, R3 and R5 as the Border Routers.
- Authenticate all OER communications using the password value of “CISCO”.
- Set up external and internal OER interfaces on all routers per the topology diagram.
- Enable OER events logging and change the keepalive interval to 5 seconds.

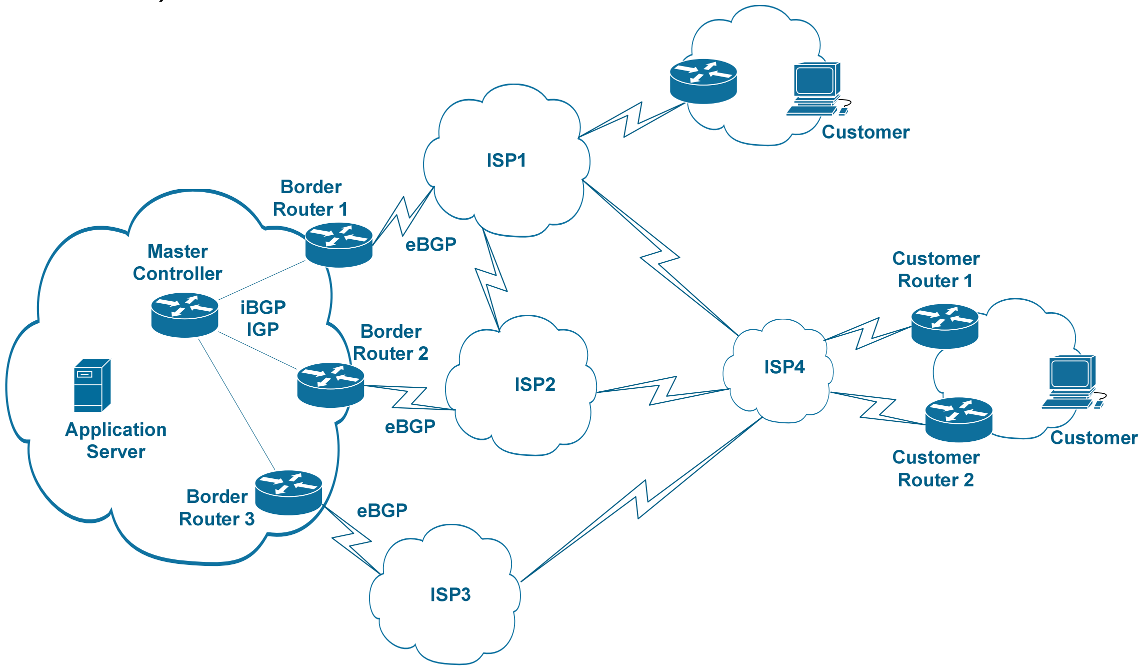
### *Configuration*

**Note**

Performance Routing (PfR), previously called Optimized Edge Routing (OER), introduces a new concept into regular IP routing. OER enhances the classic destination-based routing based the shortest path (lowest-cost metric) by taking in account application performance characteristics across different paths. For example, if there are multiple paths between nodes A and B, classic routing will select the one with the shortest “metric”. Let us assume that there is a path with high bandwidth but significant delay, e.g. a satellite link and a relatively low-bandwidth link (e.g. T1) with low delay. It would be beneficial to route VoIP calls between A and B over the T1 link and route file-transfers across the satellite link.

Regular routing-protocol may load-balance across multiple links if configured properly. Some of them even support unequal-cost load balancing, loading the links inversely proportional to the link metric. Those methods are inflexible, as they don’t account for application performance characteristics. The above goal could also be accomplished using the well-known policy-based routing feature, but PBR does not scale well in large networks, and cannot respond to dynamically changing environment. Another solution would be using MPLS traffic-engineering, but this technology is designed to be working within a single enterprise network.

OER allows for dynamic routing toward destinations outside the enterprise network based on the performance characteristics associated with network applications. Here is a short overview of a typical OER deployment. An enterprise or a service provider hosts a number of applications. For example, this could be a VoIP system, a video broadcasting server, FTP/HTTP services and so on. The enterprise networks features multiple uplinks to services providers and would like to utilize them to distribute the traffic routed *outbound* in optimal manner. Here "optimal" condition is defined as satisfying the performance policy requirements set by the administrator. OER was initially designed to control and optimize only *outbound* traffic flows, i.e. selecting an optimal exit point for every traffic flow, and the ability to control the inbound traffic (e.g. via eBGP announces) was added in IOS 12.4T.



OER deployment components include a Master Controller (MC) and Border Routers (BR). The MC is the core of OER functionality – it processes the information collected from the BRs (e.g. statistics) and uploads policy decisions (e.g. prefixes to inject into IGP) and other configurations to the BRs. The BRs connect to the edge of the network and control traffic leaving via the external links. In effect, border routers are the policy enforcement points. Additionally, it is possible to combine BR and MC in a single router.

OER process runs the following optimization loop continually to ensure the optimal exit points usage:

**`(1) OER Profile phase`**. The goal of this phase is discovering traffic classes. At this stage, OER learns traffic flows that experience performance issues automatically. Additionally, you may configure traffic classes manually.
**`(2) OER Measure phase`**. At this phase OER border routers measure traffic classes performance metrics and report them to the master controller. Measurements could be either passive, based on Netflow information and interface counters or active, where border routers simulate traffic to discover performance characteristics using IP SLA functionality.
**`(3) OER Apply Policy phase`**. You may configure the master controller with

the set of thresholds that define acceptable performance for traffic classes. The policy may contain such elements as acceptable loss, delay, throughput, link utilization and so on. Based on this policy the MC determines classes or links that are out of policy (OOP).
**`(4) OER Control phase`**. After OER has determined the classes that do not conform to the policy, it may control routing protocol decisions by injecting static or BGP routes or changing route metrics. This in effect will change the exit point selection in the network.
**`(5) OER Verify phase`**. Finally, after OER has attempted the optimization procedure, the master controller collects resulting statistics and verifies that new exit point selection brings all traffic classes and links to in-policy states.

Before we discuss all phases in depth, let’s start with the OER basic components setup. When configuring OER, you should set up both the BRs and the MC. Most of OER configurations are applied to the MC and setting a border router is relatively simple – you only need to specify the master’s address and authentication information if needed. Use the command **`oer border`** to configure a border router and specify the master router using the master command. Don’t forget to specify the local interface in the border router for communications with the master.

Use the command **`oer master`** to enter the OER configuration mode for the MC. Specify every border router (even the local one) using the command **`border <IP> [key-chain <NAME>]`**. You can use a locally configured key-chain for OER session authentication. Under the border router configuration sub-mode you may specify the router-interfaces controlled by OER using the **`interface <Name> {internal|external}`** command. At least one external interface should be defined for every border router, and at least two external interfaces must be configured for the group of BRs. The external interfaces are used for traffic monitoring and forwarding out of the local network.

Our scenario presents AS 200 having three exit points to reach AS 100. The bandwidth of the exit-point between R4 and R5 is 128Kbps, the bandwidth In this task, we configure R2 as an MC and BR simultaneously. The other BRs are R3 and R5 and the external interfaces are the links connected to other ASes. All authentications are authenticated using the key value of CISCO.

**R2:**
```
key chain OER
 key 1
  key-string CISCO
!
oer master
 keepalive 5
 logging
 !
 border 150.1.3.3 key-chain OER
  interface FastEthernet0/0 external
  interface Serial1/0.1 internal
  interface Serial1/3 internal
  interface Serial 1/2 external
 !
 border 150.1.5.5 key-chain OER
  interface Serial0/0 internal
  interface FastEthernet0/0 internal
  interface Serial0/1 external
!
 border 150.1.2.2 key-chain OER
  interface Serial0/0.1 internal
  interface FastEthernet0/0 external
  interface Serial0/1 internal
!
oer border
 local Loopback0
 master 150.1.2.2 key-chain OER
```

**R3 & R5:**
```
key chain OER
 key 1
  key-string CISCO
!
oer border
 local Loopback0
 master 150.1.2.2 key-chain OER
```

***Verification***

**Note**

Check the OER master status. Make sure all border peers are “ACTIVE” and “UP” in the output and there are no AuthFail messages. Note that there are four exits from this AS, defined in 3 border routers.

```
Rack1R2#show oer master
OER state: ENABLED and ACTIVE
  Conn Status: SUCCESS, PORT: 3949
  Number of Border routers: 3
  Number of Exits: 4
  Number of monitored prefixes: 0 (max 5000)
  Max prefixes: total 5000 learn 2500
  Prefix count: total 0, learn 0, cfg 0

Border           Status   UP/DOWN             AuthFail
150.1.2.2        ACTIVE   UP       00:23:33          0
150.1.5.5        ACTIVE   UP       00:19:50          0
150.1.3.3        ACTIVE   UP       00:19:55          0

Global Settings:
  max-range-utilization percent 20
  mode route metric bgp local-pref 5000
  mode route metric static tag 5000
  trace probe delay 1000
  logging

Default Policy Settings:
  backoff 300 3000 300
  delay relative 50
  holddown 300
  periodic 0
  mode route observe
  mode monitor both
  mode select-exit good
  loss relative 10
  unreachable relative 50
  resolve delay priority 11 variance 20
  resolve utilization priority 12 variance 20

Learn Settings:
  current state : DISABLED
  time remaining in current state : 0 seconds
  no throughput
  no delay
  no protocol
  monitor-period 5
  periodic-interval 120
  aggregation-type prefix-length 24
  prefixes 100
  expire after time 720
```

**Note**

Check the OER border router interfaces. The interfaces are configured in the master router, and the information is uploaded to the border routers.

```
Rack1R2#show oer border
OER BR 150.1.2.2 ACTIVE, MC 150.1.2.2 UP/DOWN: UP 01:18:51,
  Auth Failures: 0
  Conn Status: SUCCESS, PORT: 3949
  Exits
  Fa0/0            EXTERNAL
  Se0/0.1          INTERNAL
  Se0/1            INTERNAL
Rack1R2#

Rack1R3#show oer border
OER BR 150.1.3.3 ACTIVE, MC 150.1.2.2 UP/DOWN: UP 01:03:57,
  Auth Failures: 0
  Conn Status: SUCCESS, PORT: 3949
  Exits
  Fa0/0            EXTERNAL
  Se1/0.1          INTERNAL
  Se1/2            EXTERNAL
  Se1/3            INTERNAL

Rack1R5#show oer border
OER BR 150.1.5.5 ACTIVE, MC 150.1.2.2 UP/DOWN: UP 01:15:26,
  Auth Failures: 0
  Conn Status: SUCCESS, PORT: 3949
  Exits
  Fa0/0            INTERNAL
  Se0/0            INTERNAL
  Se0/1            EXTERNAL
Rack1R5#
```

