Balaji Bandi (BB)

My CCIE Journey My Wish

Just started my CCIE discovering Labs – spend more time – but its worth going all the Blue print Labs and learn things.

Learning Lessons, more Labs more hands give you more expertise

Lab Training Videos

Understanding Current Topologies (up-to-date)

Solid Troubleshooting Skills

Right resource and Lab server

Time Management is key for the CCIE Lab

Dedicated to: Family and Friends (Especially GOD) who give me courage to be here as part of my Technical Journey

1.0 Layer 2 Technologies 20%

1.1 LAN switching technologies

1.1.a Implement and troubleshoot switch administration
1.1.a [i] Managing MAC address table
1.1.a [ii] errdisable recovery
1.1.a [iii] L2 MTU

1.1.b Implement and troubleshoot layer 2 protocols
1.1.b [i] CDP, LLDP
1.1.b [ii] UDLD

1.1.c Implement and troubleshoot VLAN
1.1.c [i] access ports
1.1.c [ii] VLAN database
1.1.c [iii] normal, extended VLAN, voice VLAN

1.1.d Implement and troubleshoot trunking
1.1.d [i] VTPv1, VTPv2, VTPv3, VTP pruning
1.1.d [ii] dot1Q
1.1.d [iii] Native VLAN
1.1.d [iv] Manual pruning

1.1.e Implement and troubleshoot etherchannel
1.1.e [i] LACP, PAgP, manual
1.1.e [ii] layer 2, layer 3
1.1.e [iii] load-balancing
1.1.e [iv] etherchannel misconfiguration guard

1.1.f Implement and troubleshoot spanning-tree
1.1.f [i] PVST+/RPVST+/MST
1.1.f [ii] switch priority, port priority, path cost, STP timers
1.1.f [iii] port fast, BPDUguard, BPDUfilter
1.1.f [iv] loopguard, rootguard

1.1.g Implement and troubleshoot other LAN switching technologies
1.1.g [i] SPAN, RSPAN, ERSPAN

1.2 Layer 2 Multicast

1.2.a Implement and troubleshoot IGMP

1.2.a [I] IGMPv1, IGMPv2, IGMPv3
1.2.a [ii] IGMP snooping
1.2.a [iii] IGMP querier
1.2.a [iv] IGMP filter
1.2.a [v] IGMP proxy

1.3 Layer 2 WAN circuit technologies

1.3.a Implement and troubleshoot HDLC
1.3.b Implement and troubleshoot PPP
1.3.b [i] authentication [PAP, CHAP]
1.3.b [ii] PPPoE
1.3.b [iii] MLPPP

1.4 Troubleshooting layer 2 technologies

1.4.a Use IOS troubleshooting tools
1.4.a [i] debug, conditional debug
1.4.a [ii] ping, traceroute with extended options
1.4.a [iii] Embedded packet capture

1.4.b Apply troubleshooting methodologies
1.4.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]
1.4.b [ii] Design and implement valid solutions according to constraints
1.4.b [iii] Verify and monitor resolution

1.4.c Interpret packet capture
1.4.c [i] Using wireshark trace analyzer
1.4.c [ii] Using IOS embedded packet capture

2.0 Layer 3 Technologies 40%

2.1 Addressing technologies

2.1.a Identify, implement and troubleshoot IPv4 addressing and sub-netting
2.1.a [i] Address types, VLSM
2.1.a [ii] ARP

2.1.b Identify, implement and troubleshoot IPv6 addressing and sub-netting
2.1.b [i] Unicast, multicast
2.1.b [ii] EUI-64
2.1.b [iii] ND, RS/RA
2.1.b [iv] Autoconfig/SLAAC temporary addresses [RFC4941]

2.1.b [v] Global prefix configuration feature

2.2 Layer 3 Multicast

2.2.a Troubleshoot reverse path forwarding
2.2.a [i] RPF failure
2.2.a[ii] RPF failure with tunnel interface

2.2.b Implement and troubleshoot IPv4 protocol independent multicast
2.2.b [i] PIM dense mode, sparse mode, sparse-dense mode
2.2.b [ii] Static RP, auto-RP, BSR
2.2.b [iii] Bidirectional PIM
2.2.b [iv] Source-specific multicast
2.2.b [v] Group to RP mapping
2.2.b [vi] Multicast boundary

2.2.c Implement and troubleshoot multicast source discovery protocol
2.2.c.[i] Intra-domain MSDP [anycast RP]
2.2.c.[ii] SA filter

2.3 Fundamental routing concepts

2.3.a Implement and troubleshoot static routing
2.3.b Implement and troubleshoot default routing
2.3.c Compare routing protocol types
2.3.c [i] distance vector
2.3.c [ii] link state
2.3.c [iii] path vector

2.3.d Implement, optimize and troubleshoot administrative distance
2.3.e Implement and troubleshoot passive interface
2.3.f Implement and troubleshoot VRF lite
2.3.g Implement, optimize and troubleshoot filtering with any routing protocol
2.3.h Implement, optimize and troubleshoot redistribution between any routing protocol
2.3.i Implement, optimize and troubleshoot manual and auto summarization with any routing protocol
2.3.j Implement, optimize and troubleshoot policy-based routing
2.3.k Identify and troubleshoot sub-optimal routing
2.3.l Implement and troubleshoot bidirectional forwarding detection
2.3.m Implement and troubleshoot loop prevention mechanisms
2.3.m [i] Route tagging, filtering
2.3.m [ii] Split horizon
2.3.m [iii] Route poisoning

2.3.n Implement and troubleshoot routing protocol authentication
2.3.n [i] MD5

2.3.n [ii] key-chain
2.3.n [iii] EIGRP HMAC SHA2-256bit
2.3.n [iv] OSPFv2 SHA1-196bit
2.3.n [v] OSPFv3 IPsec authentication

2.4 RIP v2

2.4.a Implement and troubleshoot RIPv2

2.5 EIGRP [for IPv4 and IPv6]

2.5.a Describe packet types
2.5.a [i] Packet types [hello, query, update, and such]
2.5.a [ii] Route types [internal, external]

2.5.b Implement and troubleshoot neighbor relationship
2.5.b [i] Multicast, unicast EIGRP peering

2.5.c Implement and Troubleshoot Loop free path selection
2.5.c [i] RD, FD, FC, successor, feasible successor
2.5.c [ii] Classic metric
2.5.c [iii] Wide metric

2.5.d Implement and troubleshoot operations
2.5.d [i] General operations
2.5.d [ii] Topology table, update, query, active, passive
2.5.d [iii] Stuck in active
2.5.d [iv] Graceful shutdown

2.5.e Implement and troubleshoot EIGRP stub
2.5.e [i] stub
2.5.e [ii] leak-map

2.5.f Implement and troubleshoot load-balancing
2.5.f [i] equal-cost
2.5.f [ii] unequal-cost
2.5.f [iii] add-path

2.5.g Implement EIGRP [multi-address] named mode
2.5.g [i] Types of families
2.5.g [ii] IPv4 address-family
2.5.g [iii] IPv6 address-family

2.5.h Implement, troubleshoot and optimize EIGRP convergence and scalability
2.5.h [i] Describe fast convergence requirements
2.5.h [ii] Control query boundaries

2.5.h [iii] IP FRR/fast reroute [single hop]
2.5.h [iv] Summary leak-map
2.5.h [v] Summary metric

2.6 OSPF [v2 and v3]

2.6.a Describe packet types
2.6.a [i] LSA types [1, 2, 3, 4, 5, 7, 9]
2.6.a [ii] Route types [N1, N2, E1, E2]

2.6.b Implement and troubleshoot neighbor relationship
2.6.c Implement and troubleshoot OSPFv3 address-family support
2.6.c [i] IPv4 address-family
2.6.c [ii] IPv6 address-family

2.6.d Implement and troubleshoot network types, area types and router types
2.6.d [i] Point-to-point, multipoint, broadcast, non-broadcast
2.6.d [ii] LSA types, area type: backbone, normal, transit, stub, NSSA, totally stub
2.6.d [iii] Internal router, ABR, ASBR
2.6.d [iv] Virtual link

2.6.e Implement and troubleshoot path preference
2.6.f Implement and troubleshoot operations
2.6.f [i] General operations
2.6.f [ii] Graceful shutdown
2.6.f [iii] GTSM [generic TTL security mechanism]

2.6.g Implement, troubleshoot and optimize OSPF convergence and scalability
2.6.g [i] Metrics
2.6.g [ii] LSA throttling, SPF tuning, fast hello
2.6.g [iii] LSA propagation control [area types, ISPF]
2.6.g [iv] IP FR/fast reroute [single hop]
2.6.g [v] LFA/loop-free alternative [multi hop]
2.6.g [vi] OSPFv3 prefix suppression

2.7 BGP

2.7.a Describe, implement and troubleshoot peer relationships
2.7.a [i] Peer-group, template
2.7.a [ii] Active, passive
2.7.a [iii] States, timers
2.7.a [iv] Dynamic neighbors

2.7.b Implement and troubleshoot IBGP and EBGP
2.7.b [i] EBGP, IBGP
2.7.b [ii] 4 bytes AS number

2.7.b [iii] Private AS

2.7.c Explain attributes and best-path selection
2.7.d Implement, optimize and troubleshoot routing policies
2.7.d [i] Attribute manipulation
2.7.d [ii] Conditional advertisement
2.7.d [iii] Outbound route filtering
2.7.d [iv] Communities, extended communities
2.7.d [v] Multi-homing

2.7.e Implement and troubleshoot scalability
2.7.e [i] Route-reflector, cluster
2.7.e [ii] Confederations
2.7.e [iii] Aggregation, AS set

2.7.f Implement and troubleshoot multi-protocol BGP
2.7.f [i] IPv4, IPv6, VPN address-family

2.7.g Implement and troubleshoot AS path manipulations
2.7.g [i] Local AS, allow AS in, remove private AS
2.7.g [ii] Prepend
2.7.g [iii] Regexp

2.7.h Implement and Troubleshoot Other Features
2.7.h [i] Multipath
2.7.h [ii] BGP synchronization
2.7.h [iii] Soft reconfiguration, route refresh

2.8 Troubleshooting layer 3 technologies

2.8.a Use IOS troubleshooting tools
2.8.a [i] debug, conditional debug
2.8.a [ii] ping, traceroute with extended options
2.8.a [iii] Embedded packet capture

2.8.b Apply troubleshooting methodologies
2.8.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]
2.8.b [ii] Design and implement valid solutions according to constraints
2.8.b [iii] Verify and monitor resolution

2.8.c Interpret packet capture
2.8.c [i] Using wireshark trace analyzer
2.8.c [ii] Using IOS embedded packet capture

3.0 VPN Technologies 20%

3.1 Tunneling

3.1.a Implement and troubleshoot MPLS operations
3.1.a [i] Label stack, LSR, LSP
3.1.a [ii] LDP
3.1.a [iii] MPLS ping, MPLS traceroute

3.1.b Implement and troubleshoot basic MPLS L3VPN
3.1.b [i] L3VPN, CE, PE, P
3.1.b [ii] Extranet [route leaking]

3.1.c Implement and troubleshoot encapsulation
3.1.c [i] GRE
3.1.c [ii] Dynamic GRE

3.1.d Implement and troubleshoot DMVPN [single hub]
3.1.d [i] NHRP
3.1.d [ii] DMVPN with IPsec using preshared key
3.1.d [iii] QoS profile
3.1.d [iv] Pre-classify

3.2 Encryption

3.2.a Implement and troubleshoot IPsec with preshared key
3.2.a [i] IPv4 site to IPv4 site
3.2.a [ii] IPv6 in IPv4 tunnels
3.2.a [iii] Virtual tunneling interface [VTI]

3.3 Troubleshooting VPN technologies

3.3.a Use IOS troubleshooting tools
3.3.a [i] debug, conditional debug
3.3.a [ii] ping, traceroute with extended options
3.3.a [iii] Embedded packet capture

3.3.b Apply troubleshooting methodologies
3.3.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]
3.3.b [ii] Design and implement valid solutions according to constraints
3.3.b [iii] Verify and monitor resolution

3.3.c Interpret packet capture
3.3.c [i] Using wireshark trace analyzer

3.3.c [ii] Using IOS embedded packet capture

4.0 Infrastructure Security 5%

4.1 Device security

4.1.a Implement and troubleshoot IOS AAA using local database
4.1.b Implement and troubleshoot device access control
4.1.b [i] Lines [VTY, AUX, console]
4.1.b [ii] SNMP
4.1.b [iii] Management plane protection
4.1.b [iv] Password encryption

4.1.c Implement and troubleshoot control plane policing

4.2 Network security

4.2.a Implement and troubleshoot switch security features
4.2.a [i] VACL, PACL
4.2.a [ii] Stormcontrol
4.2.a [iii] DHCP snooping
4.2.a [iv] IP source-guard
4.2.a [v] Dynamic ARP inspection
4.2.a [vi] Port-security
4.2.a [vii] Private VLAN

4.2.b Implement and troubleshoot router security features
4.2.b [i] IPv4 access control lists [standard, extended, time-based]
4.2.b [ii] IPv6 traffic filter
4.2.b [iii] Unicast reverse path forwarding

4.2.c Implement and troubleshoot IPv6 first hop security
4.2.c [i] RA guard
4.2.c [ii] DHCP guard
4.2.c [iii] Binding table
4.2.c [iv] Device tracking
4.2.c [v] ND inspection/snooping
4.2.c [vi] Source guard
4.2.c [vii] PACL

4.3 Troubleshooting infrastructure security

4.3.a Use IOS troubleshooting tools
4.3.a [i] debug, conditional debug

4.3.a [ii] ping, traceroute with extended options

4.3.a [iii] Embedded packet capture

4.3.b Apply troubleshooting methodologies

4.3.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]

4.3.b [ii] Design and implement valid solutions according to constraints

4.3.b [iii] Verify and monitor resolution

4.3.c Interpret packet capture

4.3.c [i] Using wireshark trace analyzer

4.3.c [ii] Using IOS embedded packet capture

5.0 Infrastructure Services 15%

5.1 System management

5.1.a Implement and troubleshoot device management

5.1.a [i] Console and VTY

5.1.a [ii] telnet, HTTP, HTTPS, SSH, SCP

5.1.a [iii] [T]FTP

5.1.b Implement and troubleshoot SNMP

5.1.b [i] v2c, v3

5.1.c Implement and troubleshoot logging

5.1.c [i] Local logging, syslog, debug, conditional debug

5.1.c [ii] Timestamp

5.2 Quality of service

5.2.a Implement and troubleshoot end to end QoS

5.2.a [i] CoS and DSCP mapping

5.2.b Implement, optimize and troubleshoot QoS using MQC

5.2.b [i] Classification

5.2.b [ii] Network based application recognition [NBAR]

5.2.b [iii] Marking using IP precedence, DSCP, CoS, ECN

5.2.b [iv] Policing, shaping

5.2.b [v] Congestion management [queuing]

5.2.b [vi] HQoS, sub-rate ethernet link

5.2.b [vii] Congestion avoidance [WRED]

5.3 Network services

5.3.a Implement and troubleshoot first-hop redundancy protocols
5.3.a [i] HSRP, GLBP, VRRP
5.3.a [ii] Redundancy using IPv6 RS/RA

5.3.b Implement and troubleshoot network time protocol
5.3.b [i] NTP master, client, version 3, version 4
5.3.b [ii] NTP authentication

5.3.c Implement and troubleshoot IPv4 and IPv6 DHCP
5.3.c [i] DHCP client, IOS DHCP server, DHCP relay
5.3.c [ii] DHCP options
5.3.c [iii] DHCP protocol operations
5.3.c [iv] SLAAC/DHCPv6 interaction
5.3.c [v] Stateful, stateless DHCPv6
5.3.c [vi] DHCPv6 prefix delegation

5.3.d Implement and troubleshoot IPv4 network address translation
5.3.d [i] Static NAT, dynamic NAT, policy-based NAT, PAT
5.3.d [ii] NAT ALG

5.4 Network optimization

5.4.a Implement and troubleshoot IP SLA
5.4.a [i] ICMP, UDP, jitter, VoIP

5.4.b Implement and troubleshoot tracking object
5.4.b [i] Tracking object, tracking list
5.4.b [ii] Tracking different entities [e.g. interfaces, routes, IPSLA, and such]

5.4.c Implement and troubleshoot netflow
5.4.c [i] Netflow v5, v9
5.4.c [ii] Local retrieval
5.4.c [iii] Export [configuration only]
5.4.d Implement and troubleshoot embedded event manager
5.4.d [i] EEM policy using applet

5.5 Troubleshooting infrastructure services

5.5.a Use IOS troubleshooting tools
5.5.a [i] debug, conditional debug
5.5.a [ii] ping, traceroute with extended options
5.5.a [iii] Embedded packet capture

5.5.b Apply troubleshooting methodologies
5.5.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]

5.5.b [ii] Design and implement valid solutions according to constraints
5.5.b [iii] Verify and monitor resolution

5.5.c Interpret packet capture
5.5.c [i] Using wireshark trace analyzer
5.5.c [ii] Using IOS embedded packet capture

1.0 Layer 2 Technologies 20%

1.1 LAN switching technologies

1.1.a Implement and troubleshoot switch administration

1.1.a [i] Managing MAC address table

Managing MAC address table

The MAC address table contains address information that the switch uses to forward traffic between ports. All MAC addresses in the address table are associated with one or more ports. The address table includes these types of addresses:

- Dynamic address: a source MAC address that the switch learns and then ages when it is not in use.
- Static address: a manually entered unicast address that does not age and that is not lost when the switch resets.