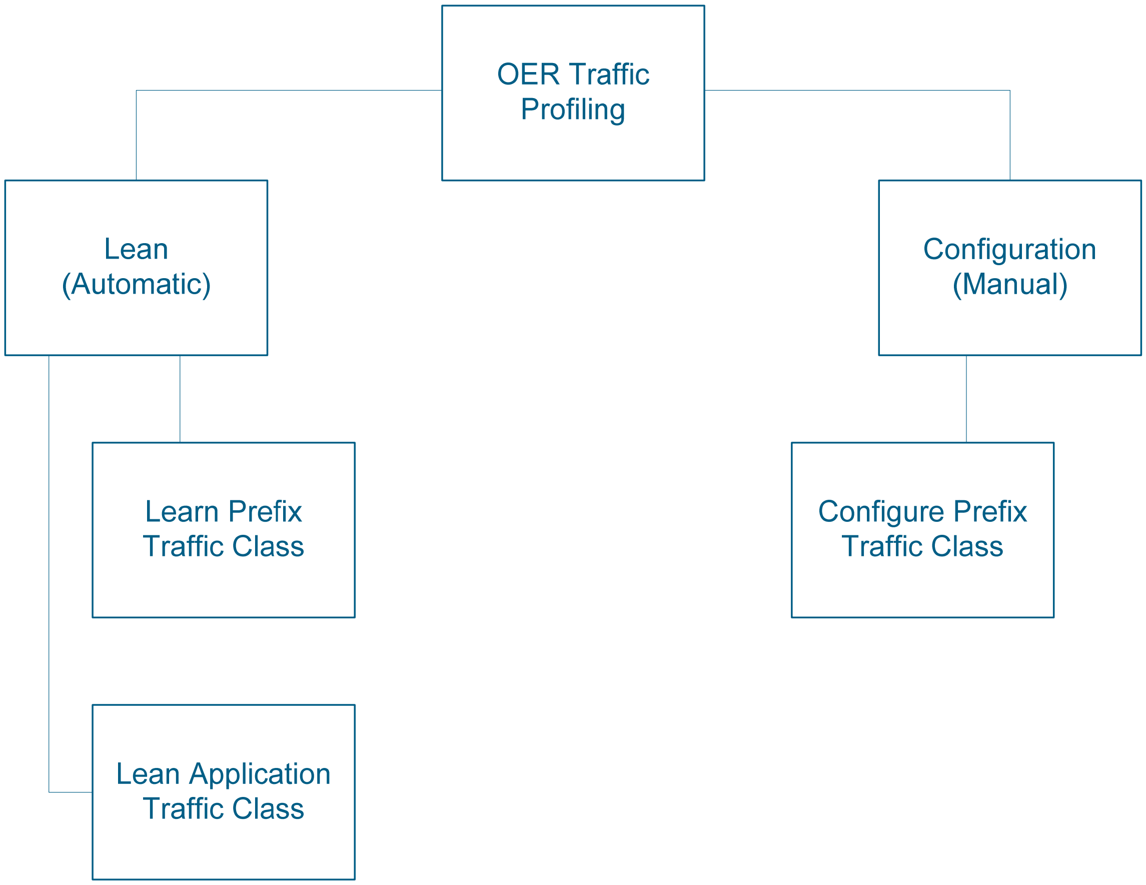
## 3.15 OER Profile Phase

- Configure the OER deployment to learn network prefixes based on maximum throughput and delay.
- The address information should be summarized based on the BGP table contents.
- Run periodic learning procedure every 5 minutes for the duration of 3 minutes and do not store more that 10 prefixes in the MTC.
- Only store the traffic flows corresponding to ICMP data exchange, WWW transactions and VoIP bearer traffic (UDP port range 16384-32767).
- Manually configure a prefix traffic class 112.0.0.0/24.

## *Configuration*

### Note

The first OER phase, known as **`OER Profile`** phase, is about discovering or defining traffic classes. Traffic class could be as simple as network prefix or be a complex object, such as network prefix and application port numbers. The routers configured for OER (BRs) could discover the classes automatically (learn) or you can configure the classes manually in the master controller. All learned or configured traffic-classes are stored in the Monitored Traffic Classes (MTC) table. Let’s look at the chart of OER traffic profiling options and describe all options in details.


OER Traffic Profiling
Lean (Automatic)
Configuration (Manual)
Learn Prefix Traffic Class
Configure Prefix Traffic Class
Lean Application Traffic Class

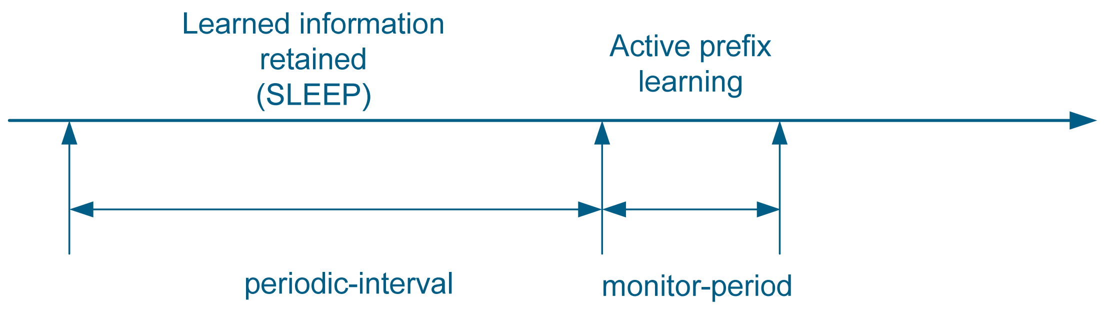
## Learning Prefix Traffic Class

Automatic prefix traffic-class learning is based on the NetFlow top-talkers feature. The Master Controller collects NetFlow information from the border routers and selects the prefixes with the highest throughput or delay using the information collected. OER classifies prefixes as either external (learned from other ASes) or internal (originated into BGP locally). Use the following command to enable automatic learning of the prefix traffic-classes.

```
oer master
  learn
    throughput
    delay
```

This will instruct all border routers to collect the NetFlow information on the *external* prefixes with the highest *outbound* throughput (number of bytes transferred) and delay (RTT time). Since the traffic flows are exchanged between hosts, the number of learned flows could exceed the MTC capacity. This is why OER applies prefix aggregation automatically. By default, external traffic sources are aggregated in /24 subnets and the resulting metrics are summarized or averaged. You may change this mode using the command **`aggregation-type {bgp | non-bgp | prefix-length <prefix-mask>}`**. When “bgp” mode is configured, the MC will look into the BGP table to find a matching prefix for a flow. This prefix is then used as the aggregate and the information is stored in the MTC. If the “non-bgp” keyword is specified, the static routes table is used for the same purpose. In the case of “prefix-length <prefix-mask>” option, the MC will automatically aggregate all flows addresses up to the prefix-length specified. This is the default mode with the prefix-length value of 24.

Prefix learning starts every time interval specified by the command **`periodic-interval <minutes>,`** with the default value of 120 minutes. The router will start monitoring traffic flows for the duration of time specified by the **`monitor-period <minutes>`** command, which defaults to 5 minutes.



There is a limit to the number of prefixes learned by the MC and stored in the MTC. By default, 100 top flows are learned, and this value could be changed using the command **`prefixes <number>`**. The prefixes stored in the MTC are expired if the MC learns no further information about these prefixes, i.e. if there are no traffic flows for these prefixes. You may specify the expiration interval using the command **`expire after {session <number> | time <minutes>}.`** When you set the expiry timeout using the **`session <number>`** command, the prefixes are expired after <number> of monitoring sessions has occurred (i.e. after **`<number>x(<periodic-interval>+<monitor-period>)).`** You may set the expiry period explicitly using the option **`time <minutes>`** defining the amount of time an inactive prefix is stored in the MTC.

## Learning Application Traffic Class

Applications are identified by the protocols and/or the port numbers they use. For example, if your network hosts an HTTP server, then traffic flows to external prefixes sourced off the local TCP port 80 could be associated with WWW application. In order to configure the master controller to learn application traffic classes, add the command **`protocol {number | tcp | udp} [port <port-number> | gt <port-number> | lt <port-number> | range <lower-number upper-number>] [dst | src]`**. This will filter the prefixes information based on the port numbers and aggregate only flows matching the specified port criteria. Use the keywords **`dst`** or **`src`** to specify whether the configured ports are source or destination. For example, to specify the internal WWW applications use the command **`protocol tcp port 80 src`**. You may configure multiple protocol commands and flows matching any of the command are stored in the MTC list.

## Configuring a Prefix Traffic Class

It is possible to specify a set of prefixes for OER monitoring and control manually, by using the **`oer-map`** command. The command syntax is similar to the route-map command - **`oer-map`** is a numbered sequence of entries, where every entry has **`match`** and **`set`** clauses. Unlike the route-map, there could be only one match clause per entry and there are no “deny” entries. If you want to include a subnet under OER control but exclude some subnets, use a prefix-list with “deny” entries. For example, if you want to monitor statistics for the prefix 114.0.0.0/8 but exclude the subnet 114.0.1.0/24 use the following construct:

```
ip prefix-list MONITOR deny 114.0.1.0/24
ip prefix-list MONITOR permit 114.0.0.0/8
!
oer-map OER 10
 match ip address prefix-list OER

oer master
 policy-rules OER
```

Notice the use of the command **`policy-rules <NAME>`** to associate the OER map with the MC configuration. The original purpose of the oer-map is to specify an OER policy – we will discuss the policy configuration in separate task.

The scenario asks to configure automatic application traffic classes learning for WWW, ICMP and VoIP RTP traffic. We filter the ICMP flows based on the protocol number “1” and the WWW traffic based on the source port of 80. VoIP bearer traffic is identified as sourced off the UDP ports in range 16384 32767. Based on this configuration, the border routers will identify traffic leaving over the external interfaces and send the statistics to the Master Controller. Notice that until there are active traffic flows in the network no information will be send to the master controller

**R2:**

```
ip prefix-list NET112 permit 112.0.0.0/24
!
oer-map OER 10
 match ip address prefix-list NET112
!
oer master
 policy-rules OER
!
 learn
  throughput
  delay
  protocol tcp port 80 src
  protocol 1
  protocol udp port range 16384 32767 src
  periodic-interval 5
  monitor-period 3
  aggregation-type bgp
```

### *Verification*

**Note**

To verify the configuration, we stat the following SLA probes to generate synthetic traffic:

1) ICMP pings between R1 and SW2
2) Jitter operation emulating G.729 codec between R6 and SW2
3) File transfer via HTTP from SW2 to R4.

We're going to see if the border routers can actually discover all traffic flows.

**R1:**
```
no ip sla monitor 1
ip sla monitor 1
 type echo protocol ipIcmpEcho 150.1.8.8 source-interface Loopback0
 timeout 100
 frequency 1
!
ip sla monitor schedule 1 start-time now life forever
```

**R4:**
```
ip http client source-interface Loopback0

Rack1R4#copy http://cisco:cisco@150.1.8.8/c3560-advipservicesk9-mz.122-
44.SE2.bin null:
```

**R6:**
```
no ip sla monitor 2
ip sla monitor 2
 type jitter dest-ipaddr 150.1.8.8 dest-port 16384 source-ipaddr
150.1.6.6 codec g729a codec-numpackets 10 codec-interval 10
 timeout 100
 frequency 1
!
ip sla monitor schedule 2 life forever start-time now
```

**SW2:**
```
ip http server
ip http path flash:
ip sla responder
```

**Note**

After you've got everything set up, you can check the prefixes learned by the border routers (it may take some time for the MC to process and learn all information). Notice that all three external prefixes are listed in the MTC, plus the manually configured prefix. List the prefixes learned based on highest throughput or delay.

```
Rack1R2#show oer master prefix learned
OER Prefix Statistics:
 Pas - Passive, Act - Active, S - Short term, L - Long term, Dly - Delay (ms),
 Los - Packet Loss (packets-per-million), Un - Unreachable (flows-per-million),
 E - Egress, I - Ingress, Bw - Bandwidth (kbps), N - Not applicable
 U - unknown, * - uncontrolled, + - control more specific, @ - active probe all

Prefix                  State     Time Curr BR         CurrI/F         Protocol
                      PasSDly  PasLDly   PasSUn   PasLUn  PasSLos  PasLLos
                      ActSDly  ActLDly   ActSUn   ActLUn      EBw      IBw
--------------------------------------------------------------------------------
-
150.1.1.0/24         INPOLICY       39 150.1.3.3       Fa0/0           BGP
                            U        U        0        0     3173     2631
                           12       12        0        0       31        4
150.1.4.0/24         OOPOLICY      412 150.1.3.3       Fa0/0           BGP
                            N        N        N        N        N        N
                          260      270        0        0        N        N
150.1.6.0/24         INPOLICY      @16 150.1.3.3       Fa0/0           BGP
                            U      173        0        0     3812     3163
                            9        9        0        0       19        2
```

**Note**

We will discuss the numerous fields in the output above in the task dedicated to OSPF measurement phase. So far, make sure the prefixes are just learned by the controller. Now, for automatically learned prefixes you may want to check the criterion used for prefix learning: either delay or throughput (based on Netflow measurements). Notice that some prefixes may experience both excessive delay and generate maximum throughput.

```
Rack1R2#show oer master prefix learned delay
OER Prefix Statistics:
 Pas - Passive, Act - Active, S - Short term, L - Long term, Dly - Delay (ms),
 Los - Packet Loss (packets-per-million), Un - Unreachable (flows-per-million),
 E - Egress, I - Ingress, Bw - Bandwidth (kbps), N - Not applicable
 U - unknown, * - uncontrolled, + - control more specific, @ - active probe all

Prefix                  State     Time Curr BR         CurrI/F         Protocol
                      PasSDly  PasLDly   PasSUn   PasLUn  PasSLos  PasLLos
                      ActSDly  ActLDly   ActSUn   ActLUn      EBw      IBw
--------------------------------------------------------------------------------
-
150.1.6.0/24         INPOLICY       @6 150.1.3.3       Fa0/0           BGP
                            U      173        0        0     2038     3055
                           12        9        0        0       10        4
150.1.1.0/24         INPOLICY       @9 150.1.3.3       Fa0/0           BGP
                            U        U        0        0     3831     2599
                           16       16        0        0        8        1

Rack1R2#show oer master prefix learned throughput
OER Prefix Statistics:
 Pas - Passive, Act - Active, S - Short term, L - Long term, Dly - Delay (ms),
 Los - Packet Loss (packets-per-million), Un - Unreachable (flows-per-million),
 E - Egress, I - Ingress, Bw - Bandwidth (kbps), N - Not applicable
 U - unknown, * - uncontrolled, + - control more specific, @ - active probe all

Prefix                  State     Time Curr BR         CurrI/F         Protocol
                      PasSDly  PasLDly   PasSUn   PasLUn  PasSLos  PasLLos
                      ActSDly  ActLDly   ActSUn   ActLUn      EBw      IBw
--------------------------------------------------------------------------------
-
150.1.1.0/24         INPOLICY       @6 150.1.3.3       Fa0/0           BGP
                            U        U        0        0     3831     2599
                           16       16        0        0        8        1
150.1.6.0/24         INPOLICY       @2 150.1.3.3       Fa0/0           BGP
                            U      173        0        0     2038     3055
                           12        9        0        0       10        4
150.1.4.0/24         OOPOLICY      287 150.1.3.3       Fa0/0           BGP
                            N        N        N        N        N        N
                          234      266        0        0        N        N
```

**Note**

You may use the following command the display the Netflow Cache entries used to learn the traffic-classes.

```
Rack1R3#show oer border passive cache prefix

OER Passive Prefix Cache, State: enabled, 278544 bytes
  5 active, 4091 inactive, 1256 added
  29471 ager polls, 0 flow alloc failures
  Active flows timeout in 1 minutes
  Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 21640 bytes
  10 active, 1014 inactive, 2512 added, 1256 added to flow
  0 alloc failures, 0 force free
  1 chunk, 2 chunks added

Prefix              NextHop         Src If         Dst If          Flows
                      Pkts   B/Pk  Active   sDly   #Dly  PktLos  #UnRch
-------------------------------------------------------------------------
150.1.6.0/24        0.0.0.0         Se1/2          Se1/0.1           28
                       151     58    13.0     40      1       0       0
150.1.1.0/24        0.0.0.0         Se1/2          Se1/0.1            1
                        30     64     0.0      0      0       0       0
150.1.1.0/24        0.0.0.0         Se1/2          Se1/3              1
                        37     45     1.8     32      1       0       0
150.1.1.0/24        155.1.37.7      Se1/3          Fa0/0              2
                        36    104     5.0      0      0       0       0
150.1.6.0/24        155.1.37.7      Se1/3          Fa0/0            128

Prefix              NextHop         Src If         Dst If          Flows
                      Pkts   B/Pk  Active   sDly   #Dly  PktLos  #UnRch
-------------------------------------------------------------------------
                       652     58    68.0      0      0       0       0
```

## 3.16 OER Measure Phase

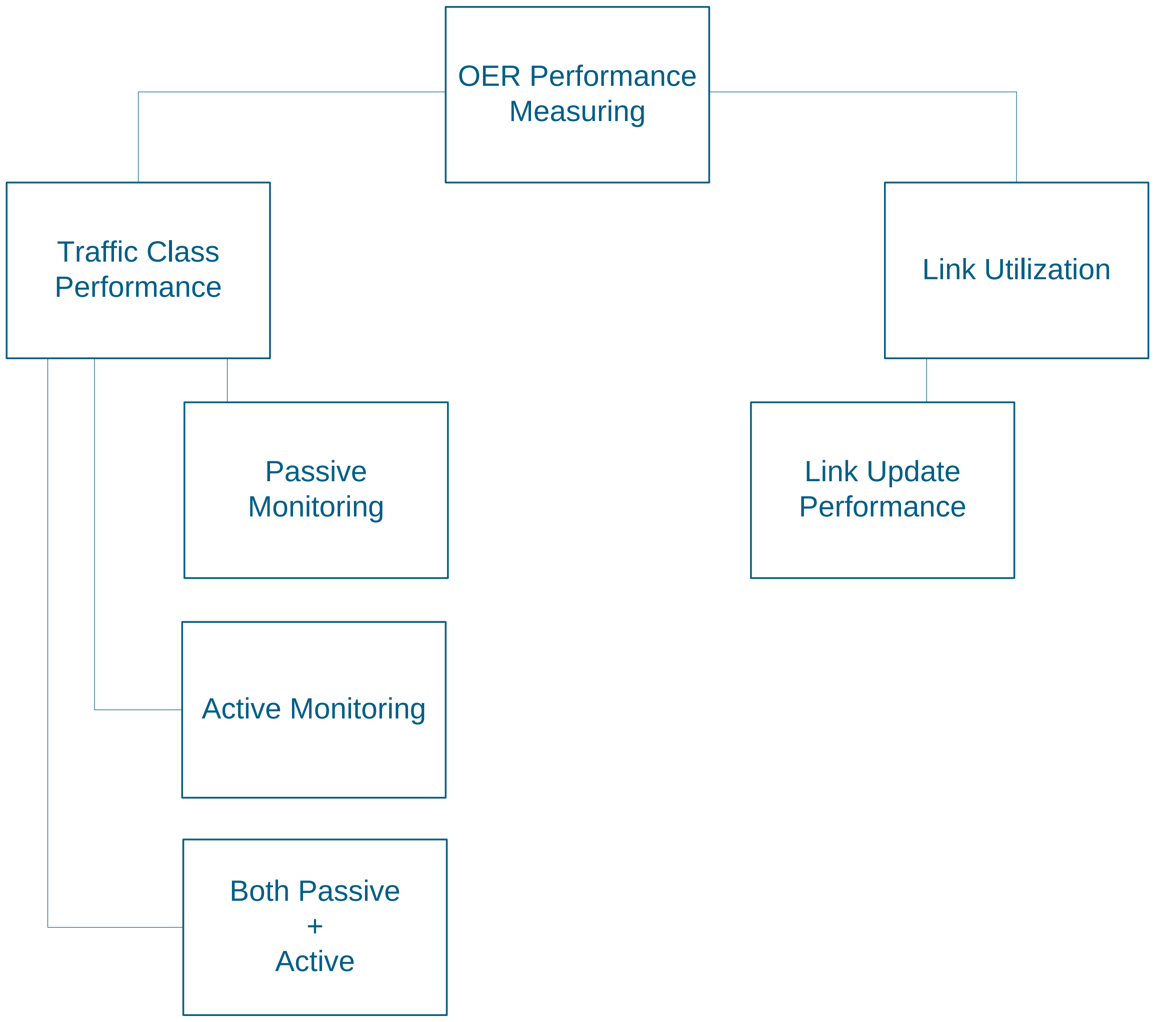
- Use the combined monitoring mode and configured manual TCP connect probes for the IP addresses of R1, R4 and R6 Loopback0 interface.
- Configure the master controller to use active-monitoring only for the prefix 150.X.4.0/24.
- Configure the border routers so that any interface utilization changes are detected as fast as possible.