By default, switches (IOS) age out MAC table entries after 300 seconds (5 mins). It's probably not something you'll have to tune or adjust often, but it's good to remember that it's happening. Further, you can change this value if needed with the global **mac address-table aging-time *value*** command.

You can disable and Enable MAC Address Learning on an interface or VLAN

By default, MAC address learning is enabled on all interfaces and VLANs on the router. You can control MAC address learning on an interface or VLAN to manage the available MAC address table space by controlling which interfaces or VLANs can learn MAC addresses. Before you disable MAC address learning, be sure that you are familiar with the network topology and the router system configuration. Disabling MAC address learning on an interface or VLAN could cause flooding in the network.

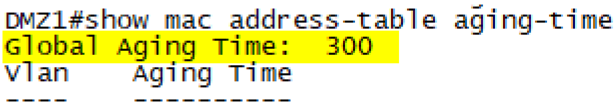
Note: Follow these guidelines when disabling MAC address learning on an interface or VLAN:

- Use caution before disabling MAC address learning on an interface or VLAN with a configured switch virtual interface (SVI). The switch then floods all IP packets in the Layer 2 domain.
- You can disable MAC address learning on a single VLAN ID from 1 to 4094 (for example, no mac address-table learning vlan 223) or a range of VLAN IDs, separated by a hyphen or comma (for example, no mac address-table learning vlan 1-10, 15).
- We recommend that you disable MAC address learning only in VLANs with two ports. If you disable MAC address learning on a VLAN with more than two ports, every packet entering the switch is flooded in that VLAN domain.
- You cannot disable MAC address learning on a VLAN that is used internally by the router. If the VLAN ID that you enter is an internal VLAN, the switch generates an error message and rejects the command. To view internal VLANs in use, enter the show vlan internal usage privileged EXEC command.
- If you disable MAC address learning on a VLAN that includes a secure port, MAC address learning is not disabled on that port.

Command	Description
show mac address-table address	Displays MAC address table information for the specified MAC address.
show mac address-table aging-time	Displays the aging time in all VLANs or the specified VLAN.
show mac address-table count	Displays the number of addresses present in all VLANs or the specified VLAN.
show mac address-table dynamic	Displays only dynamic MAC address table entries.

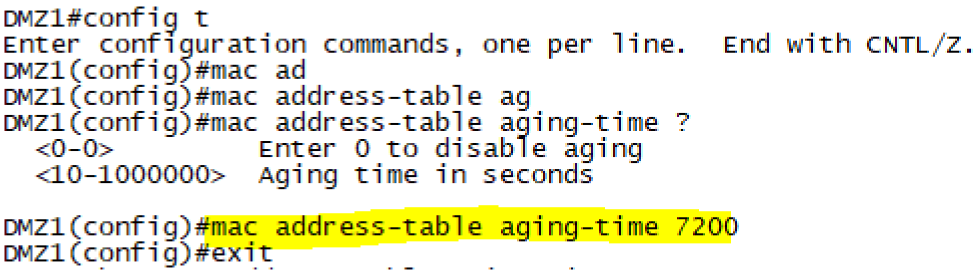
show mac address-table interface	Displays the MAC address table information for the specified interface.
show mac address-table learning	Displays MAC address learning status of all VLANs or the specified VLAN.
show mac address-table static	Displays only static MAC address table entries.
show mac address-table vlan	Displays the MAC address table information for the specified VLAN.

Here is the example screen shot to verify default settings of mac address aging-time (which is 300seconds (5min) by default)



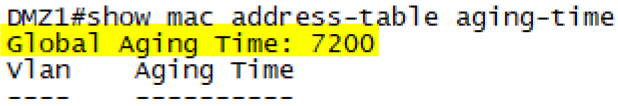
```
DMZ1#show mac address-table aging-time
Global Aging Time:  300
Vlan    Aging Time
----    ----------
```

Here is the example we change aging time to 7200 Seconds(2hours)



```
DMZ1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
DMZ1(config)#mac ad
DMZ1(config)#mac address-table ag
DMZ1(config)#mac address-table aging-time ?
  <0-0>         Enter 0 to disable aging
  <10-1000000>  Aging time in seconds

DMZ1(config)#mac address-table aging-time 7200
DMZ1(config)#exit
```

Here is the output after we change to 2hours mac address aging time.



```
DMZ1#show mac address-table aging-time
Global Aging Time: 7200
Vlan    Aging Time
----    ----------
```

Notes: Nexus OS for Nexus 7K has different mac aging time out 1800 seconds (at the time of writing this document, this may change check the cisco document).

Nexus 5000 remain same as 300 Seconds by default.

1.1.a [ii] errdisable recovery

If the configuration shows a port to be enabled, but software on the switch detects an error situation on the port, the software shuts down that port. In other words, the port is automatically disabled by the switch operating system software because of an error condition that is encountered on the port.

When a port is error disabled, it is effectively shut down and no traffic is sent or received on that port. The port LED is set to the color orange and, when you issue the **show interfaces** command, the port status shows err-disabled. Here is an example of what an error-disabled port looks like from the command-line interface (CLI) of the switch.

Causes of Errdisable

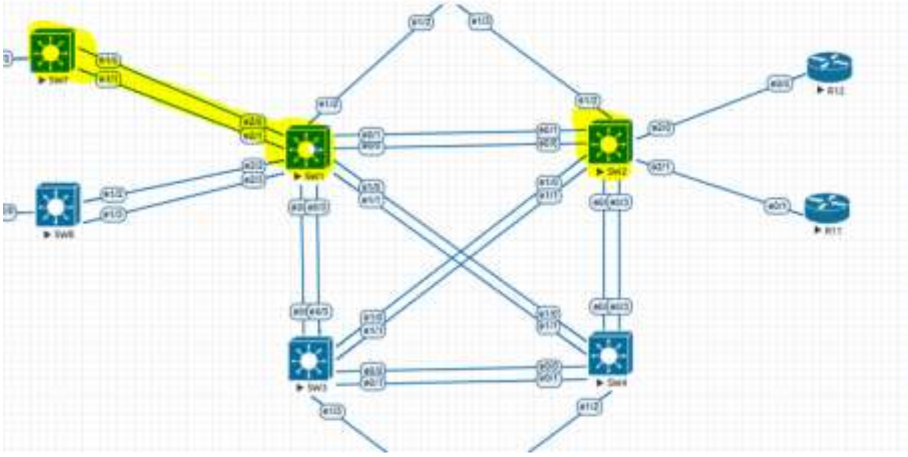
This feature was first implemented in order to handle special collision situations in which the switch detected excessive or late collisions on a port. Excessive collisions occur when a frame is dropped because the switch encounters 16 collisions in a row. Late collisions occur after every device on the wire should have recognized that the wire was in use. Possible causes of these types of errors include:

A cable that is out of specification (either too long, the wrong type, or defective)
A bad network interface card (NIC) card (with physical problems or driver problems)
A port duplex misconfiguration

A port duplex misconfiguration is a common cause of the errors because of failures to negotiate the speed and duplex properly between two directly connected devices (for example, a NIC that connects to a switch). Only half-duplex connections should ever have collisions in a LAN. Because of the carrier sense multiple access (CSMA) nature of Ethernet, collisions are normal for half duplex, as long as the collisions do not exceed a small percentage of traffic.
There are various reasons for the interface to go into errdisable. The reason can be:

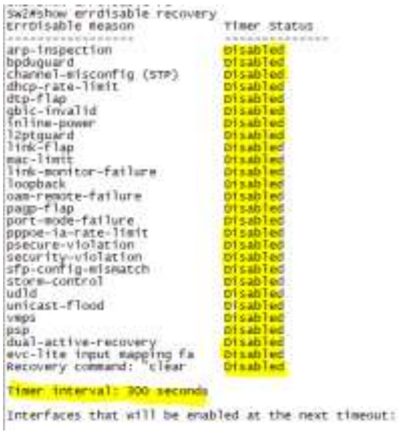
- Duplex mismatch
- Port channel misconfiguration
- BPDU guard violation
- UniDirectional Link Detection (UDLD) condition
- Late-collision detection
- Link-flap detection
- Security violation
- Port Aggregation Protocol (PAgP) flap
- Layer 2 Tunneling Protocol (L2TP) guard
- DHCP snooping rate-limit
- Incorrect GBIC / Small Form-Factor Pluggable (SFP) module or cable
- Address Resolution Protocol (ARP) inspection
- Inline power

Configuration Example : (below one of test setup)



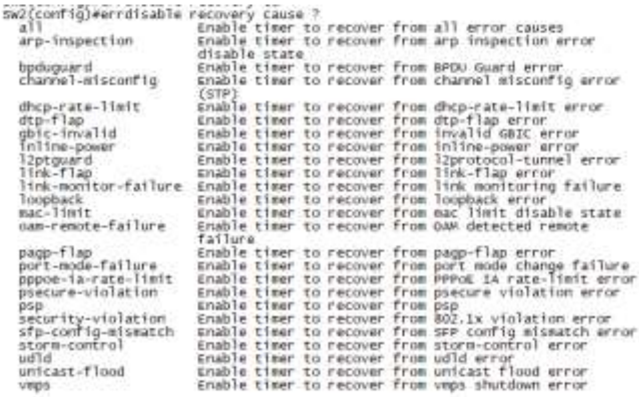
SW2 have default configuration :

All errdisable recovery disabled by default and recovery time is 300 Seconds by default



```
Sw2#show errdisable recovery
ErrDisable Reason            Timer Status
-----------------            --------------
arp-inspection               Disabled
bpduguard                    Disabled
channel-misconfig (STP)      Disabled
dhcp-rate-limit              Disabled
dtp-flap                     Disabled
gbic-invalid                 Disabled
inline-power                 Disabled
l2ptguard                    Disabled
link-flap                    Disabled
mac-limit                    Disabled
link-monitor-failure         Disabled
loopback                     Disabled
oam-remote-failure           Disabled
pagp-flap                    Disabled
port-mode-failure            Disabled
pppoe-ia-rate-limit          Disabled
psecure-violation            Disabled
security-violation           Disabled
sfp-config-mismatch          Disabled
storm-control                Disabled
udld                         Disabled
unicast-flood                Disabled
vmps                         Disabled
psp                          Disabled
dual-active-recovery         Disabled
evc-lite input mapping fa    Disabled
Recovery command: "clear     Disabled

Timer interval: 300 seconds

Interfaces that will be enabled at the next timeout:
```

Either you can configure recovery each reason or you can do globally enable all recovery to enabled:



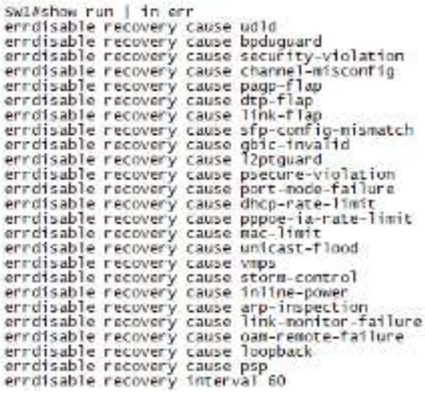
```
sw2(config)#errdisable recovery cause ?
  all                     Enable timer to recover from all error causes
  arp-inspection          Enable timer to recover from arp inspection error
                          disable state
  bpduguard               Enable timer to recover from BPDU Guard error
  channel-misconfig       Enable timer to recover from channel misconfig error
                          (STP)
  dhcp-rate-limit         Enable timer to recover from dhcp-rate-limit error
  dtp-flap                Enable timer to recover from dtp-flap error
  gbic-invalid            Enable timer to recover from invalid GBIC error
  inline-power            Enable timer to recover from inline-power error
  l2ptguard               Enable timer to recover from l2protocol-tunnel error
  link-flap               Enable timer to recover from link-flap error
  link-monitor-failure    Enable timer to recover from link monitoring failure
  loopback                Enable timer to recover from loopback error
  mac-limit               Enable timer to recover from mac limit disable state
  oam-remote-failure      Enable timer to recover from OAM detected remote
                          failure
  pagp-flap               Enable timer to recover from pagp-flap error
  port-mode-failure       Enable timer to recover from port mode change failure
  pppoe-ia-rate-limit     Enable timer to recover from PPPoE IA rate-limit error
  psecure-violation       Enable timer to recover from psecure violation error
  psp                     Enable timer to recover from psp
  security-violation      Enable timer to recover from 802.1x violation error
  sfp-config-mismatch     Enable timer to recover from SFP config mismatch error
  storm-control           Enable timer to recover from storm-control error
  udld                    Enable timer to recover from udld error
  unicast-flood           Enable timer to recover from unicast flood error
  vmps                    Enable timer to recover from vmps shutdown error
```

You can also change error recovery time in Seconds.

```
SW2(config)#errdisable recovery interval ?
  <30-86400>  timer-interval(sec)
```

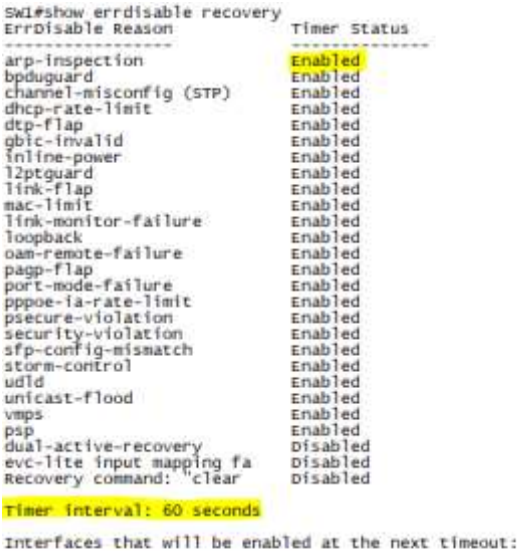
On SW1 I have enabled all for testing as below and change recovery time to 60Seconds (1Min)

Here is my running configuration on SW1



```
SW1#show run | in err
errdisable recovery cause udld
errdisable recovery cause bpduguard
errdisable recovery cause security-violation
errdisable recovery cause channel-misconfig
errdisable recovery cause pagp-flap
errdisable recovery cause dtp-flap
errdisable recovery cause link-flap
errdisable recovery cause sfp-config-mismatch
errdisable recovery cause gbic-invalid
errdisable recovery cause l2ptguard
errdisable recovery cause psecure-violation
errdisable recovery cause port-mode-failure
errdisable recovery cause dhcp-rate-limit
errdisable recovery cause pppoe-ia-rate-limit
errdisable recovery cause mac-limit
errdisable recovery cause unicast-flood
errdisable recovery cause vmps
errdisable recovery cause storm-control
errdisable recovery cause inline-power
errdisable recovery cause arp-inspection
errdisable recovery cause link-monitor-failure
errdisable recovery cause oam-remote-failure
errdisable recovery cause loopback
errdisable recovery cause psp
errdisable recovery interval 60
```

Output of above configuration as below:



```
Sw1#show errdisable recovery
ErrDisable Reason            Timer Status
-----------------            --------------
arp-inspection               Enabled
bpduguard                    Enabled
channel-misconfig (STP)      Enabled
dhcp-rate-limit              Enabled
dtp-flap                     Enabled
gbic-invalid                 Enabled
inline-power                 Enabled
l2ptguard                    Enabled
link-flap                    Enabled
mac-limit                    Enabled
link-monitor-failure         Enabled
loopback                     Enabled
oam-remote-failure           Enabled
pagp-flap                    Enabled
port-mode-failure            Enabled
pppoe-ia-rate-limit          Enabled
psecure-violation            Enabled
security-violation           Enabled
sfp-config-mismatch          Enabled
storm-control                Enabled
udld                         Enabled
unicast-flood                Enabled
vmps                         Enabled
psp                          Enabled
dual-active-recovery         Disabled
evc-lite input mapping fa    Disabled
Recovery command: "clear     Disabled

Timer interval: 60 seconds

Interfaces that will be enabled at the next timeout:
```

Let’s do some testing’s

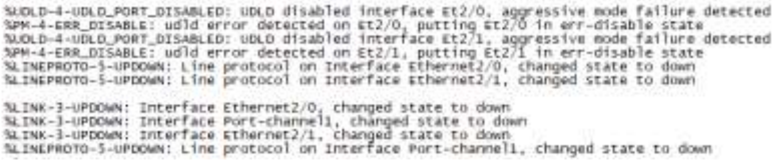
By Default, errdisable recovery – disabled and we can see behavior as below.