***Configuration***

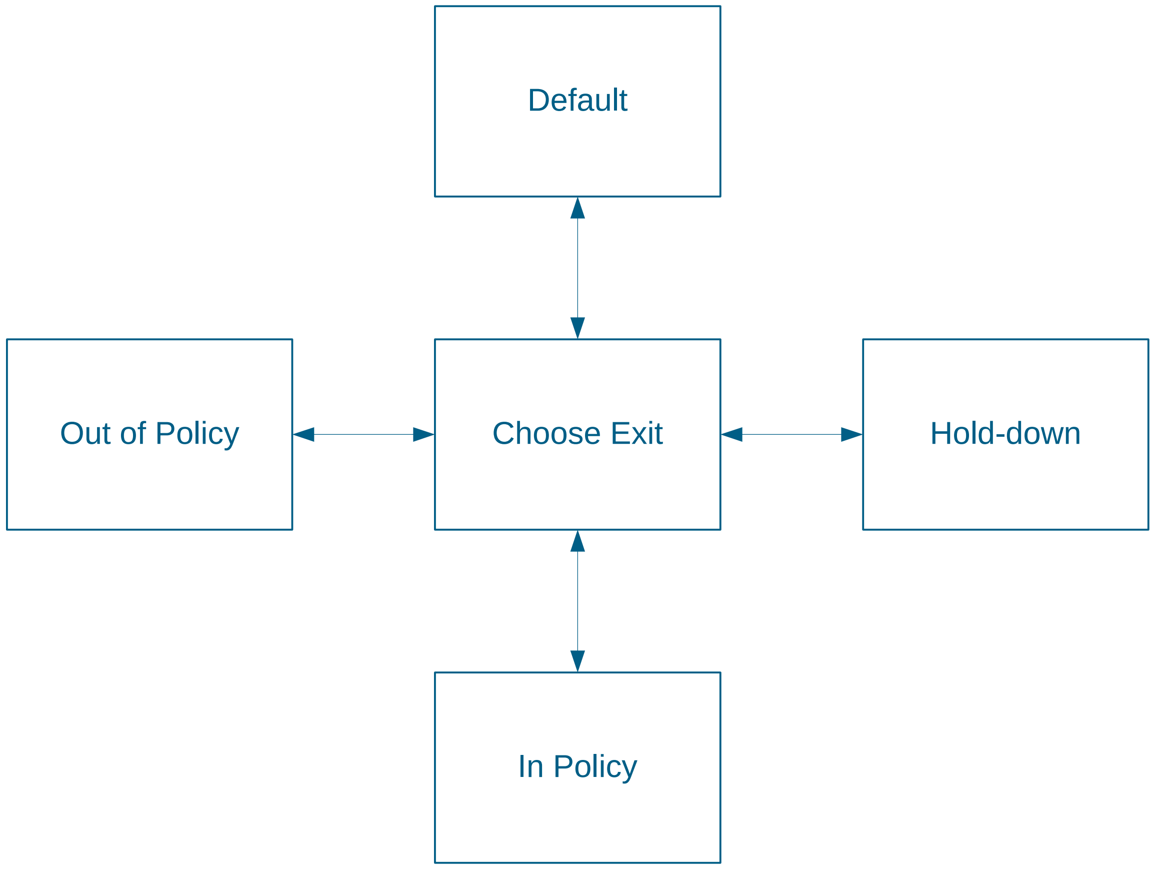
**Note**

**`OER Measure Phase`**

This phase is the second step in OER performance optimization loop. This step starts after the traffic-classes have been learned automatically of configured by the administrator. During this phase, border routers passively monitor traffic flows or actively probe selected targets. In addition to monitoring traffic-classes, border routers also monitor external links performance, based on link load and error rates. The collected information is reported back to the MC, where this information is associated with the traffic-classes stored in the MTC. Here is an outline of the measurement process:



OER measures the performance of both traffic classes and external links, but before monitoring a traffic class or a link, OER checks its current state. OER transitions traffic-class or link through a series of states per the following diagram:



The current state selected for a traffic-class or link is called the policy decision point (PDP). In some states, OER does not initiate monitoring. Here is the list of the states:

**`Default`** – means the traffic class is not under OER control. Traffic classes are placed in the default state when they are initially added to the MTC. A traffic class will transition into and out of the default state depending on the measurement results and policy configuration.

**`Choose Exit`** – This is a transient state, where the OER compares performance characteristics with the traffic policy and select an exit point (BR/external interface) for this class. OER will try to keep a traffic class flowing through its current exit but performance measurements, timers, and policy configurations can cause the MC to start looking for another exit point. The traffic class remains in this state until it is moved to the new exit.

**`Holddown`** - A traffic class is placed in this state when the MC requests a border router to monitor the traffic class using active probes. Measurements are collected for the selected traffic class until the hold-down timer expires.

**`In-Policy`** - After performance metrics are compared against the policy settings and an exit point selection is made, the traffic class enters an in-policy state. When a traffic class is in this state, the traffic is forwarded through an exit that satisfies the default or user-defined policy settings. The MC keeps monitoring the traffic class, but no action is taken until the periodic timer expires, or an out-of-policy even is generated. In this case, the traffic class transitions back to the "Choose Exit" state.

**`Out-of-Policy (OOP)`** - A traffic class is enters this state when there are no exit points conforming the performance policy settings for this class. While the traffic class is in this state, the backoff timer controls exiting from this state. Each time the traffic class enters this state, the amount of time the traffic class spends in this state increases exponentially. The backoff timer is reset for a traffic class when the traffic class enters the "in-policy" state. If all exit links are out-of-policy, the master controller may select the best available exit.

As mentioned, OER uses three methods of performance measurement:

## Passive monitoring

BRs measure the performance metrics of a traffic class while the corresponding traffic is flowing through the device. Passive monitoring uses NetFlow functionality and enabled by default. Passive monitoring can also be enabled explicitly using the **`mode monitor passive`** command for a traffic class. NetFlow support is enabled by default on the border routers when passive monitoring is enabled. Passive monitoring uses only existing traffic - additional traffic is not generated. Border routers collect and report passive monitoring statistics to the master controller approximately once per minute. OER uses passive monitoring to measure the following metrics for all traffic classes:

**`Delay`** - OER measures the average delay of TCP flows for a given prefix. Delay is measured as the round-trip response time (RTT) between the transmission of a TCP segment and receipt of the TCP ACK packet. This metric is measured only for the TCP flows.

**`Packet loss`** - OER measures packet loss by tracking TCP sequence numbers for each TCP flow. OER estimates packet loss by storing the highest TCP sequence number and comparing it the subsequent packets. If a subsequent packet is received with a lower sequence number, meaning retransmission, OER increments the packet loss counter. Packet loss is measured in packets per million and only measured for TCP flows.

**`Reachability`** - OER measures reachability by tracking TCP SYN messages that have been sent repeatedly without receiving a TCP ACK packet. Again, this metric applies only to TCP flows.

**`Throughput`** - OER measures throughput by measuring the total number of bytes and packets for each traffic class for a given interval of time. This metric is monitored for all types of flows, not just TCP-based.

Passive monitoring is enabled by default for all automatically learned prefixes. You may enforce the default passive mode monitoring using the **`oer master`** configuration mode command **`mode passive`** or the **`oer-map`** command **`set mode passive`**. The last command allows you to selectively enforce a particular monitoring mode for a manually configured prefix.

## Active monitoring

In this mode, the master controller instructs the border routers to create a stream of synthetic traffic replicating a traffic class behavior as closely as possible and measuring the performance metrics of the resulting flows. The results of the performance metrics of the synthetic traffic are applied to the corresponding traffic class in the MTC list. Active monitoring uses the IP Service Level Agreements (IP SLAs) functionality to implement active traffic probes. Active monitoring could measure metrics such as delay or reachability for any types of flows, not just TCP flows as with passive monitoring.

In active monitoring mode, the OER activates the probes on all the border routers to find the best performance path for the specific traffic class. The active probes for that traffic class are not activated again unless the traffic class goes OOP (e.g. due to a link utilization OOP event). In Cisco IOS Release 12.4(4)T and earlier releases, the frequency of an active probing used by OER is fixed to 60 seconds.

OER uses active monitoring to measure the following metrics for all the traffic classes (not just TCP-based traffic classes!):

**`Delay`** - OER measures the average delay of TCP, UDP, and ICMP flows for a given prefix. Delay is the measurement of the round-trip response time (RTT) between the transmission of a TCP synchronization message and receipt of the TCP acknowledgement.

**`Reachability`** - OER measures reachability by tracking TCP synchronization messages that have been sent repeatedly without receiving a TCP acknowledgement.

**`Jitter`** - Jitter means inter-packet delay variance. OER measures jitter by sending multiple packets to a target address and a specified target port number, and measuring the delay interval between packets arriving at the destination.

**`MOS`** - Mean Opinion Score (MOS) is a standards-based method of measuring voice quality. MOS scores range between 1 representing the worst voice quality, and 5 representing the best voice quality.

Use the **`oer master`** mode command **`mode active`** to switch the monitoring mode to active by default. You can change the mode for a manually configured prefix using an oer-map and the clause **`set mode monitoring active.`** This allows for per traffic-class granularity when setting the monitoring modes.

## Types of Active Probes

As mentioned, active probes are implemented using IP SLA functionality in IOS routers. The following types of active probes can be configured for use by OER.

**`ICMP Echo`** - A ping is sent to the target address. OER uses ICMP Echo probes, by default, when an active probe is automatically created (described later).

**`Jitter`** - A jitter probe is sent to the target address. A target port number must be specified. A remote responder must be enabled on the target device, regardless of the configured port number. Jitter probe support was introduced in Cisco IOS Release 12.4(6)T.

**`TCP Connection`** - A TCP connection probe is sent to the target address and port number. A remote responder must be listening on the port configured in the target settings, or IP SLA responder must be enabled in the remote IOS router.

**`UDP Echo`** - A UDP echo probe is sent to the target address and port pair. A remote responder must be enabled on the target IOS device.

## Creation of Active Probes

To create an active probe for a traffic class, a probe type has to be *discovered*, and a probe target *assigned* to the traffic class. To discover a probe type, OER uses one of the following methods:

**`Learned probe`** - Active probes are automatically generated when a traffic class is learned using the NetFlow learn mechanism (highest delay or throughput flows). Five targets are learned for each traffic class and, by default, the active probe type is set as an ICMP echo probe.

**`Configured probe`** - Active probes can be configured on the master controller by specifying the probe type, target address and port if needed. These probes are assigned to traffic classes based on the longest match (see below). Additionally, manually configured traffic classes can use any of the IP SLA active probes bound to them using an oer-map. Use the **`oer master`** mode subcommand **`active-probe <Parameters>`** to configure a manual probe (see below on the probe to traffic-class association process).

When creating an active probe through an external interface for a specified target, the target should be *reachable* through the external interface. To test the reachability of the specified target, OER performs a route lookup in the BGP and static routing tables for the specified target and external interface. If the target is reachable, OER uses one of the following methods to assign a probe target for a traffic class:

**`Longest match`** - By default, OER assigns a probe target to the traffic class with the longest matching prefix in the MTC list. This the default probe assignment. This procedure works for both learned and configured probes, i.e. the configured probe will not be activated until any of the BRs report a matching prefix.

**`Forced assignment`** - An IP SLA probe can be configured using an OER map (**`set active-probe`** oer-map clause) and the results of the measurements are assigned to the specific traffic class associated with this OER map entry. This specific assignment of active probe is called a forced target probe assignment. This feature is not available in IOS 12.4 code.

Here is a sample configuration of active probes in the master controller:

```
oer master
 active-probe echo 10.1.1.1
 active-probe tcp-conn 10.1.1.2 target-port 80
 active-probe udp-echo 150.1.6.6 target-port 123
```

Keep in mind that in order for the TCP/UDP probes to work, you need to make sure the remote host is configured to respond on this port. For example, you may need to configure an IOS router using the command **`ip sla monitor responder type tcpConnect ipaddress <IP> port <number>`** to start the SLA responder on this port.

Since active probes are assigned to the border routers the probe traffic is transmitted over an external interface using the interface IP address to source the packets. Notice that all border routers will run the probe, even through the external interfaces that may not be an optimal exit point for the traffic class. The reason behind this is collecting performance metrics on all alternate exit paths for this traffic-class.

## Combined Monitoring

Includes both active and passive monitoring - uses both active and passive monitoring in order to generate a more complete picture of traffic flows within the network. This is the monitoring mode implemented by default - all traffic classes are passively monitored using integrated NetFlow functionality and out-of-policy traffic classes are actively monitored using IP SLA functionality. This allows for selection of the new best exit point without collecting any information passively, as a give traffic-class uses just one exit point.

One example scenario is when you want to learn traffic classes and then monitor them passively, but you also want to determine the alternate path performance metrics in order to control the traffic classes. The alternate metrics in the absence of the actual traffic can be measured using the active probes. OER automates this process by learning traffic classes at five targets (learned probles) and probing through all the alternate paths using active probes when detecting an OOP event.

## OER Link Utilization Measurement

### Link Utilization Threshold

After an external interface is configured for a border router, OER automatically monitors the utilization of the external link, which typically links to an ISP. Every 20 seconds, by default, the border router reports the link utilization to the master controller. If the exit link utilization is above the default threshold of 75 percent, the exit link is in an OOP state and OER starts the monitoring process to find an alternative link for the traffic classes using this link. The OER polls the interface load counters that are calculated by the border router's IOS. In order to increase the reaction time to loading changes, you may want to set the **`load-interval`** value to the minimum of 30 seconds on all external interfaces.

Although configuring link threshold is a part of the Apply Policy phase, we are going to discuss it here. The link utilization threshold can be manually configured either as an absolute value in kilobits per second (kbps) or as a percentage using the external interface configuration command **`max-xmit-utilization {absolute kbps | percentage value}`**. Look at the configuration sample that sets the transmit bandwidth usage for an interface:

```
oer master
  border 1.1.1.1
    interface Serial 0/0 external
       max-xmit-utilization percentage value 50
```

### Link Utilization Range

You may also configure OER to calculate the range of utilization over all the links, i.e. to detect if the exit links are not being used equally. By default, OER automatically monitors the utilization of external links on a border router every 20 seconds, and the border router reports the utilization to the master controller. If the utilization range between all the exit links exceeds 20%, the master controller tries to equalize the traffic load by moving some traffic classes to another exit link. The maximum utilization range is configured as a percentage using the **`oer master`** mode command **`max-range-utilization percent <maximum>`**.
.
The scenario changes external links load-interval and adds three active probes manually, using the TCP connection operation. Since the default port 23 used for the TCP operation, there is no need to configure IP SLA responder in the target systems. For the prefix 150.X.4.0/24 the monitoring mode is changed to active.

**R2:**
```
ip prefix-list R4_LO0 permit 150.1.4.0/24
!
oer-map OER 20
 match ip address prefix-list R4_LO0
 set mode monitor active
!
oer master
 active-probe tcp-conn 150.1.1.1 target-port 23
 active-probe tcp-conn 150.1.4.4 target-port 23
 active-probe tcp-conn 150.1.6.6 target-port 23
!
interface FastEthernet 0/0
 load-interval 30
```

**R3:**
```
interface FastEthernet 0/0
 load-interval 30
!
interface Serial 1/2
 load-interval 30
```

**R5:**
```
interface Serial 0/1
 load-interval 30
```

## *Verification*

**Note**

Let's configure R1, R4 and R6 for downloading of IOS images via HTTP from R2, SW2 and SW2 again respectively. We are going to use the Loopback0 interfaces on R1, R4 and R6 to source the connections.

**R1:**
```
ip http client source-interface Loopback0

copy http://cisco:cisco@150.1.2.2/c2600-adventerprisek9-mz.124-17.bin
nullll:
```

**R4:**
```
ip http client source-interface Loopback0

Rack1R4#copy http://cisco:cisco@150.1.8.8/c3560-advipservicesk9-mz.122-
44.SE2.bin null:
```

**R6:**
```
ip http client source-interface Loopback0

copy http://cisco:cisco@150.1.8.8/c3560-advipservicesk9-mz.122-
44.SE2.bin null:
```

**R2 & SW2:**
```
ip http server
ip http path flash:
```

**Note**

Check once again the prefixes learned and configured in the master controller. Now for the description of the fields in the output:

PasSDly – passively measured delay, short-range (5 minutes interval)
ActSDly – actively measured delay (proving), short-range
PasLDly – passively measure delay, long-range (60 minutes interval)
PasSUn/PasLUn – passively measure short and long range unreachable metric
ActSUn/ActLUn – the same unreachable, just measured actively
PasSLos/PasSLos – short and long range loss, measured passively.
EBw/IBw – egress and ingress bandwidth usage for this class in kbps.
State – any of the states from the traffic state diagram.
Time – the amount of time spent in the state.
Curr BR – current border router selected for this class.
Curr I/F – current exit interface selected for this class.
Protocol – protocol used to influence routing for this traffic class.

```
Rack1R2#show oer master prefix
OER Prefix Statistics:
 Pas - Passive, Act - Active, S - Short term, L - Long term, Dly - Delay (ms),
 Los - Packet Loss (packets-per-million), Un - Unreachable (flows-per-million),
 E - Egress, I - Ingress, Bw - Bandwidth (kbps), N - Not applicable
 U - unknown, * - uncontrolled, + - control more specific, @ - active probe all

Prefix                     State     Time Curr BR         CurrI/F         Protocol
                         PasSDly  PasLDly   PasSUn   PasLUn  PasSLos  PasLLos
                         ActSDly  ActLDly   ActSUn   ActLUn      EBw      IBw
--------------------------------------------------------------------------------
112.0.0.0/24           DEFAULT*       57 U                U
150.1.1.0/24           INPOLICY      @45 150.1.3.3        Fa0/0           BGP
                              16       16        0        0     1669     2277
                              22       16        0        0       24        3
150.1.4.0/24           HOLDDOWN      310 150.1.5.5        Se0/1           BGP
                               N        N        N        N        N        N
                               U        U        0        0        N        N
150.1.6.0/24           INPOLICY      @33 150.1.3.3        Fa0/0           BGP
                             316      316        0        0     1978     2534
                              10       10        0        0       15        5
```

**Note**

Now check the probes existing in the OER domain. Notice that the “echo” probes were auto-generated for the traffic classes learned by the MC. The TCP connect probes that we configured manually have been assigned to the respective classes based on the longest-match criteria.

Finally, the last part of the output shows the currently active probes and the border routers running them.

```
Rack1R2#show oer master active-probes
        OER Master Controller active-probes
Border   = Border Router running this Probe
State    = Un/Assigned to a Prefix
Prefix   = Probe is assigned to this Prefix
Type     = Probe Type
Target   = Target Address
TPort    = Target Port
How      = Was the probe Learned or Configured
N - Not applicable

The following Probes exist:

State      Prefix             Type     Target          TPort How
Assigned   150.1.4.0/24       tcp-conn 150.1.4.4          23 Cfgd
Assigned   150.1.4.0/24       echo     150.1.4.4           N Lrnd
Assigned   112.0.0.0/24       echo     112.0.0.1           N Lrnd
Assigned   150.1.6.0/24       tcp-conn 150.1.6.6          23 Cfgd
Assigned   150.1.6.0/24       echo     150.1.6.6           N Lrnd
Assigned   150.1.1.0/24       tcp-conn 150.1.1.1          23 Cfgd
Assigned   150.1.1.0/24       echo     150.1.1.1           N Lrnd

The following Probes are running:

Border          State    Prefix             Type     Target          TPort
150.1.5.5       ACTIVE   150.1.1.0/24       echo     150.1.1.1           N
150.1.3.3       ACTIVE   150.1.1.0/24       echo     150.1.1.1           N
150.1.3.3       ACTIVE   150.1.1.0/24       echo     150.1.1.1           N
150.1.3.3       ACTIVE   150.1.4.0/24       tcp-conn 150.1.4.4          23
150.1.5.5       ACTIVE   150.1.4.0/24       tcp-conn 150.1.4.4          23
150.1.3.3       ACTIVE   150.1.4.0/24       tcp-conn 150.1.4.4          23
```

## 3.17 OER Apply Policy Phase

- Configure the network's performance policy to look for the best exit for the traffic classes.
- OER should generate an OOP event once any link utilization exceeds 80% or when the range between links utilization is above 10%.
- Any acceptable exit point should have relative loss of no higher that 3% and delay no higher than 200ms.
- Best exit should be selected based on the minimum external utilization, and then based on the lowest delay for the prefix-class.
- Reverse the above policy logic for the prefix 150.X.6.0/24 and change the acceptable delay to 100ms.
- All exit points with the performance metrics within 15% of the best exit should be considered equivalent.
- Ensure for periodic re-optimization occurring every minimal time interval.
- After an OOP event, the traffic class should stick to the new exit for no less than 10 minutes.

### *Configuration*

**Note**

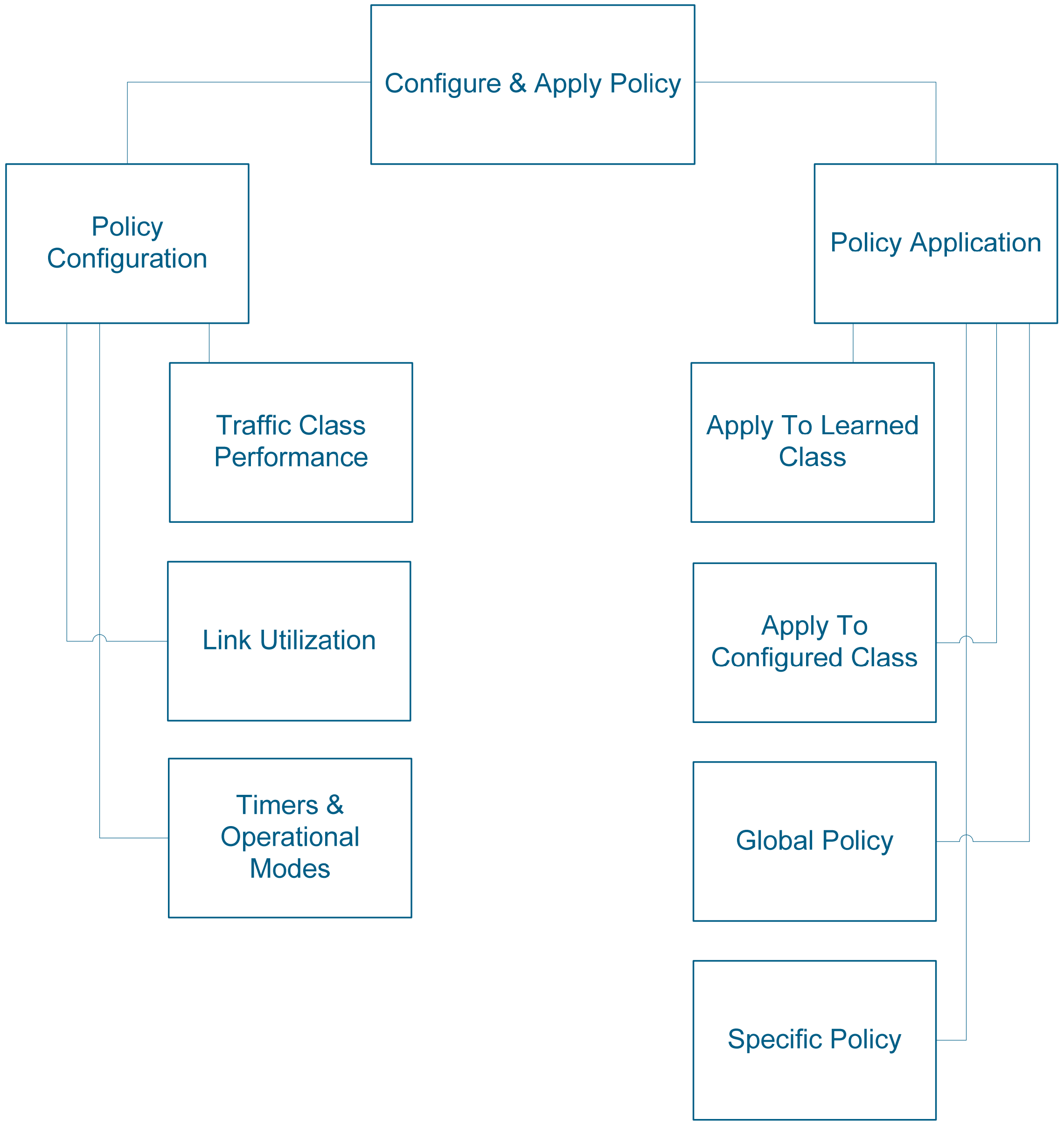
The OER apply policy phase is the third step in the OER performance optimization loop. After the profile phase has identified traffic classes and the measure phase has determined their performance characteristics OER compares the metrics collected against the default or configured thresholds to see if the traffic classes meet the performance requirements. If the performance metrics do not conform to the policy, a decision is made by OER to move the traffic class into another state: e.g. look for a better exit point or start additional probing.