Here is the SW1 and SW7 Trunk configuration:

```
SW1#show run interface port-channel 1
Building configuration...

Current configuration : 184 bytes
!
interface Port-channel1
 description DM1Z1
 switchport trunk allowed vlan 2-9,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
end

SW1#show run interface eth 2/0
Building configuration...

Current configuration : 255 bytes
!
interface Ethernet2/0
 description DM1Z1
 switchport trunk allowed vlan 2-9,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 1 mode on
 spanning-tree guard loop
end

SW1#show run interface eth 2/1
Building configuration...

Current configuration : 255 bytes
!
interface Ethernet2/1
 description DM1Z1
 switchport trunk allowed vlan 2-9,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 1 mode on
 spanning-tree guard loop
end
```

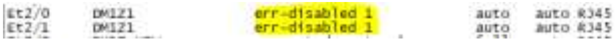
```
DMZ1#show run interface port-cha 1
Building configuration...

Current configuration : 182 bytes
!
interface Port-channel1
 description SW1
 switchport trunk allowed vlan 2-9,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
end

DMZ1#show run interface eth 1/0
Building configuration...

Current configuration : 253 bytes
!
interface Ethernet1/0
 description SW1
 switchport trunk allowed vlan 2-9,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 1 mode on
 spanning-tree guard loop
end

DMZ1#show run interface eth 1/1
Building configuration...

Current configuration : 253 bytes
!
interface Ethernet1/1
 description SW1
 switchport trunk allowed vlan 2-9,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 1 mode on
 spanning-tree guard loop
end
```

I have turned off SW7 for testing. And we can see the below messages in SW1



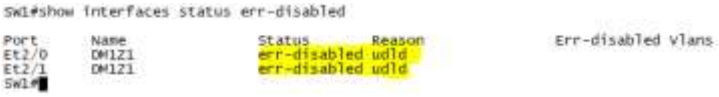
```
%UDLD-4-UDLD_PORT_DISABLED: UDLD disabled interface Et2/0, aggressive mode failure detected
%PM-4-ERR_DISABLE: udld error detected on Et2/0, putting Et2/0 in err-disable state
%UDLD-4-UDLD_PORT_DISABLED: UDLD disabled interface Et2/1, aggressive mode failure detected
%PM-4-ERR_DISABLE: udld error detected on Et2/1, putting Et2/1 in err-disable state
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet2/0, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet2/1, changed state to down

%LINK-3-UPDOWN: Interface Ethernet2/0, changed state to down
%LINK-3-UPDOWN: Interface Port-channel1, changed state to down
%LINK-3-UPDOWN: Interface Ethernet2/1, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1, changed state to down
```

Port mode change to err-disabled.



```
Et2/0      DM121                err-disabled 1          auto   auto RJ45
Et2/1      DM121                err-disabled 1          auto   auto RJ45
```

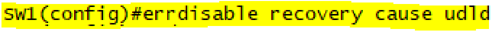
Disabled reason UDLD



```
SW1#show interfaces status err-disabled

Port      Name                Status       Reason              Err-disabled Vlans
Et2/0     DM121               err-disabled udld
Et2/1     DM121               err-disabled udld
SW1#
```

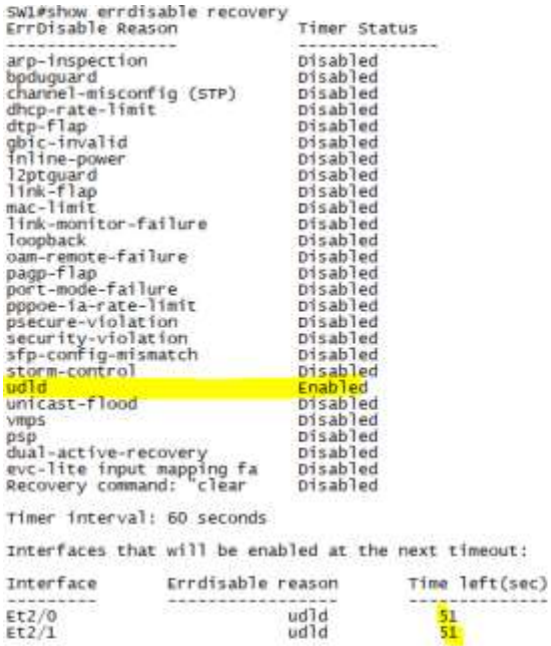
Even though I have powered SW7 but we did not see the link come back online again, because of err-disabled mode, in this case either we need to shutdown the Port-channel and do no shutdown to restore or configure error recovery.

Let us configure UDLD recovery configuration



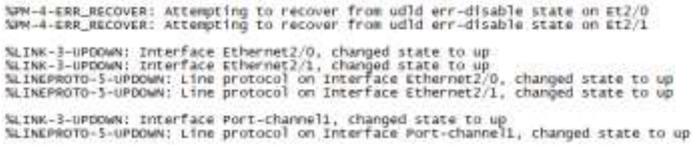
```
SW1(config)#errdisable recovery cause udld
```

As soon we configure above command we can see countdown started, as we configured 60 seconds recovery.



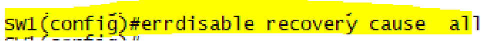
```
SW1#show errdisable recovery
ErrDisable Reason            Timer Status
-----------------            --------------
arp-inspection               Disabled
bpduguard                    Disabled
channel-misconfig (STP)      Disabled
dhcp-rate-limit              Disabled
dtp-flap                     Disabled
gbic-invalid                 Disabled
inline-power                 Disabled
l2ptguard                    Disabled
link-flap                    Disabled
mac-limit                    Disabled
link-monitor-failure         Disabled
loopback                     Disabled
oam-remote-failure           Disabled
pagp-flap                    Disabled
port-mode-failure            Disabled
pppoe-ia-rate-limit          Disabled
psecure-violation            Disabled
security-violation           Disabled
sfp-config-mismatch          Disabled
storm-control                Disabled
udld                         Enabled
unicast-flood                Disabled
vmps                         Disabled
psp                          Disabled
dual-active-recovery         Disabled
evc-lite input mapping fa    Disabled
Recovery command: "clear     Disabled

Timer interval: 60 seconds

Interfaces that will be enabled at the next timeout:

Interface       Errdisable reason       Time left(sec)
---------       -----------------       --------------
Et2/0                      udld               51
Et2/1                      udld               51
```

After 60 Seconds the Port-channel come up automatically.

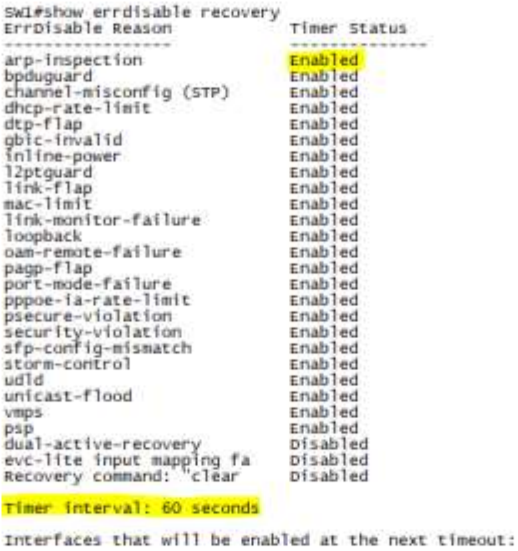


```
%PM-4-ERR_RECOVER: Attempting to recover from udld err-disable state on Et2/0
%PM-4-ERR_RECOVER: Attempting to recover from udld err-disable state on Et2/1

%LINK-3-UPDOWN: Interface Ethernet2/0, changed state to up
%LINK-3-UPDOWN: Interface Ethernet2/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet2/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet2/1, changed state to up

%LINK-3-UPDOWN: Interface Port-channel1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1, changed state to up
```

Now enabled for all recovery.



```
SW1(config)#errdisable recovery cause  all
```



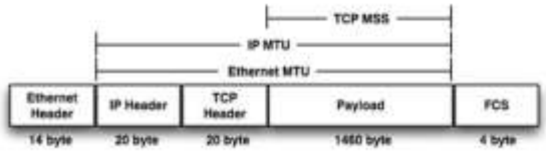
```
Sw1#show errdisable recovery
ErrDisable Reason            Timer Status
-----------------            --------------
arp-inspection               Enabled
bpduguard                    Enabled
channel-misconfig (STP)      Enabled
dhcp-rate-limit              Enabled
dtp-flap                     Enabled
gbic-invalid                 Enabled
inline-power                 Enabled
l2ptguard                    Enabled
link-flap                    Enabled
mac-limit                    Enabled
link-monitor-failure         Enabled
loopback                     Enabled
oam-remote-failure           Enabled
pagp-flap                    Enabled
port-mode-failure            Enabled
pppoe-ia-rate-limit          Enabled
psecure-violation            Enabled
security-violation           Enabled
sfp-config-mismatch          Enabled
storm-control                Enabled
udld                         Enabled
unicast-flood                Enabled
vmps                         Enabled
psp                          Enabled
dual-active-recovery         Disabled
evc-lite input mapping fa    Disabled
Recovery command: "clear     Disabled

Timer interval: 60 seconds

Interfaces that will be enabled at the next timeout:
```

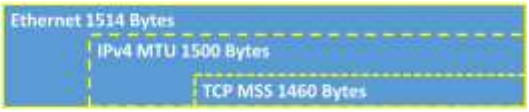
More information can be found in Cisco site.

https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/69980-errdisable-recovery.html

1.1.a [iii] L2 MTU

The *Maximum Transmission Unit (MTU)* is the maximum length of data that can be transmitted by a protocol in one instance. For example, the MTU of Ethernet (by default 1500) is the largest number of bytes that can be carried by an Ethernet frame (excluding the header and trailer). MTUs are found at various layers of the OSI model, and can often be tweaked to more efficiently transport large volumes of data.



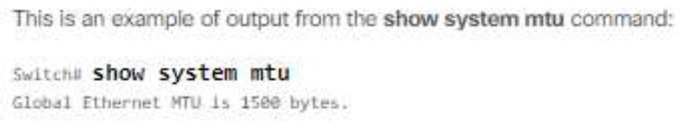


bytes	Set the system MTU for ports that are set to 10 or 100 Mb/s. The range is 1500 to 1998 bytes. This is the maximum MTU received at 10/100-Mb/s Ethernet switch ports.
jumbo *bytes*	Set the system jumbo MTU for Gigabit Ethernet ports operating at 1000 Mb/s or greater. The range is 1500 to 9000 bytes. This is the maximum MTU received at the physical port for Gigabit Ethernet ports.
routing*bytes*	Set the maximum MTU for routed packets. You can also set the maximum MTU to be advertised by the routing protocols that support the configured MTU size. The range is 1500 bytes to the system MTU value. The system routing MTU is the maximum MTU for routed packets and is also the maximum MTU that the switch advertises in routing updates for protocols such as OSPF.

MTU Manipulation:

Ethernet

The default Ethernet MTU is 1500 bytes, not including the header or trailer. Sometimes a slightly higher MTU is preferable to accommodate Q-in-Q tunneling or other encapsulation. The MTU can be raised on Cisco IOS with the *system mtu* command under global configuration:



This is an example of output from the **show system mtu** command:

```
Switch# show system mtu
Global Ethernet MTU is 1500 bytes.
```

IP

As with Ethernet frames, the MTU can be adjusted for IP packets. However, the IP MTU is configured per interface rather than system-wide, with the ***ip mtu*** command:

TCP

There are two contexts in which the TCP *Maximum Segment Size (MSS)* can be configured: **transient traffic and terminating traffic**.

Transient Traffic

When a TCP client initiates a connection to a server, it includes its MSS as an option in the first (SYN) packet. On an Ethernet interface, this value is typically 1460 (1500 byte Ethernet MTU - 20 byte IP header - 20 byte TCP header).

However, links beyond the host often have a lower effective MSS and full-size packets from the client may be dropped. To inspect and alter the MSS option included in TCP SYN packets passing through the router, use the *ip tcp adjust-mss* command on the interface:

Terminating Traffic

Terminating traffic refers to TCP packets which originate from or are destined for the local router (for example, SSH or BGP). In this context, the router itself is considered the TCP client and/or server. The local MSS can be configured with the *ip tcp mss* command under global configuration:
More information good example from Cisco:

https://www.cisco.com/c/en/us/support/docs/ip/generic-routing-encapsulation-gre/25885-pmtud-ipfrag.html

1.1.b Implement and troubleshoot layer 2 protocols

1.1.b [i] CDP, LLDP

Cisco Discovery Protocol (CDP)
– runs over Layer 2 (data link layer)

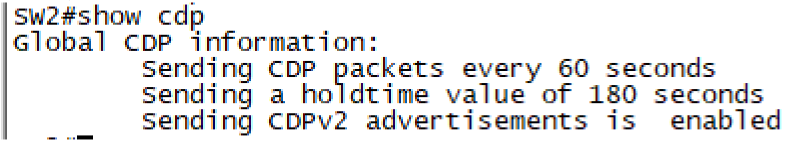
– used by all Cisco devices
– sends periodic messages to a multicast address
– advertises at least one IP address that can receive SNMP messages
– advertisements contain time-to-live or holddown information

Default CDP Configuration
– CDP global state: enabled
– CDP interface state: enabled
– CDP timer: 60 seconds
– CDP holdtime: 180 seconds
– CDP Version-2 advertisements: enabled

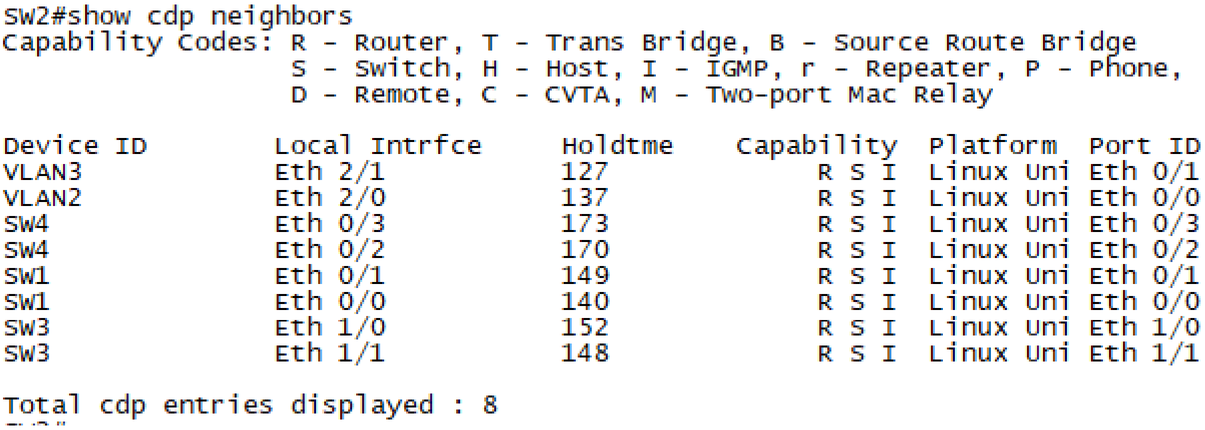
Link Layer Discovery Protocol (LLDP)
– supports non-Cisco devices and allow interoperability between those devices
– neighbor discovery protocol
– runs over the data link layer
– LLDP attributes are called TLVs (type, length, value descriptions)

Default LLDP Configuration
– LLDP global state: disabled
– LLDP holdtime: 120 seconds
– LLDP timer: 30 seconds
– LLDP reinitialization delay: 2 seconds
– LLDP tlv-select: disabled to send an receive all TLVs
– LLDP interface state: disabled
– LLDP receive: disabled
– LLDP transmit: disabled
– LLDP med-tlv-select: disabled to send all LLDP-MED TLVs

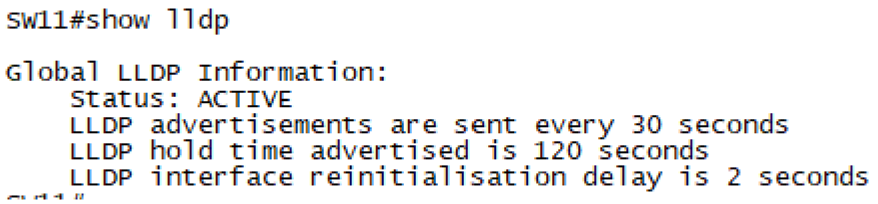
Examples CDP :



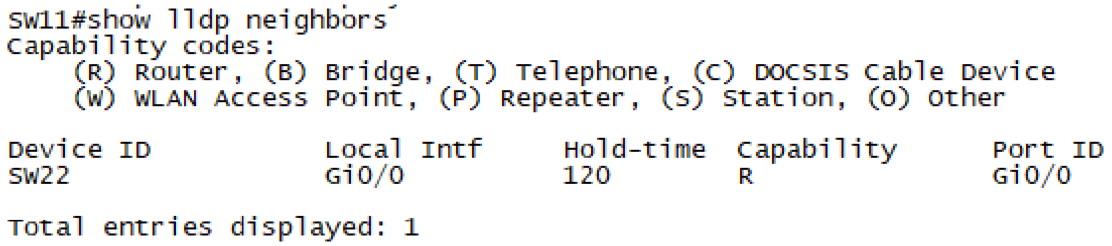
```
SW2#show cdp
Global CDP information:
        Sending CDP packets every 60 seconds
        Sending a holdtime value of 180 seconds
        Sending CDPv2 advertisements is  enabled
```



```
SW2#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone,
                  D - Remote, C - CVTA, M - Two-port Mac Relay

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
VLAN3            Eth 2/1           127             R S I  Linux Uni Eth 0/1
VLAN2            Eth 2/0           137             R S I  Linux Uni Eth 0/0
SW4              Eth 0/3           173             R S I  Linux Uni Eth 0/3
SW4              Eth 0/2           170             R S I  Linux Uni Eth 0/2
SW1              Eth 0/1           149             R S I  Linux Uni Eth 0/1
SW1              Eth 0/0           140             R S I  Linux Uni Eth 0/0
SW3              Eth 1/0           152             R S I  Linux Uni Eth 1/0
SW3              Eth 1/1           148             R S I  Linux Uni Eth 1/1

Total cdp entries displayed : 8
```



```
SW11#show lldp

Global LLDP Information:
    Status: ACTIVE
    LLDP advertisements are sent every 30 seconds
    LLDP hold time advertised is 120 seconds
    LLDP interface reinitialisation delay is 2 seconds
```



```
SW11#show lldp neighbors
Capability codes:
    (R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
    (W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other

Device ID           Local Intf     Hold-time  Capability      Port ID
SW22                Gi0/0          120        R               Gi0/0

Total entries displayed: 1
```

Monitoring CDP

clear cdp counters
clear cdp table
show cdp
show cdp entry
show cdp int fa0/1
show cdp neighbors
show cdp neighbors fa0/1 detail
show cdp traffic

Monitoring LLDP

clear lldp counters
clear lldp table
show lldp entry
show lldp interface
show lldp neighbors
show lldp neighbors fa0/1 detail
show lldp traffic

1.1.b [ii] UDLD

Unidirectional Link Detection (UDLD) is a Cisco Proprietary protocol used to detect issues with links that utilize separate cables (fiber).
UDLD can be enabled and becomes active when it detects a neighbor from the other side (both sides must be configured)

UDLD can only successfully work when configured on both ends of a link.
UDLD protects us from:
- Connecting cables to ourselves by accident
- A link connected to a HUB or dumb switch (multiple UDLD peers on one link is not good)
- Cut transmit or receive (in aggressive mode)
- Misconnected fiber strands (connected to proper device but wrong ports)

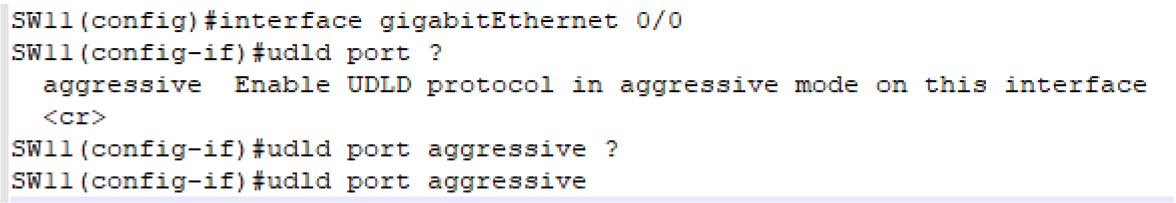
If UDLD hits the hold time for a neighbor, before declaring the number dead the UDLD enabled device will shoot 8 UDLD messages in a row to the neighbor.

UDLD hellos are sent by default every 15 seconds.

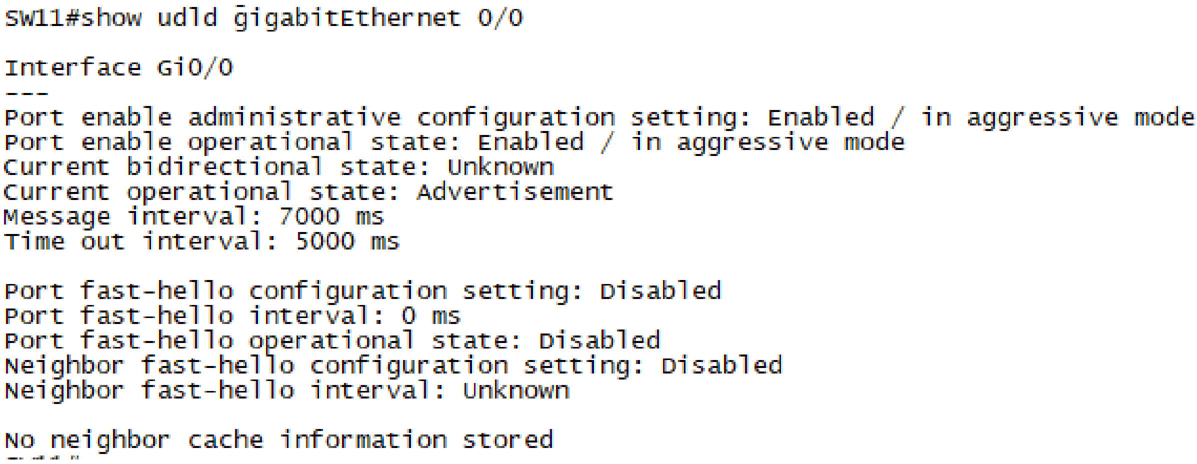
UDLD is disabled by default.
There are 2 ways to enable UDLD, globally or per interface.
If we enable UDLD globally, it ONLY is enabled on fiber interfaces.

conf t
udld <enable | aggressive>   <-- globally enabled)

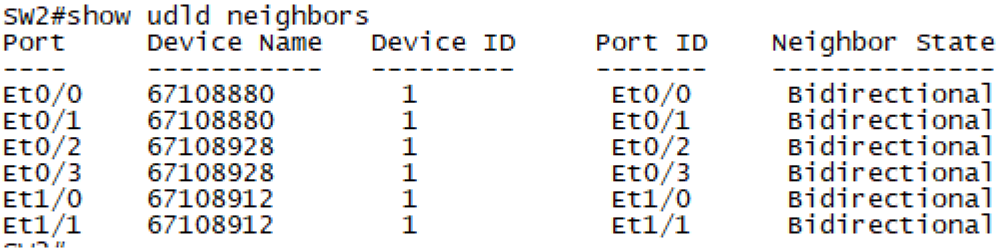
You can also configure per interface to override global config



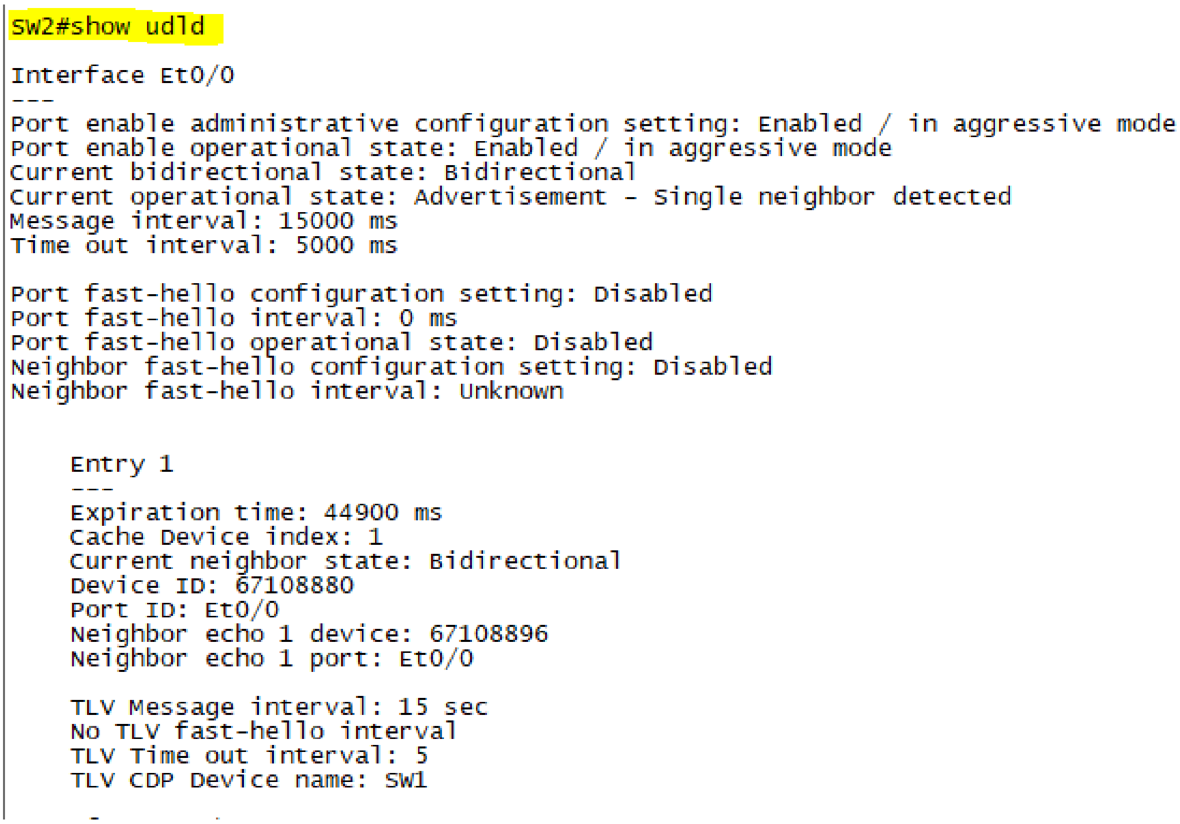
```
SW11(config)#interface gigabitEthernet 0/0
SW11(config-if)#udld port ?
  aggressive  Enable UDLD protocol in aggressive mode on this interface
  <cr>
SW11(config-if)#udld port aggressive ?
SW11(config-if)#udld port aggressive
```



```
Sw11#show udld gigabitEthernet 0/0

Interface Gi0/0
---
Port enable administrative configuration setting: Enabled / in aggressive mode
Port enable operational state: Enabled / in aggressive mode
Current bidirectional state: Unknown
Current operational state: Advertisement
Message interval: 7000 ms
Time out interval: 5000 ms

Port fast-hello configuration setting: Disabled
Port fast-hello interval: 0 ms
Port fast-hello operational state: Disabled
Neighbor fast-hello configuration setting: Disabled
Neighbor fast-hello interval: Unknown

No neighbor cache information stored
```



```
Sw2#show udld neighbors
Port     Device Name   Device ID     Port ID    Neighbor State
----     -----------   ---------     -------    --------------
Et0/0    67108880        1            Et0/0       Bidirectional
Et0/1    67108880        1            Et0/1       Bidirectional
Et0/2    67108928        1            Et0/2       Bidirectional
Et0/3    67108928        1            Et0/3       Bidirectional
Et1/0    67108912        1            Et1/0       Bidirectional
Et1/1    67108912        1            Et1/1       Bidirectional
```



```
SW2#show udld

Interface Et0/0
---
Port enable administrative configuration setting: Enabled / in aggressive mode
Port enable operational state: Enabled / in aggressive mode
Current bidirectional state: Bidirectional
Current operational state: Advertisement - Single neighbor detected
Message interval: 15000 ms
Time out interval: 5000 ms

Port fast-hello configuration setting: Disabled
Port fast-hello interval: 0 ms
Port fast-hello operational state: Disabled
Neighbor fast-hello configuration setting: Disabled
Neighbor fast-hello interval: Unknown


    Entry 1
    ---
    Expiration time: 44900 ms
    Cache Device index: 1
    Current neighbor state: Bidirectional
    Device ID: 67108880
    Port ID: Et0/0
    Neighbor echo 1 device: 67108896
    Neighbor echo 1 port: Et0/0

    TLV Message interval: 15 sec
    No TLV fast-hello interval
    TLV Time out interval: 5
    TLV CDP Device name: SW1
```

1.1.c Implement and troubleshoot VLAN

1.1.c [i] access ports

Most of Cisco's switches, especially those targeted at access-layer switches default to "**switchport mode dynamic desirable**."

This means that DTP is turned on and is actively trying to force ports into a trunking state. A good practice for all of your switches is to adjust that to a more secure setting the moment you take it out of the box.

Note : So do not connect any unboxed switches to Live network, this will cause huge interruption of service due to DTP on by default.

Any new Switch need to be configured properly and appropriate configuration before you join them back to Live network.

You want to turn off the DTP, then you can use command "**switchport nonegotiate"**

You want to set the port to Access port **"switchport mode access" (**Access ports are used to configure end devices where we want to restrict their membership to a certain vlan.)

If an access port receives a frame with a .1q header, it will drop it.