The OER performance policy is a set of rules defined via an oer-map command. Every rules (oer-map entry) has the following attributes:

a) A scope, i.e. the traffic class
b) An action, e.g. insert a static route on the border router to change the exit point.
c) A condition that triggers the rule: e.g. the acceptable delay thresholds or MOS score range.

For example, a policy may require that MOS for a voice traffic class (the scope) should not fall below the value of 4 (the triggering event). In the case if this happens, OER should look for another exist point for the particular traffic class (the action). The actual policy action may be not be executed until OER is configured to control the traffic in the OER control phase (next step). By default, OER runs in an observe mode during the profile, measure, and apply policy phases and no changes are being made to the network.

During this phase you can configure and apply policies. Different types of OER policies can be configured, as outlined on the diagram below:



After an OER policy is configured, the policy can be applied to learned traffic classes or configured traffic classes. OER policies can be applied globally - to all the traffic classes - or to just a specific set of traffic classes.

## Traffic Class Performance

OER traffic class performance policy is a set of rules that specify acceptable performance characteristics for traffic classes. The classes can be network addresses (prefixes) or applications, defined via protocols, port numbers and DSCP values. The list of the performance characteristics controlled by OER includes the following:

**`Reachability`**. OER extends the concept of reachability by automatically verifying that the destination can be reached through the particular path using the active probes or passive packets exchange monitoring. Reachability is specified as the relative percentage or the absolute number of unreachable hosts, based on flows per million (fpm), that OER will permit for a single traffic class. If the measure value exceeds the configured threshold, the traffic class is declared out of policy (OOP) and OER starts looking for an alternate exit. Use the command **`[set] unreachable {relative <average%> | threshold <absolute-maximum-fpm`**>} under the oer master configuration mode (global) or oer-map configuration mode (class-specific).

The relative unreachable host percentage is based on a comparison of *short-term* and *long-term* measurements. The short-term measurement reflects the percentage of hosts that are unreachable within the last measured 5-minute period. The long-term measurement reflects the percentage of unreachable hosts within the last 60-minute period. The following formula is used to calculate this value:

```
Relative-Average% = ((short-term percentage - long-term
percentage) / long-term percentage) * 100% (1)
```

This formula allows to adaptively track the number of unreachable hosts for the traffic class – the “quick” value is normalized based on the “slow” base value. This allows for comparing the short-term traffic class behavior with its own behavior on a longer scale and avoiding “whiplash” false detections. For example, if there was 10 unreachable hosts last 60 minutes and 20 last 5 minutes, the result will be 100% jump in the unreachable hosts.

**Delay**. Defined as the time passed between when the packet was sent from the source and when it arrived at the destination. Delay is measured in passive mode for the TCP flows and in active mode for any traffic class. You may set the relative and absolute policy thresholds using the oer master mode or oer-map command **[set] delay {relative <average>%|threshold <absolute-msecs>}**. These settings reflect the global and the class-specific policy settings respectively. The formula to calculate the relative deviation in percent is the same (1).

**Packet Loss**. Measured in packet per million and monitored for TCP traffic classes in passive mode. When using the active monitored mode, this metric could be measure for any traffic class. The commands for the oer global mode and the oer-map is **[set] delay {relative <average>%|threshold <absolute-packets-per-million>}**.

**Jitter** and **Mean Opinion Score (MOS)** are two advanced metrics introduced in the recent IOS releases. For the sake of simplicity we wont cover those here, and rather provide a separate lab on their usage. Keep in mind, that in Cisco IOS Release 12.4(4)T and prior releases, only reachability, delay, and loss performance characteristics could be used.

## OER Link Policies

OER link policies apply to the external links of the OER border router and define the desired performance characteristics of the links, concerning with the performance of the link as a whole. The following performance characteristics are controlled by link policies (collected every 20 seconds during the measurement phase).

**Link Utilization** or Traffic Load. OER border routers report the current outbound load of their external interfaces taking the rate value from the interface counters. The interface load is calculated by the border router's IOS using the averaging interval specified by the interface command **load-interval <seconds>** with the minimum value of 30 seconds and the default of 300 seconds. It is recommended to set the load-interval to 30 seconds to allow the master controller to respond to interface load changes rapidly.

If the link utilization exceeds the threshold, the exit link is declared to be in an OOP state and OER starts the monitoring process to find an alternative link for the traffic classes affected by this event.

The link utilization threshold can be manually configured either as an absolute value in kilobits per second (kbps) or as a percentage using the border router external interface command **`max-xmit-utilization percentage <percents%>`** or absolute values using the command **`max-xmit-utilization absolute <Kbps>`**. In the former case, the percentage is taken off the total interface bandwidth configured via the interface **`bandwidth`** command and the default threshold is 75%. You may see the sample relative utilization configuration in the previous scenario, as part of measurement configuration.

**`Range`**. The range policy maintains all links within a certain utilization range, relative to each other in order to ensure that the traffic load is distributed evenly between the exit points. The OER range functionality allows you configuring OER to maintain the traffic utilization on a set of links within a certain percentage range of each other. If the difference between the links becomes too great, OER will attempt to bring the link back to an in-policy state by distributing traffic classes among the available links. The default difference between the links load is 20% but his value could be changed globally in **`oer master`** mode using the command **`max-range-utilization percent <percent>`**.

**`Cost`**. Cost-based optimization allows you configuring policies based on the monetary cost of each exit link in your network. To implement OER cost-based optimization the OER master controller sends traffic over exit links that provide the most cost-effective bandwidth utilization, while still maintaining the desired performance characteristics.

Cost-based optimization supports two billing models: fixed-rate billing or tier-based billing.

**`Fixed-rate`** billing reflects the flat payment rate for network access regardless of bandwidth usage. If fixed-rate billing only is configured on the exit links, all exits are considered equal with regard to cost-optimization and other parameters such as delay, loss, and link utilization are used to determine if the traffic class or exit link is in-policy. If the exit links are configured with tiered and fixed policies, then exit links with fixed policies have the highest priority with regard to cost optimization. Only if all the fixed exit links are at maximum utilization, the tiered exit links will be used.

**`Tier-based`** billing reflects the ISP billing at a tiered rate based on the percentage of exit link utilization. Each cost tier is configured separately as a percentage of bandwidth utilization with an associated monetary cost. We are not going to discuss this cost-optimization model here, due to its complexity.

Based on the facts above, we won't be concerned too much with cost optimization policies for now, as they only make sense when using the tier-based billing mode.

## OER Policy Operational Modes and Timers

So far, we've discussed the policies that apply to traffic classes or links and relate to performance metrics. Now we will look into the timers used by OER and different OER operational modes.

### OER Timers

Three types of timers can be configured as OER policy operational parameters:

**`Backoff Timer`**. After a traffic-class enters the out-of-policy state, the master controller will wait some time before attempting to find another exit for this class. The special backoff timer defines the time-interval that the MC waits and this interval gets incremented every time the master controller cannot find another optimal exit. The purpose of such behavior is allowing the master controller to have more time to make a decision after every unsuccessful attempt. The command to define the backoff timer is entered in the oer master mode: **`backoff <min-inteval-seconds> <max-interval-seconds> <step-seconds>`**. When a class enters the OOP state, the master controller waits for the initial minimum interval and attempts to find another exit. If the exit cannot be found, the minimum wait timer is incremented by the <step> value and counting down starts again. The minimum interval could be incremented until it reaches the maximum interval. The default values for all three parameters are 300, 3000 and 300 respectively.

**`Holddown Timer`** sets the minimum period of time that a new exit point selected after an OOP *must* be used before an alternate exit can be selected. This procedure prevents the traffic class entry from flapping because of rapid state changes. A traffic class entry will remain in a holddown state for the time period defined via the master-controller configuration mode command **`holddown <300-65535 sec`**> which defaults to 300 seconds. When the holddown timer expires, OER will select the best exit based on performance and policy configuration. The only event that allows for early breaking of the hold-down timer is when the prefix becomes unreachable via the currently selected exit point.

Together, the backoff and the holddown timer determine the important event dampening procedure used by OER. Dampening is about finding the compromise between quick reactions to network events (e.g. OOP even) versus the unwanted network churn that can occur when reacting to multiple events happening within a short time period (e.g. changing routes often in response to link flapping). Dampening allows the software to react quickly to initial events, while giving the network time to “relax” and adjust to the changes before starting any further actions.

**`Periodic Timer`**. By default, the master controller will look for an alternate path when an OOP event occurs. However, you may configure a complementary periodic optimization process, launched every time-interval specified using the master-controller configuration command **`periodic <180-5000 sec>`**.The periodic process finds a better path for a traffic class entry, even if the traffic class entry is in-policy on the current exit. When the periodic timer expires, the master controller evaluates current exit links for the traffic class entry and, if a better exit exists based on the current measurements, the traffic class entry is moved to a new in-policy exit link.

For all three timers, newly configured setting will immediately replace the existing setting if the value of the new setting is less than the time remaining. If the value is greater than the time remaining, the new setting will be applied when the existing timer expires or is reset.

## OER Operational Mode Options

Three types OER policy operational options could be set globally under the master controller configuration mode or per specific traffic class using the oer-map command **`set mode`**.

**`Mode Monitor`**. Set the parameters for OER measurements. We encountered this mode before, when discussing the OER Measurement Phase. Could be changed using the command **`mode monitor {active|passive|both}`** under the master-controller configuration mode or oer-map entry using the **`set mode`** command.

**`Mode Route`**. This option specifies one of three OER route control policy settings. The **`mode route control`** command enables OER to control routes associated with the traffic classes automatically, **`mode route metric`** specifies OER route protocol-related settings, and **`mode route observe`** offers route control advice, but does not take any action. We will discuss this mode in the scenario dedicated to the route control.

**`Mode Select-Exit`**. This option defines the exit selection policy. While the traffic class is in policy using the currently assigned exit, OER does not search for an alternate exit link. This is the default mode configured using the command **`[set] mode select-exit`** good under the master-controller configuration or oer-map entry. There are other scenarios, where OER selects the best performance path, not just any path that satisfies the policy requirement for the traffic class. This type of configuration can be activated by using the **`[set] mode select-exit best`** command. In this type of situation, OER measures alternate path performance metrics while the traffic class entry is in-policy on the current path and changes the current exit point if a better performance path is found. After the first selection of the best path, however, OER does not initiate another search unless the periodic timer is configured (see the timers above) or an OOP event occurs. When the periodic timer expires, the master controller evaluates current exit links for the traffic class entry and, if a better exit exists based on the current measurements and priorities, the traffic class entry is moved to a new in-policy exit link. Notice that the periodic timer is disabled by default, so you may want to enable it when selecting the best exit path at any time.