By Default, all the Access ports are configured to VLAN 1

4 port types available :
access port
Trunk (ISL or .1q)
dynamic access
voice vlan

To configure a port as access use VLAN 100:

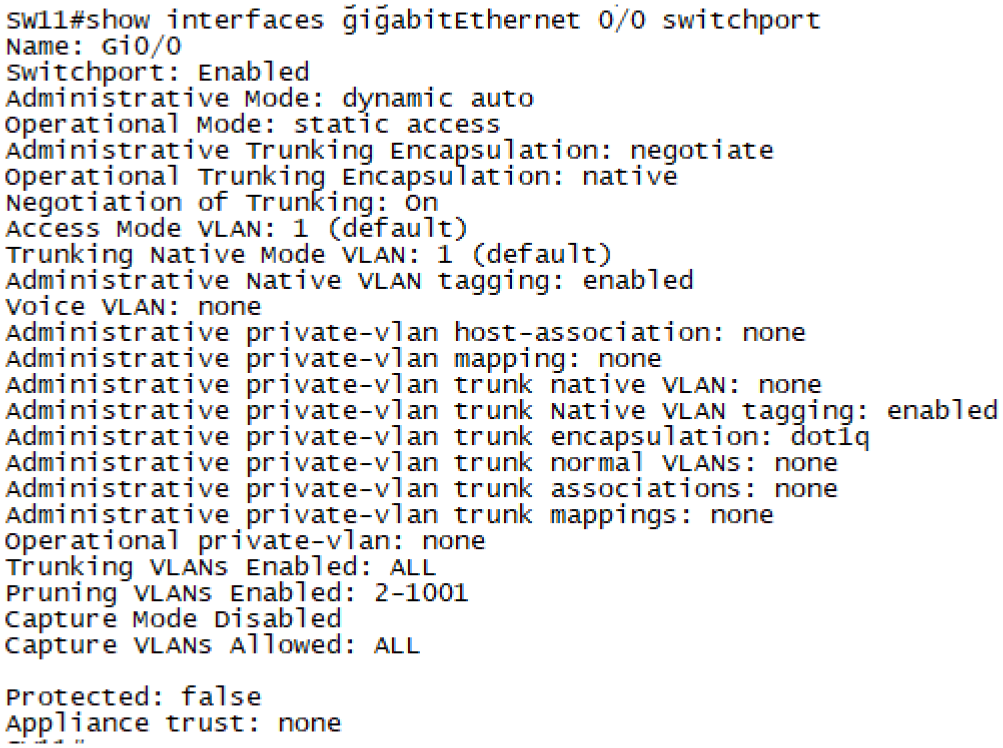
```
int g0/1
switchport mode access
switchport access vlan 100
```

There are also switchport macros which auto configure ports based on a basic set of commands that most people use. For example :

```
int g0/1
switchport host
```

This does 3 things:
switchport mode will be set to access
spanning-tree portfast will be enabled
channel group will be disabled

Below Default configuration of the Port:



```
SW11#show interfaces gigabitEthernet 0/0 switchport
Name: Gi0/0
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan: none
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Appliance trust: none
```

1.1.c [ii] VLAN database

The vlan database is what holds all VLAN and some VTP configuration.
This database is stored as the file vlan.dat in NVRAM/flash.

Since vlan information is not entirely saved in the running or startup config, an erase of these configs will not remove vlans from the switch.
VLANs are automatically added when you put a port into a non-existant VLAN with the **switchport access vlan** *number* command
You can create a VLAN
#Config t
Vlan 3 (this will create a Vlan number 3)
Name HRSTAFF (this is for identification of VLAN pupose, example HR Staff)

How to verify the VLANs in the database?
#show vlan

Delete the all VLANs from the device
#delete vlan.dat
Note: extended range vlans are not stored in the VLAN database

1.1.c [iii] normal, extended VLAN, voice VLAN

The **normal** range is 1 – 1001
The reserved range is 1002– 1005
The **extended** range is 1006 – 4094
The reserved range is reserved for FDDI and Token ring vlans.
If we use the extended range the switch must have the switch in vtp transparent mode or use vtp v3
Voice VLAN, this allows phones to tag traffic over an access port. When a Cisco phone is plugged into a port running CDP and a voice vlan, the switch uses CDP to tell the phone "tag your own traffic with this voice VLAN 802.1q tag, and leave traffic from devices attached through you untagged."

1.1.d Implement and troubleshoot trunking

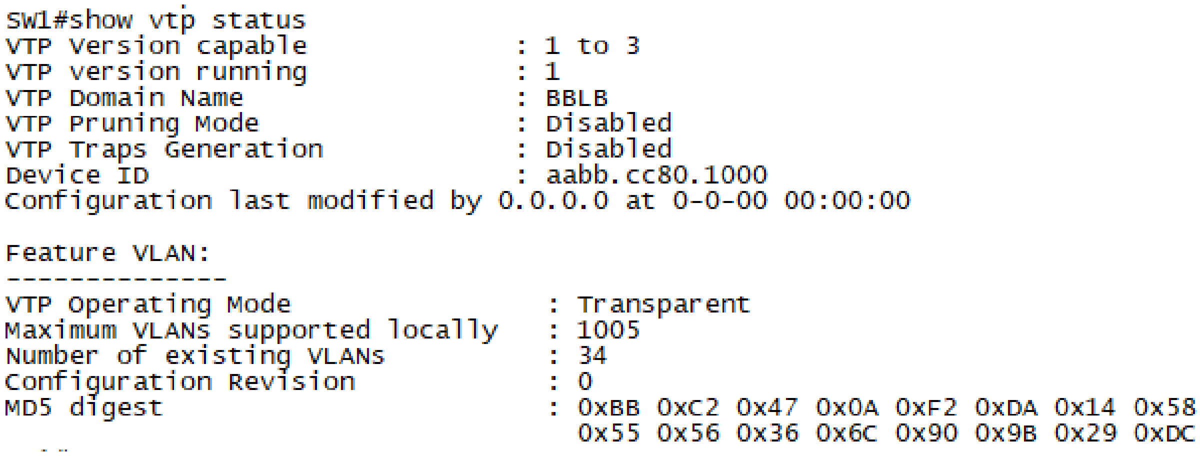
1.1.d [i] VTPv1, VTPv2, VTPv3, VTP pruning

VTP is a called as Layer 2 messaging protocol that contains VLAN configuration information by managing the addition, deletion, and renaming of VLANs within a VTP domain. A VTP domain (also called a VLAN management domain) is made up of one or more network devices that share the same VTP domain name and that are interconnected with trunks.

VTP Versions

VTP has three versions and these versions are VTPv1, VTPv2 and VTPv3. Let me share some of the insights which will show you the feature and the capabilities of the VTP versions.
VTPv1 and VTPv2 supports VLAN from 1 to 1000, while VTPv3 supports from VLAN 1 to 4096.
Extended-range VLANs are supported only in VTP version 3. If converting from VTP version 3 to VTP version 2, VLANs in the range 1006 to 4094 are removed from VTP control.
VTP version 3 supports propagation of any database in a domain by allowing you to configure a primary and secondary server
VTP version 3 is not supported on private VLAN (PVLAN) ports.
Prior to configuring VTP version 3, you must ensure that the spanning-tree extend system-id command has been enabled.
If there is insufficient DRAM available for use by VTP, the VTP mode changes to transparent.

Here is the example : iam running VTP Transparent Mode and version1



```
SW1#show vtp status
VTP Version capable             : 1 to 3
VTP version running             : 1
VTP Domain Name                 : BBLB
VTP Pruning Mode                : Disabled
VTP Traps Generation            : Disabled
Device ID                       : aabb.cc80.1000
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00

Feature VLAN:
--------------
VTP Operating Mode                : Transparent
Maximum VLANs supported locally   : 1005
Number of existing VLANs          : 34
Configuration Revision            : 0
MD5 digest                        : 0xBB 0xC2 0x47 0x0A 0xF2 0xDA 0x14 0x58
                                    0x55 0x56 0x36 0x6C 0x90 0x9B 0x29 0xDC
```

VTP Modes

There are 3 modes of VTP and these modes are:

VTP Server: With the configuration of VTP Server on the switch, we can create, delete and edit the VLANs with all features. VTP servers advertise their VLAN configuration to other network devices in the same VTP domain and synchronise their VLAN configuration with other network devices based on advertisements received over trunk links. VTP server is the default mode.

VTP Client: VTP Client just copy the way VTP server do in the VTP domain but VTP client can't create, delete and edit VLANs in the VTP domain.

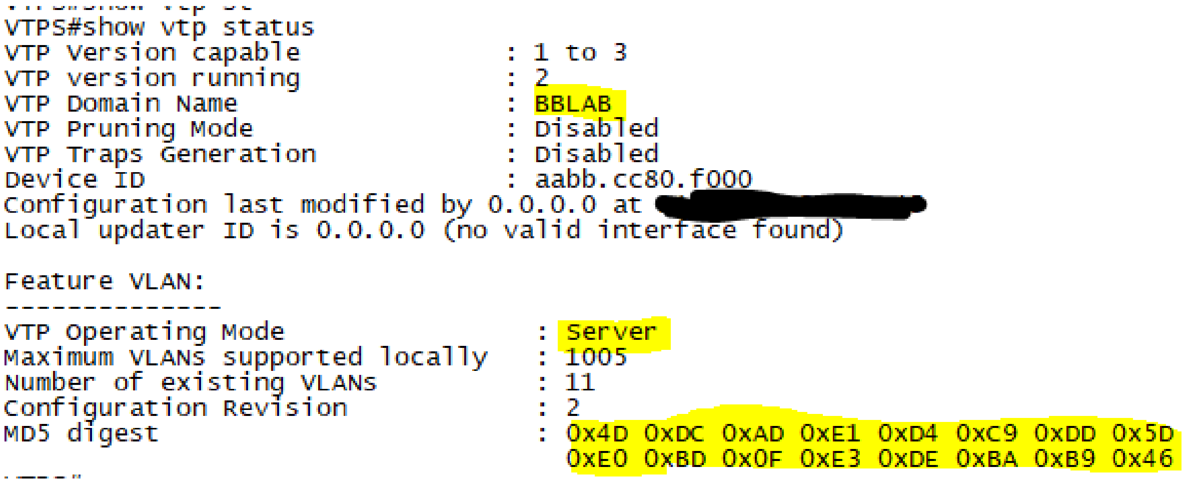
VTP transparent: VTP transparent network devices do not participate in VTP. A VTP transparent network device does not advertise its VLAN configuration and does not synchronize its VLAN configuration based on received advertisements. However, in VTP version 2, a transparent network device will forward receive VTP advertisements from its trunking LAN ports. In VTP version 3, a transparent network device is specific to an instance.

Configure SW22 Server and SW11 as client here:

Here is the summary:
Only VTP Server only can set version
Only VTP Server can only Prune the VLAN
Switch port should be DTP.

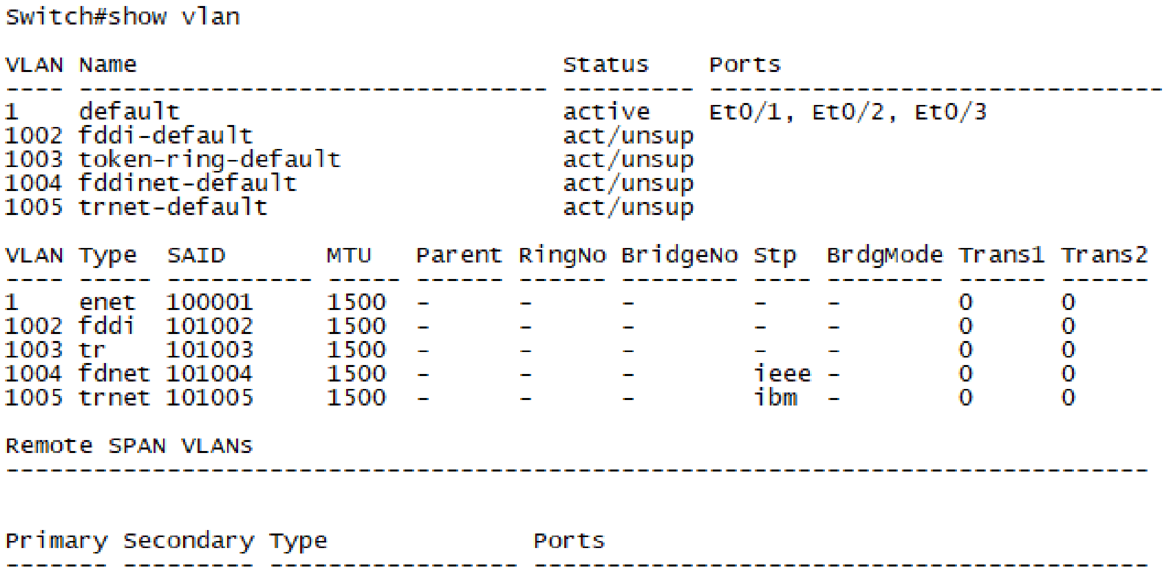
Here is the VTP Server config

```
Config t
Vtp version 2
Vtp domain BBLAB
Vtp mode server
```



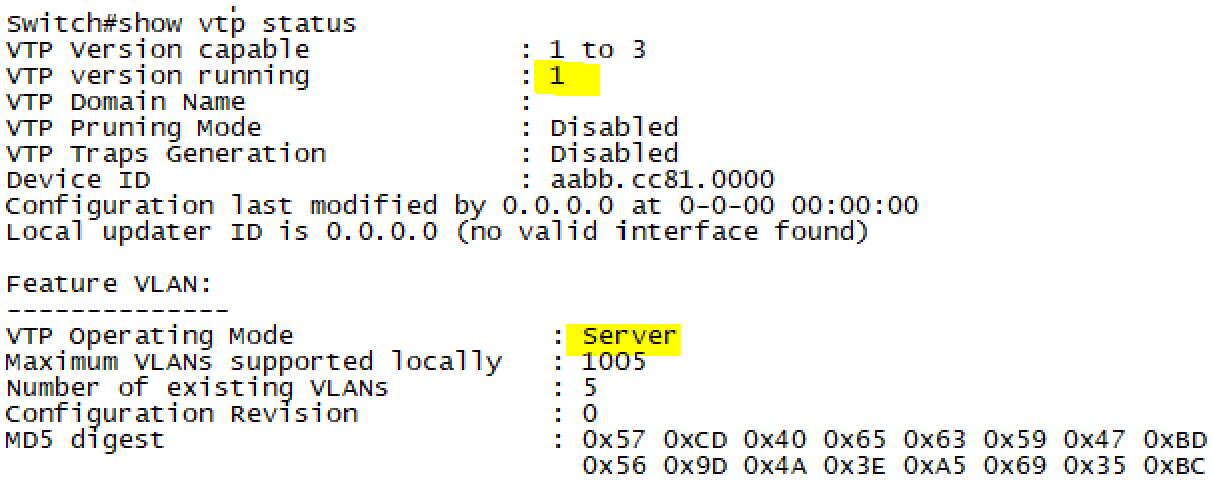
```
VTPS#show vtp status
VTP Version capable              : 1 to 3
VTP version running              : 2
VTP Domain Name                  : BBLAB
VTP Pruning Mode                 : Disabled
VTP Traps Generation             : Disabled
Device ID                        : aabb.cc80.f000
Configuration last modified by 0.0.0.0 at [redacted]
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
--------------
VTP Operating Mode                 : Server
Maximum VLANs supported locally    : 1005
Number of existing VLANs           : 11
Configuration Revision             : 2
MD5 digest                         : 0x4D 0xDC 0xAD 0xE1 0xD4 0xC9 0xDD 0x5D
                                     0xE0 0xBD 0x0F 0xE3 0xDE 0xBA 0xB9 0x46
```

Client Switch with not configuration output :



```
Switch#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Et0/1, Et0/2, Et0/3
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
1    enet  100001     1500  -      -      -        -    -        0      0
1002 fddi  101002     1500  -      -      -        -    -        0      0
1003 tr    101003     1500  -      -      -        -    -        0      0
1004 fdnet 101004     1500  -      -      -        ieee -        0      0
1005 trnet 101005     1500  -      -      -        ibm  -        0      0

Remote SPAN VLANs
------------------------------------------------------------------------------


Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------------
```