There is another difference between the select-exit mode. If OER does not find an in-policy exit when in select-exit good mode, OER transitions the traffic class entry to an *uncontrolled* state. If the same event occurs when the select-exit best mode is active, OER selects the best of the OOP exit links for the traffic class entry.

## OER Policy Application

OER policies can be applied to learned or configured traffic classes. OER policies can be applied on a global basis when the policy settings mentioned above are configured directly under OER master controller configuration mode . All traffic classes inherit global policies. If you want to apply a policy to a subset of the traffic classes, then a specific policy can be configured. Specific policies inherit global policy settings unless the same settings get overwritten by the specific policy.

To apply specific policies to learned or configured traffic classes, OER map configuration is used (this has been mentioned a few times before already). Use the match clause of the oer-map to define the traffic class of interest. It is possible to match any statically configured prefix or match OER learned prefix by means of the command **`match oer learn {delay|throughput}`**. The OER map applies the configuration of the **`set`** clause after a successful match occurs. An OER set clause can be used to set policy parameters such as the backoff timer, packet delay, holddown timer, packet loss, operational mode settings, periodic timer, resolve settings, unreachable hosts etc. The OER map could be applied using the command **`policy-list`** in the master controller configuration mode.

## Resolving Multiple OER Policies Conflicts

When configuring multiple policy criteria for a single traffic class entry, or a set of traffic classes, it is possible to have multiple conflicting policies selecting different exit points. For example, one exit point may provide best delay while the other has lowest link utilization. To resolve the potential conflict of the policy selection, OER uses its resolve function based on the priority set for a performance metric.

Each metric, such as delay, utilization range, loss and so on is assigned a unique value, and the metric with the *lowest* numerical value is used to select the optimal exit point. By default, OER assigns the highest priority to delay settings, followed by utilization settings. To configure the policy conflict resolution, use the **`resolve`** command in OER master controller configuration mode, or the **`set resolve`** command in OER map configuration mode. However, prior to a configuration example we need to discuss the metric variance settings for the resolution engine.

## Using Variance for Policy Conflict Resolution

When configuring OER resolve settings, you can also set an allowable variance for the defined policy values. Variance configures the acceptable “range” between the metrics measured for different exits that allows treating the different exits as equivalent with respect to a particular policy (e.g. a delay policy). Variance is specified in percents and defines the acceptable deviation from the *best* metric among all network exits. For example, if the delay on the best exit link for a traffic class is 80 ms and a 10% variance is configured, then any other exit links with a delay between 80 and 88 ms for the same traffic class entry are considered *equivalent* to the best exit link. This allows for OER to select the set of the exit point that are equal with respect to the highest-priority characteristic and then select the best one among the equal based on the second-priority policy.

To illustrate this concept, look at the following example. The network has three exit links with the following characteristics (ppm = packet per million).

Exit A - Delay is 100 ms, jitter is 3ms, loss threshold absolute 3000 ppm
Exit B - Delay is 220 ms, jitter is 5ms, loss threshold absolute 1200 ppm
Exit C - Delay is 240 ms, jitter is 1ms, loss threshold absolute 1000 ppm

The following OER policy conflict resolution is configured and applied to the traffic class entry:

```
oer-map TEST 10
 match ip address prefix-list NETWORK
 set resolve loss priority 1 variance 30
 set resolve delay priority 2 variance 20
 set resolve jitter priority 3 variance 10
```

Assume all exit links are within the policy (all parameters satisfy the thresholds) and OER needs to find the best exit (select-exit best mode) for the configured prefix. First, OER looks for the highest-priority policy which loss. The best Exit is C and the acceptable variance is 30%, so any link with the absolute loss of up to 1300 is equivalent to link C. This appears to be Exit B. Now OER attempts to select the best exit between B and C based on the next priority policy, which is delay. The best exit link now is Exit B (delay 220 ms) and the acceptable variance is 20%, which puts Exit C on par with B, as 220*20%=44ms and 240 ms < 220+44 ms. Now, the final resolution should be done using the lowest priority policy which is jitter. It appears that Exit C has the lowest jitter value and based on this the conflict resolution procedure selects Exit C as the optimal.

Now let’s briefly discuss the scenario itself. The task has a lot of policy requirements which should be analyzed and the following information extracted:

1) Global policy: metric thresholds and resolution priorities. From the task wording, we may conclude that delay and loss metric should be bounded. We set the relative value for the loss to 25% and absolute threshold for the delay to 200ms. The conflict resolution is configured so that utilization is preferred over delay and the variance is set to 15%.

Additionally, the link utilization and range thresholds are configured globally in the oer master controller configuration and the border router settings. We reflect the maximum outbound utilization of 60% and the range of 10%.

2) Class-specific policy: there is one prefix that requires special handling, 150.X.6.0/24. We create an OER map entry matching this prefix and tune the absolute delay threshold to 100ms. Additionally we reverse the exit selection priority so that delay is preferred over utilization with the variance for both metrics being the same 15%.

3) Operational parameters: timers and modes. We set the periodic timer to 180 seconds, as this is the minimal value. At the same time, the exit selection mode is changed to “best”, so that the best path is selected on every period. Based on the requirement to stick traffic classes to their exits for 6 minutes after an OOP we configure the hold-down timer to 360 seconds.

**R2:**

```
ip prefix-list R6_LO0 permit 150.1.6.0/24
!
oer-map OER 30
 match ip address prefix-list R6_LO0
 set delay threshold 100
 set resolve delay priority 1 variance 15
 set resolve utilization priority 2 variance 15
!
oer master
 policy-rules OER
 max-range-utilization percent 10
!
 border 150.1.3.3 key-chain OER
  interface Serial1/2 external
   max-xmit-utilization percentage 80
  interface FastEthernet0/0 external
   max-xmit-utilization percentage 80
 !
 border 150.1.5.5 key-chain OER
  interface Serial0/1 external
   max-xmit-utilization percentage 80
!
 border 150.1.2.2 key-chain OER
  interface FastEthernet0/0 external
   max-xmit-utilization percentage 80
 !
 delay threshold 200
 loss relative 25
 holddown 360
 mode select-exit best
 periodic 180
 resolve utilization priority 1 variance 15
 resolve delay priority 2 variance 15
```

## *Verification*

**Note**

At this point, policy has been applied but the route mode is set to control by default. Thus, OER take no action upon policy violation. You may check the current global (default) policy settings and the specific configuration using the following show command.

```
Rack1R2#show oer master policy
Default Policy Settings:
  backoff 90 500 90
  delay threshold 200
  holddown 360
  periodic 180
  mode route observe
  mode monitor both
  mode select-exit best
  loss relative 25
  unreachable relative 50
  resolve utilization priority 1 variance 15
  resolve delay priority 2 variance 15
oer-map OER 10
  match ip prefix-lists: NET112
  backoff 90 500 90
  delay threshold 200
  holddown 360
  periodic 180
  mode route observe
  mode monitor both
  mode select-exit best
  loss relative 25
  unreachable relative 50
  resolve utilization priority 1 variance 15
  resolve delay priority 2 variance 15
oer-map OER 20
  match ip prefix-lists: R4_LO0
  backoff 90 500 90
  delay threshold 200
  holddown 360
  periodic 180
  mode route observe
 *mode monitor active
  mode select-exit best
  loss relative 25
  unreachable relative 50
  resolve utilization priority 1 variance 15
  resolve delay priority 2 variance 15
```

```
oer-map OER 30
  match ip prefix-lists: R6_LO0
  backoff 90 500 90
 *delay threshold 100
  holddown 360
  periodic 180
  mode route observe
  mode monitor both
  mode select-exit best
  loss relative 25
  unreachable relative 50
 *resolve delay priority 1 variance 15
 *resolve utilization priority 2 variance 15

* Overrides Default Policy Setting
```

**Note**

Now check the border interface utilization thresholds. Notice that the Max BW is set based on the 80% of the interface bandwidth.

```
Rack1R2#show oer master border detail
Border           Status   UP/DOWN              AuthFail
150.1.2.2        ACTIVE   UP       01:37:54          0
 Se0/1           INTERNAL UP
 Fa0/0           EXTERNAL UP
 Se0/0.1         INTERNAL UP

 External         Capacity      Max BW   BW Used Tx Load Status         Exit Id
 Interface         (kbps)       (kbps)    (kbps)    (%)
 ---------        --------      ------   ------- ------- ------         -----
 Fa0/0              100000       80000         0       0 UP                 4
-------------------------------------------------------------------------------
Border           Status   UP/DOWN              AuthFail
150.1.5.5        ACTIVE   UP       01:37:30          0
 Se0/1           EXTERNAL UP
 Fa0/0           INTERNAL UP
 Se0/0           INTERNAL UP

 External         Capacity      Max BW   BW Used Tx Load Status         Exit Id
 Interface         (kbps)       (kbps)    (kbps)    (%)
 ---------        --------      ------   ------- ------- ------         -----
 Se0/1                 128         102         0       0 UP                 3
-------------------------------------------------------------------------------
Border           Status   UP/DOWN              AuthFail
150.1.3.3        ACTIVE   UP       01:37:37          0
 Se1/2           EXTERNAL UP
 Se1/3           INTERNAL UP
 Se1/0.1         INTERNAL UP
 Fa0/0           EXTERNAL UP

 External         Capacity      Max BW   BW Used Tx Load Status         Exit Id
 Interface         (kbps)       (kbps)    (kbps)    (%)
 ---------        --------      ------   ------- ------- ------         -----
 Se1/2                 128         102         0       0 UP                 2
 Fa0/0                 256         204       127      49 UP                 1
```

## 3.18 OER Control & Verify Phase

- Enable OER to control routes the network domain.
- Use the tag value of 1000 for injected static routes and BGP local preference value of 6000.
- Ensure that any injected static route will be redistributes into the IGP used by AS 200.

***Configuration***

**Note**

The next step in OER performance optimization loop is the Control Phase, where OER actively attempts to influence traffic routing in the network. After the traffic classes have been profiled, their performance metrics determined and compared to the policy thresholds OER needs to perform control actions to optimize the network exit points usage.

OER, by default, operates in an observation mode - the master controller monitors traffic classes and exit links based on the OER policies, reports the status of the network, but does not implement any changes.  To activate the control mode, use the OER master controller **`mode route control`** command. In this mode, the master controller coordinates information from the borders routers in the same way as observe mode, but commands are sent back to the border routers to alter routing. You may selectively control a specific traffic class by using an OER map command **`set mode route control`**. This allows for combined observe and control modes applied to different traffic classes.

OER initiates route changes when one of the following occurs:

**`A traffic class goes OOP`**, e.g. exceeds any metric threshold and OER starts looking for an alternate exit.

**`An exit link goes OOP`**, e.g. by exceeding the link utilization or losing link connectivity.

**`The periodic timer expires`** and the select exit mode is configured as select best mode. This instructs OER to looks for a better path if any exists and possible re-route the traffic class.