By default switch will be in Server mode and vtp version 1



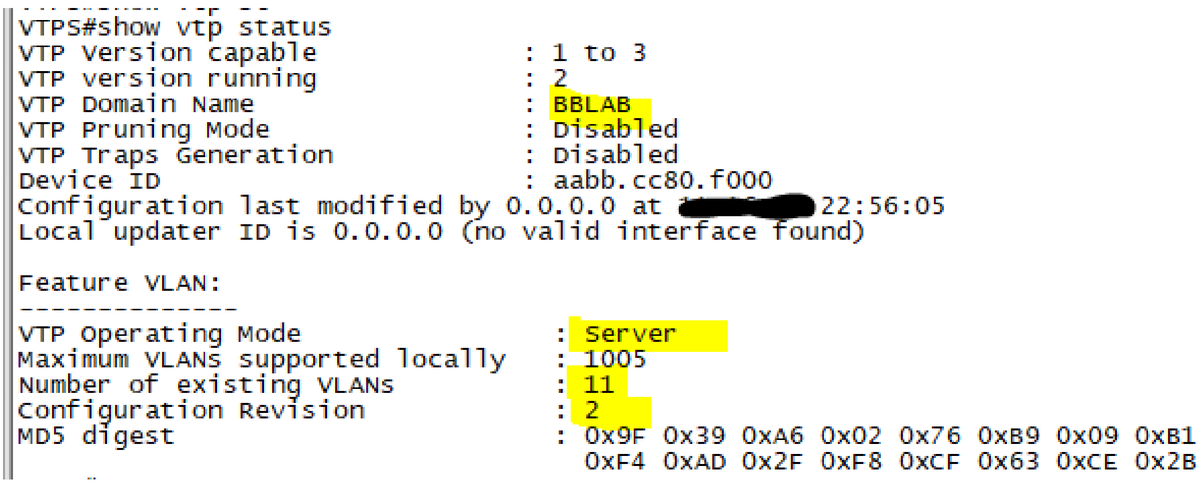
```
Switch#show vtp status
VTP Version capable               : 1 to 3
VTP version running               : 1
VTP Domain Name                   :
VTP Pruning Mode                  : Disabled
VTP Traps Generation              : Disabled
Device ID                         : aabb.cc81.0000
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
--------------
VTP Operating Mode                  : Server
Maximum VLANs supported locally     : 1005
Number of existing VLANs            : 5
Configuration Revision              : 0
MD5 digest                          : 0x57 0xCD 0x40 0x65 0x63 0x59 0x47 0xBD
                                      0x56 0x9D 0x4A 0x3E 0xA5 0x69 0x35 0xBC
```

Lets change to client mode as below configuration and take the updates from Server :

Config t
Vtp version 2
Vtp domain BBLAB
Vtp mode client

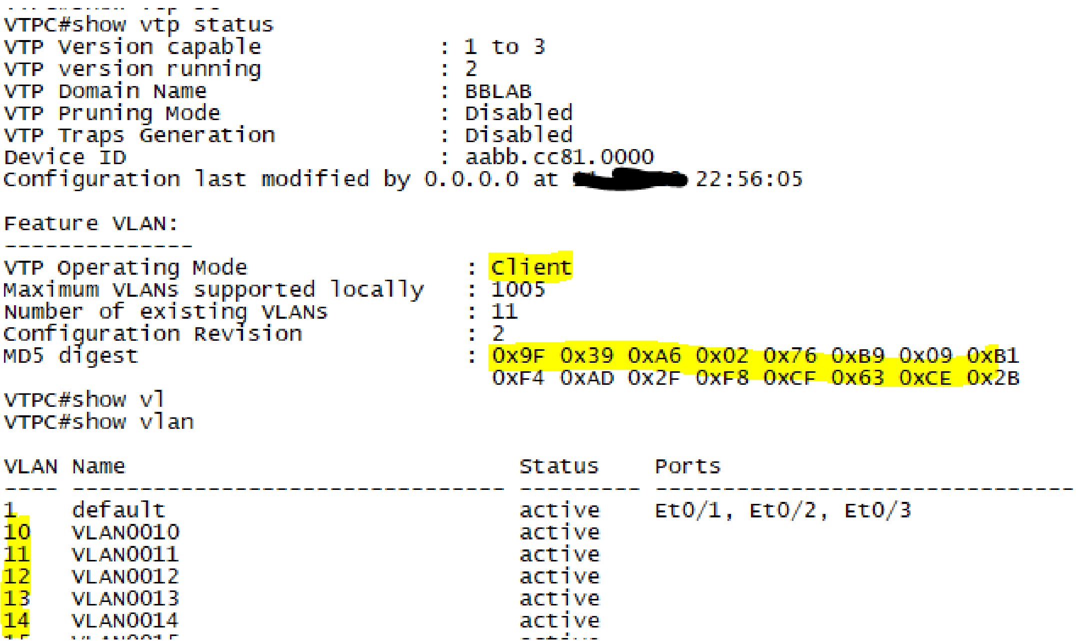
Lets add some vlan in VTP Server SW

VTPS#config t
Enter configuration commands, one per line. End with CNTL/Z.
VTPS(config)#vlan 10-15
VTPS(config-vlan)#end



```
VTPS#show vtp status
VTP Version capable               : 1 to 3
VTP version running               : 2
VTP Domain Name                   : BBLAB
VTP Pruning Mode                  : Disabled
VTP Traps Generation              : Disabled
Device ID                         : aabb.cc80.f000
Configuration last modified by 0.0.0.0 at [redacted] 22:56:05
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
--------------
VTP Operating Mode                  : Server
Maximum VLANs supported locally     : 1005
Number of existing VLANs            : 11
Configuration Revision              : 2
MD5 digest                          : 0x9F 0x39 0xA6 0x02 0x76 0xB9 0x09 0xB1
                                      0xF4 0xAD 0x2F 0xF8 0xCF 0x63 0xCE 0x2B
```

Client VLAN Updated.



```
VTPC#show vtp status
VTP Version capable               : 1 to 3
VTP version running               : 2
VTP Domain Name                   : BBLAB
VTP Pruning Mode                  : Disabled
VTP Traps Generation              : Disabled
Device ID                         : aabb.cc81.0000
Configuration last modified by 0.0.0.0 at [redacted] 22:56:05

Feature VLAN:
--------------
VTP Operating Mode                  : Client
Maximum VLANs supported locally     : 1005
Number of existing VLANs            : 11
Configuration Revision              : 2
MD5 digest                          : 0x9F 0x39 0xA6 0x02 0x76 0xB9 0x09 0xB1
                                      0xF4 0xAD 0x2F 0xF8 0xCF 0x63 0xCE 0x2B

VTPC#show vl
VTPC#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Et0/1, Et0/2, Et0/3
10   VLAN0010                         active
11   VLAN0011                         active
12   VLAN0012                         active
13   VLAN0013                         active
14   VLAN0014                         active
```

VTP Pruning

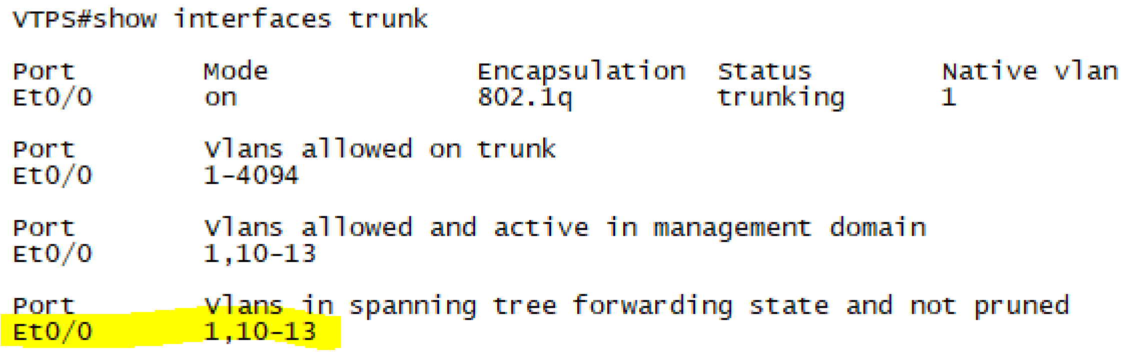
VTP pruning increases the available network bandwidth by restricting flooded traffic
– disabled by default
– blocked unneeded flooded traffic to VLANs on trunk ports
– only VLANs included in the pruning-eligible list can be pruned, 2 – 1001 by default
– VLANs 1002 – 1005 and extended-range VLANs are pruning-ineligible
– if a VLAN is configured as pruning-ineligible, flooded traffic continues
– supported in all VTP versions

Enabling VTP pruning on a VTP server enables pruning for the entire management domain

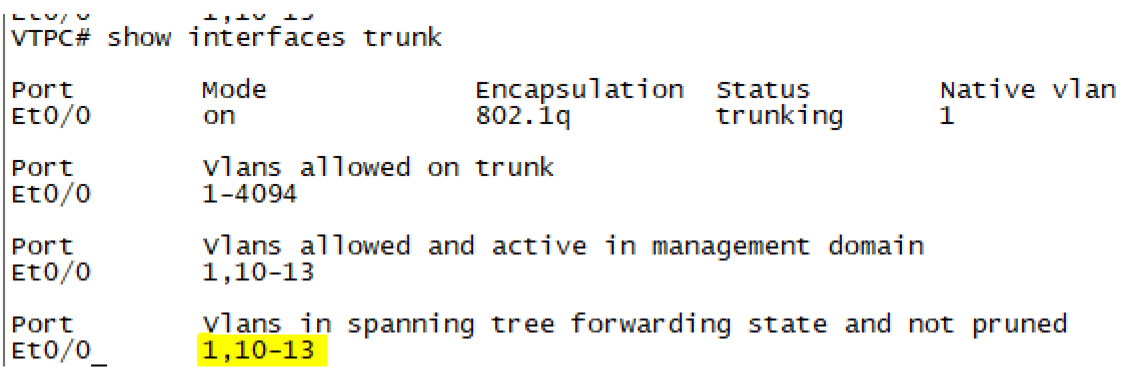
VTP pruning is not designed to function in VTP transparent mode

Define pruning eligible VLANs with the interface **switchport trunk pruning vlan** *number* command. You can verify pruning operation with the **show interface** *number* **pruning** command.

Let’s check without pruning Server and Client VTP SWITCH :



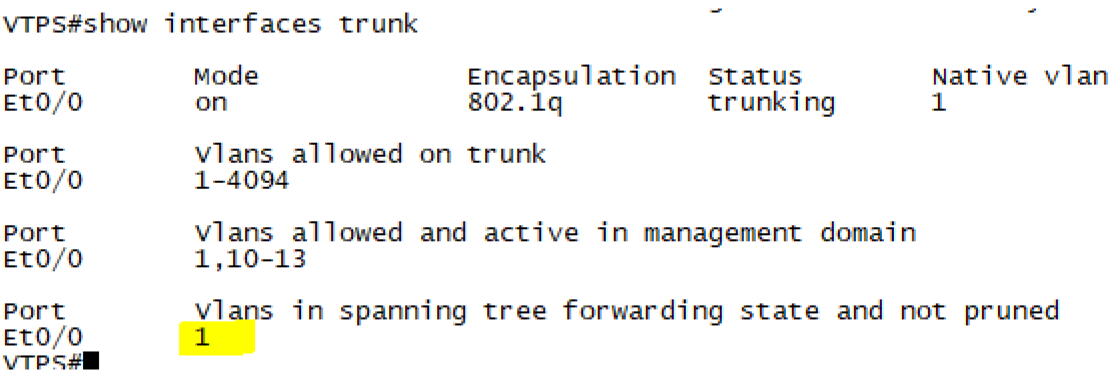
```
VTPS#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       1,10-13
```



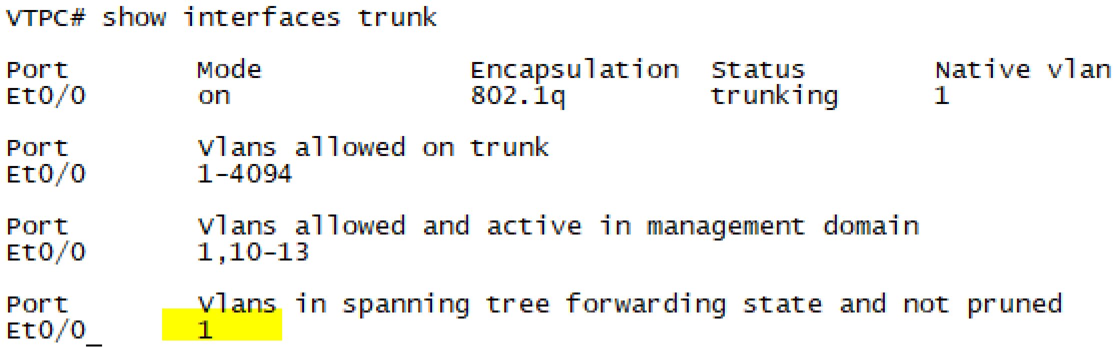
```
VTPC# show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0_      1,10-13
```

Lets Enable pruning on VTP Server side

```
Config t
!
Vtp pruning
!
End
```



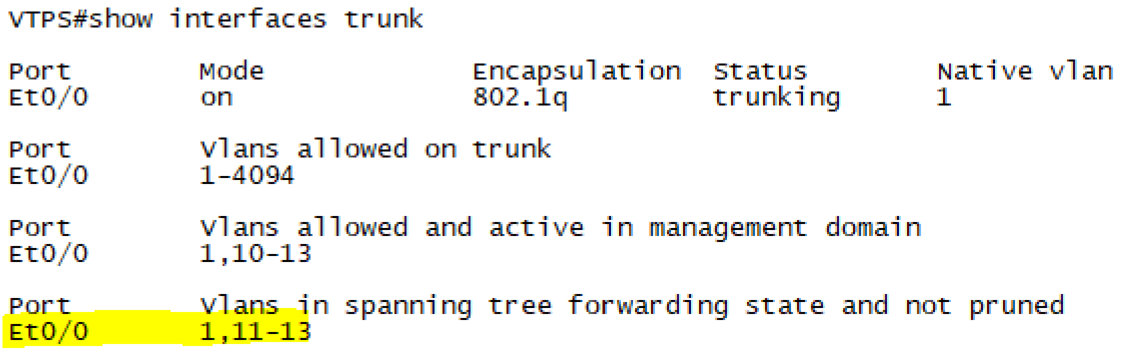
```
VTPS#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       1
VTPS#█
```



```
VTPC# show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0_      1
```

So now all the vlan 10-13 pruning. Let's only Vlan 10 for pruning.

```
VTPS#config t
Enter configuration commands, one per line.  End with CNTL/Z.
VTPS(config)#interface ethernet 0/0
VTPS(config-if)#switchport trunk pruning vlan 10
VTPS(config-if)#end
```



```
VTPS#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       1,11-13
```

1.1.d [ii] dot1Q

Most newer switches, we only have the option of 802.1q, but older switches also support ISL (Cisco Proprietary).

A trunk is a point-to-point link between one or more Ethernet switch interfaces and another network device such as a router or switch.