During the OER control phase, the master controller continues to monitor all in-policy traffic classes to ensure that they remain in-policy. Changes are only implemented for OOP traffic classes in order to bring them in-policy. The backoff and holddown timers described in the previous task can be used to provide dampening in an OER-managed network. If aggressive delay or loss policies are defined, and the exit links are seriously over-subscribed (i.e. peak usage might exceed the maximum link bandwidth), it is possible that OER will find it impossible to bring a traffic class in-policy. In this case, the master controller will either choose the link that most closely conforms to the performance policy, even though the traffic class still remains OOP, or it will remove the prefix from OER control. OER is designed to allow you to make the best use of available bandwidth, but it does not solve the problem of over-subscribed bandwidth.

## OER Traffic Class Control Techniques

We are going to review just the basic route control techniques here, available in IOS 12.4. As for the entrance link selecting features available in recent IOS revisions and other advanced features – we will cover this in separate tasks.

There are a number of techniques that can be used to influence the routing decision, by changing IGP prefix metrics, altering BGP attributes, or introducing policy-based routing using a route map. If the traffic associated with the traffic class is defined only by a prefix then a traditional routing control mechanism such as injecting a BGP route or a static route can be used. This type of control is network-wide, as new prefixes are redistributed either into IGP or BGP and propagate to all network devices, influencing path selection at all routers. Before injecting a route, OER will verify that a route exists in the BGP or static table that includes the prefix and points to the exit link. This route may be a default route or a supernet, matching the prefix being injected. This condition ensures that the border router actually has the path to reach the injected prefix. Notice that OER will automatically select the type of injected route (BGP or static) based on the matchin route present in the border router's routing table.

If the traffic associated with the traffic class is defined by a prefix and other matching criteria for the packet such as port number, then traditional routing cannot be employed to control the application traffic. In this situation, the control becomes device specific and not network specific. This device specific control is implemented by OER using policy-based routing (PBR) functionality and available since IOS 12.4(2)T. Notice that using PBR to change the routing decisions requires the border routers to be one-hop away, either physically or via a tunnel mesh. We are not going to discuss PBR-based route control in this task, but mentioned it for the sake of completeness.

## Static Route Injection

An OER master controller can enforce the use of a particular border router as the preferred exit link for a traffic class by injecting temporary static routes. These routes are temporary and intentionally not saved to the permanent configuration. There are different methods that the master controller can use to inject static routes in the border routers.

(a) Existing static routes can be overwritten with new static routes having a better routing metric.
(b) The master controller may inject a more specific route, if the border routers use default routing. The longest-match routing will attract the traffic to the proper exit point.
(c) The master controller can also use the technique known a split prefixes. This method adds a more specific prefix in the existence of a less-specific one. For example, if the border route has a static route to 10.0.0.0/8 then adding the prefix 10.0.1.0/16 to the another border router will “split” traffic for the specific subnet. OER can use split prefixes to redirect subsets of an existing prefix to a more optimal exit link, and can use split prefixes for both internal BGP (iBGP) and static routes.

When redistributing the static injected routes it is helpful to know what routes where injected by OER. To make this easy, you may instruct the OER process to associate a tag with the static routes. The injected static route will be tagged with the value specified using the command **`mode route metric static tag <value>`**. You may reference this tag when performing route redistribution using the route-map command **`match tag`**.

## BGP Control Techniques

OER uses two BGP techniques to enforce the best exit path; injecting a BGP route, or modifying the BGP local preference attribute. BGP route injection is similar to split-prefix static route injection, but it happens in the BGP table. All OER injected routes remain local to an autonomous system, and these injected routes are never shared with external BGP peers. To ensure this behavior, when OER injects a BGP route, it will set the no-export community on it. This requires enabling the send-community configuration for all iBGP peers to make sure that the prefix wont leak out of the AS. In some cases, it may also be necessary to redistribute this information into the IGP, to make sure all routers (even non-BGP speakers) learned the new information.

OER may also use BGP local preference to control traffic classes. The master controller instructs one of the border routers to apply the local preference attribute to a prefix or set of prefixes associated with a traffic class. This information is then propagated to all iBGP peers affecting best-path selection within the AS. Once the iBGP convergence is complete, the border router with the highest local-preference for the prefix will become the exit link from the network. For example, assume that all border peers learn the prefix 10.0.0.0/24 which is under OER control. In order to select a particular exit point, OER will adjust the local preference on the respective border peer and this will attract the traffic to the proper exit.

In order to specify the local-preference value used by OER injected/adjusted prefixes use the OER master controller configuration command **`mode route metric bgp local-pref <value>.`** You may change the local-preference value for a particular traffic-class using the OER map command **`set mode route metric bgp local-pref <value>`**. Notice that changing the or static metric values requires you to reset the active OER master controller connections using the command **`clear oer master border *`**.

## OER Verify Phase

The last phase of the OER performance loop is to verify that the actions taken during the OER control phase control actually change the flow of traffic and that the performance of the traffic class or link has moved to an in-policy state. OER uses NetFlow for automatic route control verification. The master controller expects a Netflow update for the traffic class from the new exit link interface and ignores Netflow updates from the previous exit. If a Netflow update does not appear after two minutes, the master controller moves the traffic class into the default state, releasing it from OER control.

In this scenario, we activate route control by default, but disable it for the prefix 150.X.6.0/24. Additionally, we configured the OER BGP local-preference value of 6000 and static route tag of 1000. We do not configure BGP community sending on iBGP peering sessions, as no new prefixes are expected to be injected into BGP. OER is only supposed to modify the Local Preference to affect the exit points.

**R2:**
```
oer-map 10
 set mode route observe
!
oer master
 mode route control
 mode route metric static tag 1000
 mode route metric bgp local-pref 6000
!
router ospf 1
 redistribute static route-map STATIC_TO_OSPF
!
route-map STATIC_TO_OSPF
 match tag 1000
```

**R3:**
```
router ospf 1
 redistribute static route-map STATIC_TO_OSPF
!
route-map STATIC_TO_OSPF
 match tag 1000
```

**R5:**
```
router ospf 1
 redistribute static route-map STATIC_TO_OSPF
!
route-map STATIC_TO_OSPF
 match tag 1000
```

## *Verification*

**Note**

To verify route control, we configure R1, R4 and R6 for downloading of IOS images via HTTP from R2, SW2 and SW2 again respectively. We are going to use the Loopback0 interfaces on R1, R4 and R6 to source the connections. This will simulate three traffic flows from AS 200 to AS 100. Additionally, we configure R5 to that traffic from SW2 is limited to approximately 112Kbps and thus may “fit” the “slow” link between R4 and R5. The transfers from R2 are limited to 128Kbps automatically by the speed of the link connecting R2 and R3.

**R1:**
```
ip http client source-interface Loopback0

copy http://cisco:cisco@150.1.2.2/c2600-adventerprisek9-mz.124-17.bin
nulll:
```

**R4:**
```
ip http client source-interface Loopback0

Rack1R4#copy http://cisco:cisco@150.1.8.8/c3560-advipservicesk9-mz.122-
44.SE2.bin null:
```

**R5:**
```
ip access-list extended HTTP
 permit tcp any eq www any
!
class-map match-all HTTP
 match access-group name HTTP
!
policy-map RATE_LIMIT
 class HTTP
    police 112000
!
interface FastEthernet 0/0
 service-policy input RATE_LIMIT
```

**R6:**
```
ip http client source-interface Loopback0

copy http://cisco:cisco@150.1.8.8/c3560-advipservicesk9-mz.122-
44.SE2.bin null:
```

**R2 & SW2:**
```
ip http server
ip http path flash:
```

**Note**

Check the exit link selection on the master controller router. Notice that two prefixes are in policy state, while prefix for R4 is being hold down.

```
Rack1R2#show oer master prefix
OER Prefix Statistics:
 Pas - Passive, Act - Active, S - Short term, L - Long term, Dly - Delay (ms),
 Los - Packet Loss (packets-per-million), Un - Unreachable (flows-per-million),
 E - Egress, I - Ingress, Bw - Bandwidth (kbps), N - Not applicable
 U - unknown, * - uncontrolled, + - control more specific, @ - active probe all

Prefix                  State     Time Curr BR         CurrI/F         Protocol
                      PasSDly  PasLDly   PasSUn   PasLUn  PasSLos  PasLLos
                      ActSDly  ActLDly   ActSUn   ActLUn      EBw      IBw
-----------------------------------------------------------------------
112.0.0.0/24         DEFAULT*        3 U               U
150.1.1.0/24         HOLDDOWN      238 150.1.5.5       Se0/1             BGP
                            U        U        0        0      628      628
                            U        U        0        0       69       10
150.1.4.0/24         INPOLICY      @48 150.1.3.3       Se1/2             BGP
                            N        N        N        N        N        N
                           16       16        0        0        N        N
150.1.6.0/24         INPOLICY      @62 150.1.3.3       Fa0/0             BGP
                            U        U        0        0        0     1509
                           16       11        0        0        3        4
```

**Note**

Now check the OER controlled routes in the BGP tables of all border routers. Notice that R3 and R5 shows the prefixes as “Controlled” meaning these prefixes have been modified by OER. The local preference value is set to 6000 per our configuration. The next-hop values in the BGP table show the exit currently used for the particular traffic class.

The current exit distribution is different from the one that would be used by default. You can see that all three exits from the AS are used concurrently; Even the path across AS 300 (via SW1) that wouldn’t be preferred under normal conditions.

```
Rack1R2#show oer border routes bgp
OER BR 150.1.2.2 ACTIVE, MC 150.1.2.2 UP/DOWN: UP 00:36:49,
  Auth Failures: 0
  Conn Status: SUCCESS, PORT: 3949
BGP table version is 64, local router ID is 150.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
OER Flags: C - Controlled, X - Excluded, E - Exact, N - Non-exact, I -
Injected

   Network          Next Hop        OER    LocPrf Weight Path
*>i150.1.1.0/24     155.1.45.4      XN      6000      0 100 i
*>i150.1.4.0/24     155.1.13.1      XN      6000      0 100 i
*>i150.1.6.0/24     155.1.37.7      XN      6000      0 300 100 i

Rack1R3#show oer border routes bg
OER BR 150.1.3.3 ACTIVE, MC 150.1.2.2 UP/DOWN: UP 00:36:56,
  Auth Failures: 0
  Conn Status: SUCCESS, PORT: 3949
BGP table version is 65, local router ID is 150.1.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
OER Flags: C - Controlled, X - Excluded, E - Exact, N - Non-exact, I -
Injected

   Network          Next Hop        OER    LocPrf Weight Path
*>i150.1.1.0/24     155.1.45.4      XN      6000      0 100 i
*> 150.1.4.0/24     155.1.13.1      CE                0 100 i
*> 150.1.6.0/24     155.1.37.7      CE                0 300 100 i
Rack1R3#

Rack1R5#show oer border routes bgp
OER BR 150.1.5.5 ACTIVE, MC 150.1.2.2 UP/DOWN: UP 00:37:02,
  Auth Failures: 0
  Conn Status: SUCCESS, PORT: 3949
BGP table version is 60, local router ID is 150.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
OER Flags: C - Controlled, X - Excluded, E - Exact, N - Non-exact, I -
Injected

   Network          Next Hop        OER    LocPrf Weight Path
*> 150.1.1.0/24     155.1.45.4      CE                0 100 i
*>i150.1.4.0/24     155.1.13.1      XN      6000      0 100 i
*>i150.1.6.0/24     155.1.37.7      XN      6000      0 300 100 i
```