Trunkinng encapsulations:
– Inter-Switch Link (ISL) – Ciscro proprietary
– IEEE 802.1Q – industry standard

Trunk negotiation is managed by the Dynamic Trunking Protocol (DTP)
– some network devices might forward DTP frames improperly, which could cause some misconfigurations
– for interfaces connected to devices that do not support DTP, disable DTP

If you do not intend to trunk across a link, use:
– switchport mode access

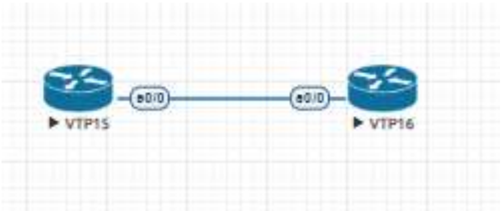
To enable trunking to a device that does not support DTP, use:
– switchport trunk encapsulation {isl|dot1q}
– switchport mode trunk
– switchport nonegotiate

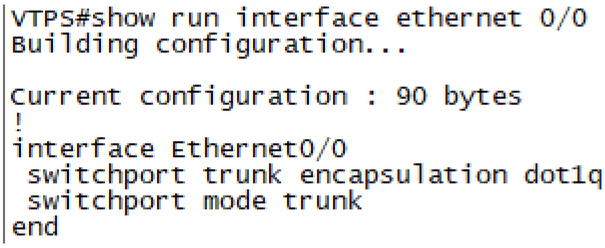
Configuring static trunks on those required the command
switchport trunk encapsulation dot1q

if you wanted ISL encap you would use
switchport trunk encapsulation ISL

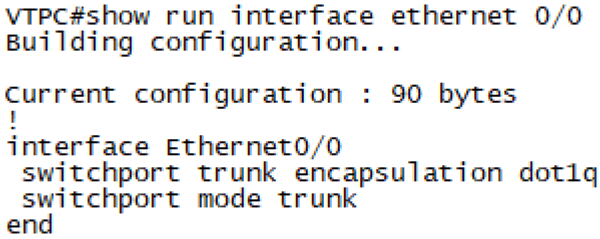
Trunk ports:
– cannot be a secure port
– cannot be a tunnel port
– for EtherChannel port groups, all interfaces must have the same configuration
– recommended that no more than 24 trunk ports in PVST mode
– recommended that no more than 40 trunk ports in MST mode
– switchport access vlan – specifies a default VLAN to be used if the interface stops trunking
– for 8021q, can received tagged and untagged (native VLAN) traffic

Let's configure between SW VTP15 and SW VTP16



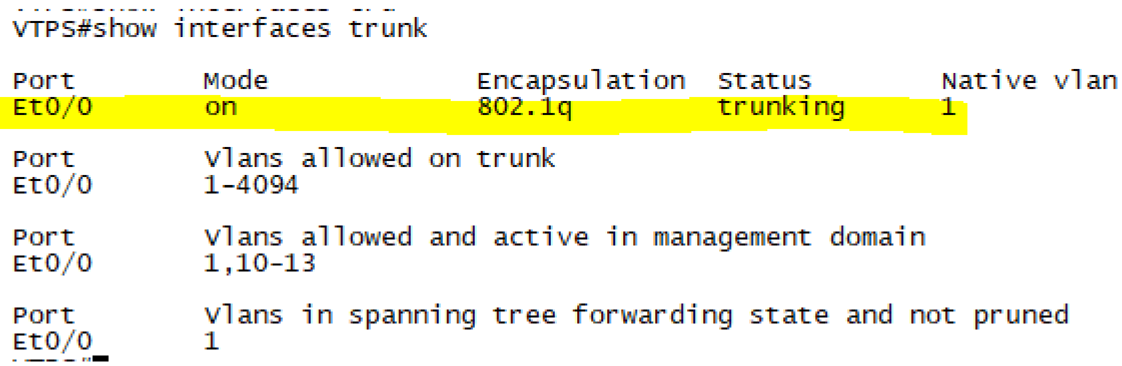


```
VTPS#show run interface ethernet 0/0
Building configuration...

Current configuration : 90 bytes
!
interface Ethernet0/0
 switchport trunk encapsulation dot1q
 switchport mode trunk
end
```

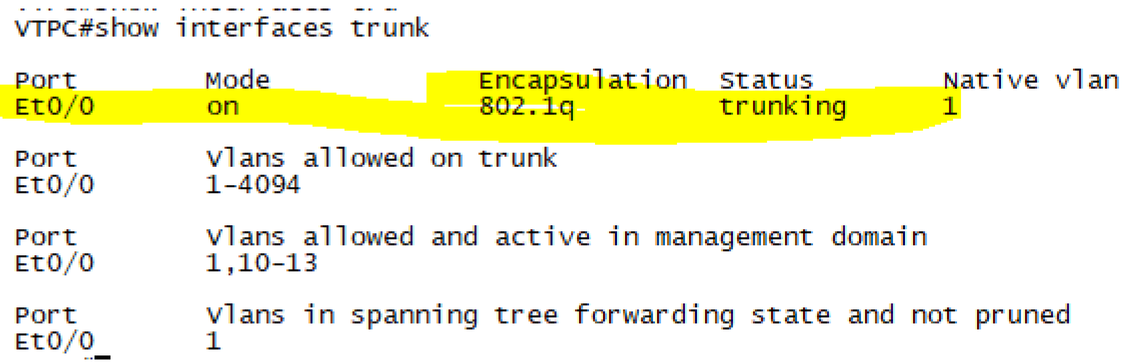


```
VTPC#show run interface ethernet 0/0
Building configuration...

Current configuration : 90 bytes
!
interface Ethernet0/0
 switchport trunk encapsulation dot1q
 switchport mode trunk
end
```

Verification:



```
VTPS#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       1
```



```
VTPC#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       1
```

1.1.d [iii] Native VLAN

The native vlan is the untagged vlan on the trunk. When the port is a trunk the native vlan by default is vlan 1, however we can change that with **switchport trunk native vlan x.**
The traffic for the native vlan goes out with NO 802.1q header, while tagged traffic on the trunk goes out with the 802.1q header to be placed in the proper vlan when it arrives.

In ISL we only had the concept of VLAN tagging, however 802.1q introduced the concept of the native vlan. That is a port could be configured with an untagged vlan, such that untagged traffic would be placed in this vlan.

CDP is able to detect and report native vlan mismatches due to it being sent out in the advertisement. So is PVST+.

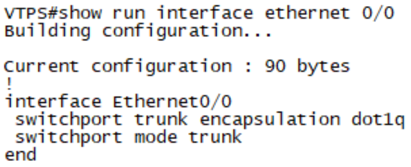
It is possible to make a Cisco switch TAG the native vlan on a trunk via the global config command **vlan dot1q tag native**.

If the Native VLAN for one end of a trunk link is different from the Native VLAN on the other end, spanning-tree loops might result

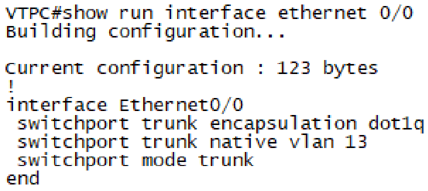
Disabling STP on the Native VLAN can potentially cause spanning-tree loops

By default, an interface on a switch is in Layer 2 mode

Native VLAN
– untagged traffic
– default VLAN 1
– the switch forwards all untagged traffic to the native VLAN
– can be assigned any VLAN ID

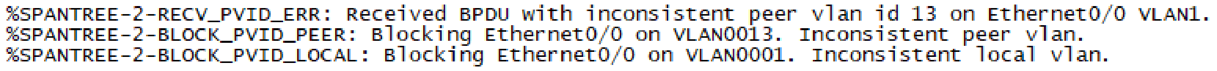


```
VTPS#show run interface ethernet 0/0
Building configuration...

Current configuration : 90 bytes
!
interface Ethernet0/0
 switchport trunk encapsulation dot1q
 switchport mode trunk
end
```

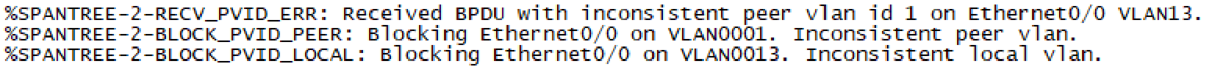


```
VTPC#show run interface ethernet 0/0
Building configuration...

Current configuration : 123 bytes
!
interface Ethernet0/0
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 13
 switchport mode trunk
end
```

As soon as you change in SW VTPC native vlan 13, you see below errors mismatch.

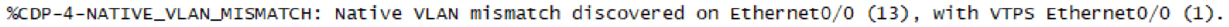


```
%SPANTREE-2-RECV_PVID_ERR: Received BPDU with inconsistent peer vlan id 13 on Ethernet0/0 VLAN1.
%SPANTREE-2-BLOCK_PVID_PEER: Blocking Ethernet0/0 on VLAN0013. Inconsistent peer vlan.
%SPANTREE-2-BLOCK_PVID_LOCAL: Blocking Ethernet0/0 on VLAN0001. Inconsistent local vlan.
```



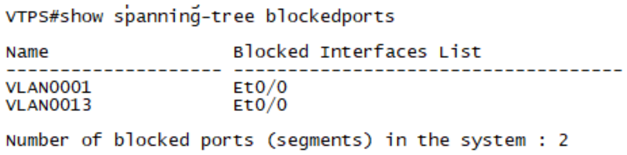
```
%SPANTREE-2-RECV_PVID_ERR: Received BPDU with inconsistent peer vlan id 1 on Ethernet0/0 VLAN13.
%SPANTREE-2-BLOCK_PVID_PEER: Blocking Ethernet0/0 on VLAN0001. Inconsistent peer vlan.
%SPANTREE-2-BLOCK_PVID_LOCAL: Blocking Ethernet0/0 on VLAN0013. Inconsistent local vlan.
```

You keep see this message until we add native vlan matches both the sides.

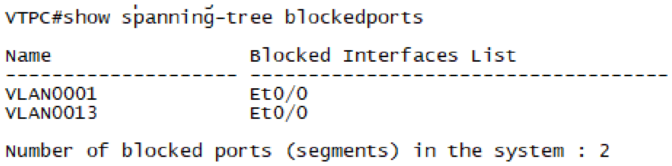


```
%CDP-4-NATIVE_VLAN_MISMATCH: Native VLAN mismatch discovered on Ethernet0/0 (13), with VTPS Ethernet0/0 (1).
```

This put the vlan in blocking mode in STP

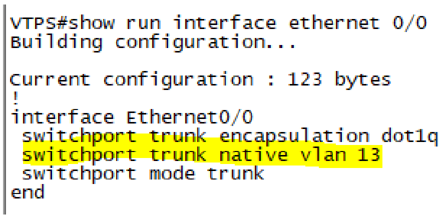


```
VTPS#show spanning-tree blockedports

Name                 Blocked Interfaces List
-------------------- ------------------------------------
VLAN0001             Et0/0
VLAN0013             Et0/0

Number of blocked ports (segments) in the system : 2
```



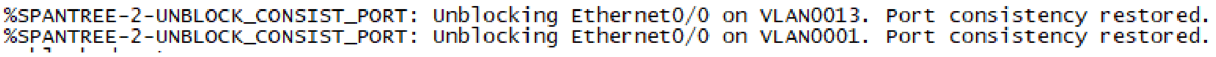
```
VTPC#show spanning-tree blockedports

Name                 Blocked Interfaces List
-------------------- ------------------------------------
VLAN0001             Et0/0
VLAN0013             Et0/0

Number of blocked ports (segments) in the system : 2
```

Lets fix this issue and check again

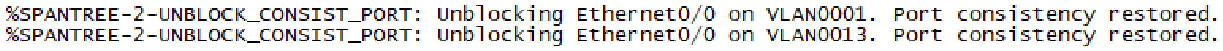


```
VTPS#show run interface ethernet 0/0
Building configuration...

Current configuration : 123 bytes
!
interface Ethernet0/0
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 13
 switchport mode trunk
end
```

As soon as you add above native vlan the trunk remove stp blocking for the vlan 13. And you will notice below logs in the both the switches.

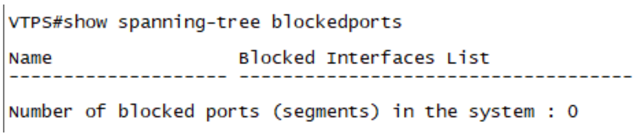


```
%SPANTREE-2-UNBLOCK_CONSIST_PORT: Unblocking Ethernet0/0 on VLAN0013. Port consistency restored.
%SPANTREE-2-UNBLOCK_CONSIST_PORT: Unblocking Ethernet0/0 on VLAN0001. Port consistency restored.
```



```
%SPANTREE-2-UNBLOCK_CONSIST_PORT: Unblocking Ethernet0/0 on VLAN0001. Port consistency restored.
%SPANTREE-2-UNBLOCK_CONSIST_PORT: Unblocking Ethernet0/0 on VLAN0013. Port consistency restored.
```

No more STP Block ports.

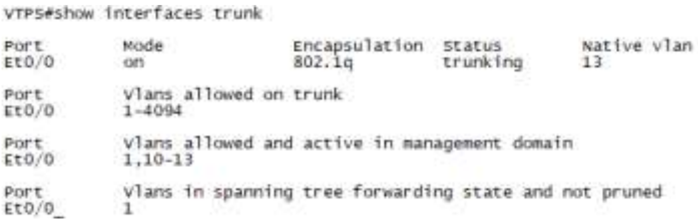


```
VTPS#show spanning-tree blockedports

Name                 Blocked Interfaces List
-------------------- ------------------------------------

Number of blocked ports (segments) in the system : 0
```

1.1.d [iv] Manual pruning

VTP is great, in concept, however in the real world I just don't see it deployed to often. The more likely situation is you will be tagging trunks manually, and pruning them manually.

Default all vlan allowed if you do not configure.

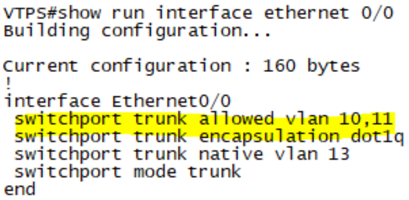


```
VTPS#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      13

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       1
```

The more "real-word" solution to pruning trunks is to just do manual pruning via "**switchport trunk allowed vlan x**"

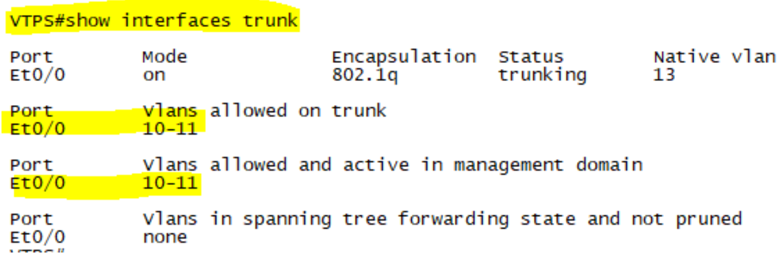
Lets only allow Vlan 10, 11 in the trunk

The command above overwrites the allowed vlan list on the interface, so to add or delete we could do the following
switchport trunk allowed vlan add x



```
VTPS#show run interface ethernet 0/0
Building configuration...

Current configuration : 160 bytes
!
interface Ethernet0/0
 switchport trunk allowed vlan 10,11
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 13
 switchport mode trunk
end
```

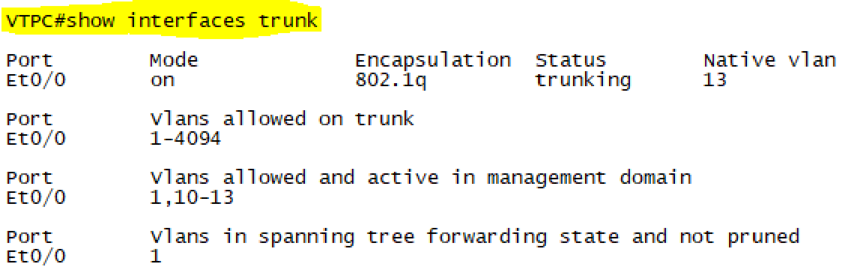
To verify which vlans are manually pruned from the trunk use
show interface trunk



```
VTPS#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      13

Port        Vlans allowed on trunk
Et0/0       10-11

Port        Vlans allowed and active in management domain
Et0/0       10-11

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       none
```

If you like to remove you can use below command to remove
switchport trunk allowed vlan remove x

Please note that if we only had the command
switchport mode trunk
It would mean ALL vlans are allowed on the trunk



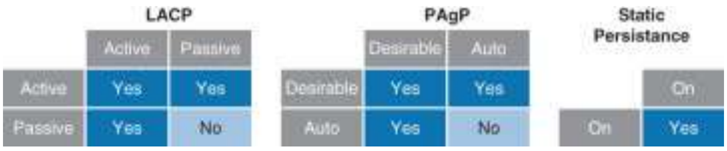
```
VTPC#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      13

Port        Vlans allowed on trunk
Et0/0       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,10-13

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       1
```

1.1.e Implement and troubleshoot etherchannel

1.1.e [i] LACP, PAgP, manual

EtherChannel provides fault-tolerant high-speed links between switches, routers, and servers.
– used to increase bandwidth
– provides automatic recovery from the loss of a link
– consists of individual Gigabit Ethernet links bundled into a single logical link
– provides full-duplex bandwidth up to 8Gb/s or 80Gb/s
– consist of up to eight compatibly configured Ethernet ports
– layer 2 or layer 3
– the number of EtherChannels is limited to 48

Layer 3 EtherChannels are not supported on switches running the LAN base feature set



LACP

	Active	Passive
Active	Yes	Yes
Passive	Yes	No

PAgP

	Desirable	Auto
Desirable	Yes	Yes
Auto	Yes	No

Static Persistance

	On
On	Yes

EtherChannel modes
– Link Aggregation Protocol (LACP)
– Port Aggregation Protocol (PAgP)
– On (Manual)

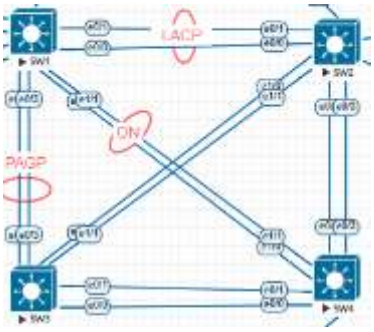
LACP
– defined by IEEE 802.3ad
– DTP and CDP send and receive packets over the physical ports in the EtherChannel
– trunk ports send and receive LACP packets on the lowest numbered VLAN
– for Layer 2, the first port in the channel provides the MAC address for the EtherChannel
–These LACP values are used when we have a hardware limitation like a max of 8 links, however we have more than that in the bundle (we can have stand by links)
– for Layer 3, the switch allocates a MAC address when the logical interface is created

PAgP is a Cisco proprietary protocol
– DTP and CDP send and receive packets over the physical ports in the EtherChannel
– for Layer 2, the first port in the channel provides the MAC address for the EtherChannel

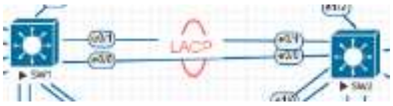
– for Layer 3, the switch allocates a MAC address when the logical interface is created
– trunk ports send and receive PAgP packets on the lowest numbered VLAN

EtherChannel On Mode (Manula)
– used to manually configure an EtherChannel
– forces a port to join an EtherChannel without negotiations
– can be useful if the remote device does not support PAgP or LACP
– all ports must be configured the same
– if the EtherChannel group is misconfigured, packet loss or spanning tree loops can occur

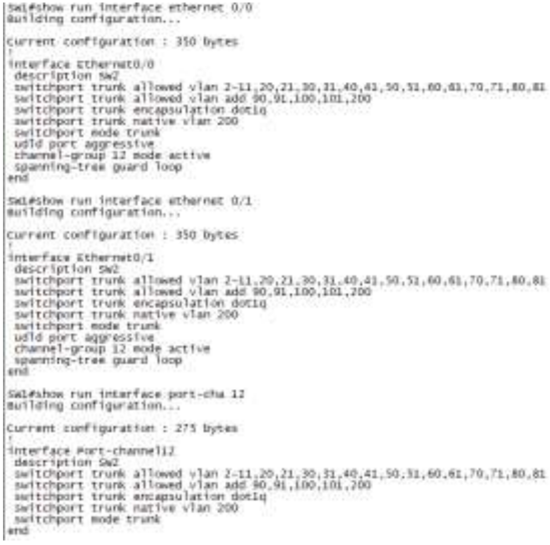
Below Topology we configure LACP, PAGP, ON



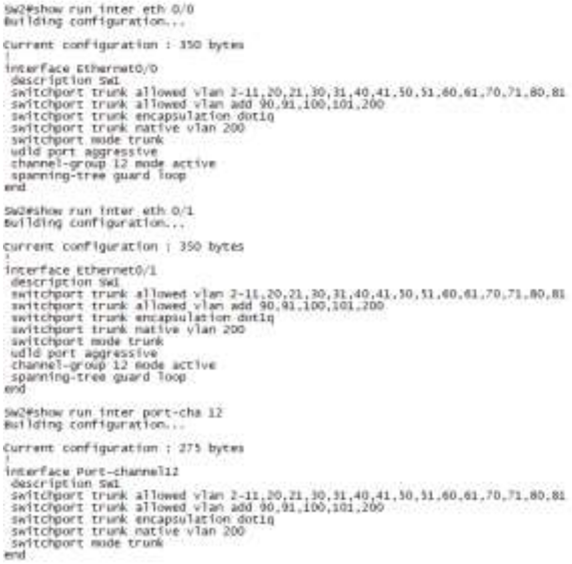
SW1 to SW2 – LACP



SW1 config as below :

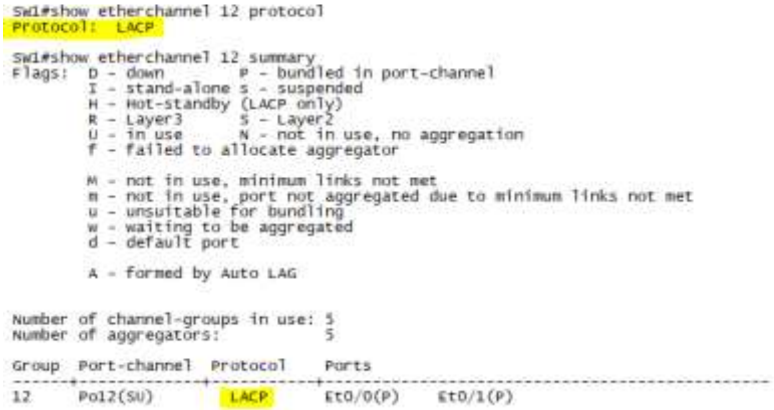


```
SW1#show run interface ethernet 0/0
Building configuration...

Current configuration : 350 bytes
!
interface Ethernet0/0
 description SW2
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 12 mode active
 spanning-tree guard loop
end

SW1#show run interface ethernet 0/1
Building configuration...

Current configuration : 350 bytes
!
interface Ethernet0/1
 description SW2
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 12 mode active
 spanning-tree guard loop
end

SW1#show run interface port-cha 12
Building configuration...

Current configuration : 275 bytes
!
interface Port-channel12
 description SW2
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
end
```

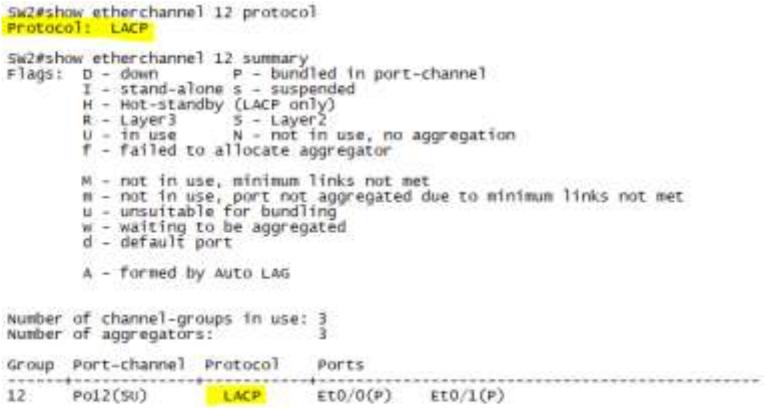


```
SW2#show run inter eth 0/0
Building configuration...

Current configuration : 350 bytes
!
interface Ethernet0/0
 description SW1
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 12 mode active
 spanning-tree guard loop
end

SW2#show run inter eth 0/1
Building configuration...

Current configuration : 350 bytes
!
interface Ethernet0/1
 description SW1
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 12 mode active
 spanning-tree guard loop
end

SW2#show run inter port-cha 12
Building configuration...

Current configuration : 275 bytes
!
interface Port-channel12
 description SW1
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
end
```

Verification:

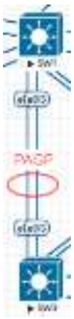


```
SW1#show etherchannel 12 protocol
Protocol:  LACP

SW1#show etherchannel 12 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 5
Number of aggregators:           5

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
12     Po12(SU)        LACP      Et0/0(P)    Et0/1(P)
```

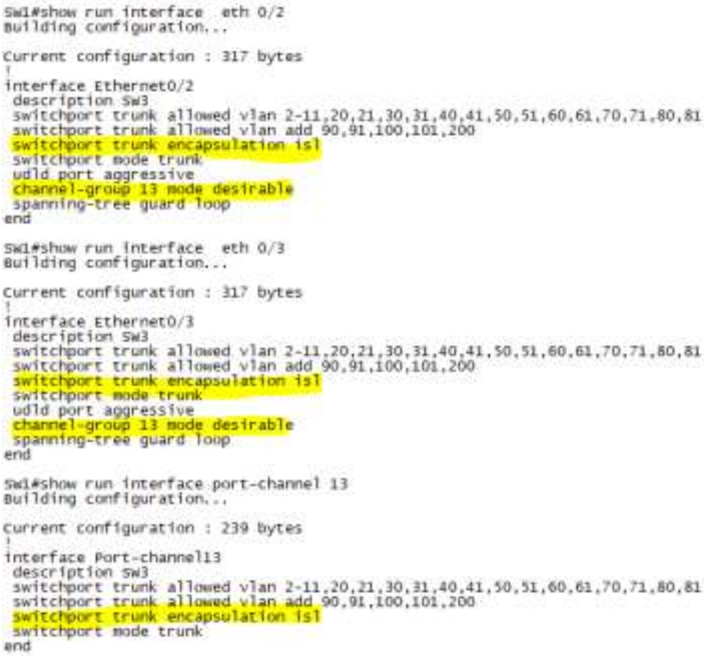


```
SW2#show etherchannel 12 protocol
Protocol:  LACP

SW2#show etherchannel 12 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 3
Number of aggregators:           3

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
12     Po12(SU)        LACP      Et0/0(P)    Et0/1(P)
```

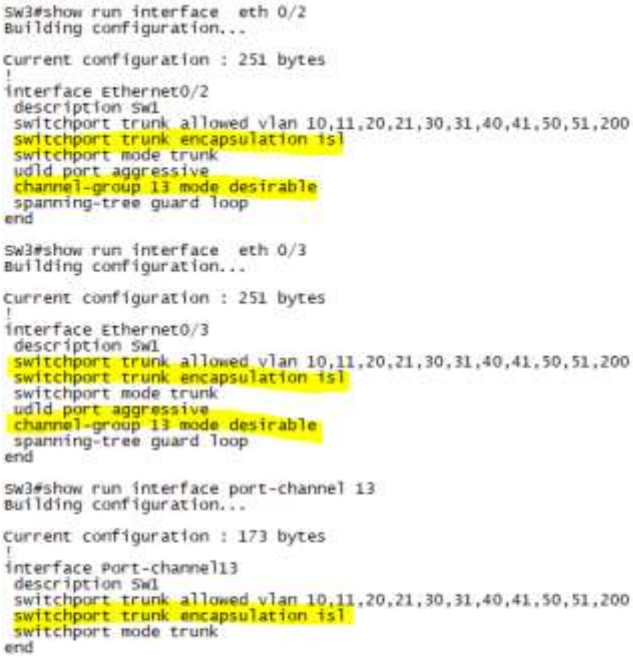
SW1 to SW3 channel group PAGP



SW1 and SW3 configuration:

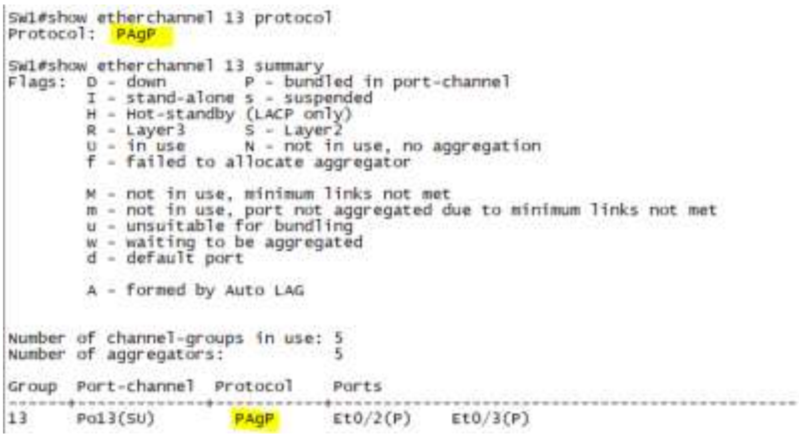


```
Sw1#show run interface  eth 0/2
Building configuration...

Current configuration : 317 bytes
!
interface Ethernet0/2
 description Sw3
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation isl
 switchport mode trunk
 udld port aggressive
 channel-group 13 mode desirable
 spanning-tree guard loop
end

Sw1#show run interface  eth 0/3
Building configuration...

Current configuration : 317 bytes
!
interface Ethernet0/3
 description Sw3
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation isl
 switchport mode trunk
 udld port aggressive
 channel-group 13 mode desirable
 spanning-tree guard loop
end

Sw1#show run interface port-channel 13
Building configuration...

Current configuration : 239 bytes
!
interface Port-channel13
 description Sw3
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation isl
 switchport mode trunk
end
```

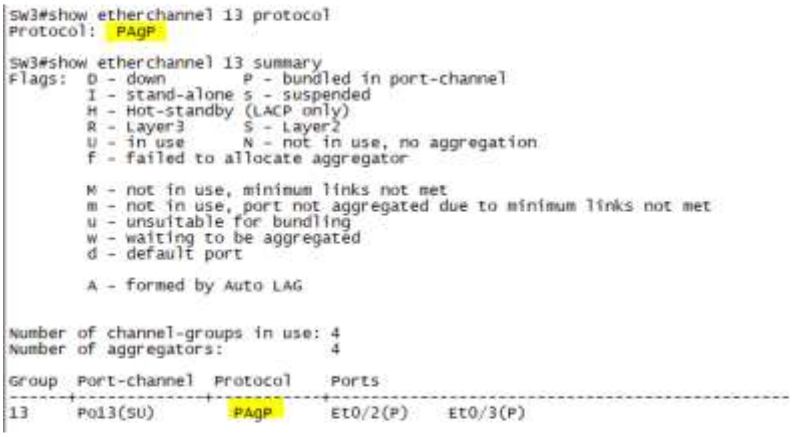


```
SW3#show run interface  eth 0/2
Building configuration...

Current configuration : 251 bytes
!
interface Ethernet0/2
 description SW1
 switchport trunk allowed vlan 10,11,20,21,30,31,40,41,50,51,200
 switchport trunk encapsulation isl
 switchport mode trunk
 udld port aggressive
 channel-group 13 mode desirable
 spanning-tree guard loop
end

SW3#show run interface  eth 0/3
Building configuration...

Current configuration : 251 bytes
!
interface Ethernet0/3
 description SW1
 switchport trunk allowed vlan 10,11,20,21,30,31,40,41,50,51,200
 switchport trunk encapsulation isl
 switchport mode trunk
 udld port aggressive
 channel-group 13 mode desirable
 spanning-tree guard loop
end

SW3#show run interface port-channel 13
Building configuration...

Current configuration : 173 bytes
!
interface Port-channel13
 description SW1
 switchport trunk allowed vlan 10,11,20,21,30,31,40,41,50,51,200
 switchport trunk encapsulation isl
 switchport mode trunk
end
```

Verify the Port-channel output

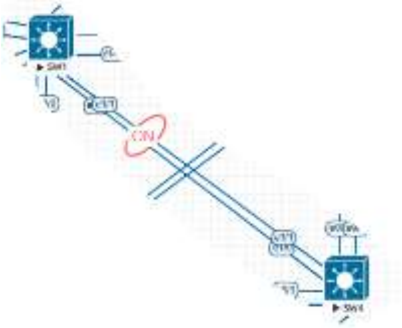


```
SW1#show etherchannel 13 protocol
Protocol:  PAgP

SW1#show etherchannel 13 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 5
Number of aggregators:           5

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
13     Po13(SU)         PAgP      Et0/2(P)     Et0/3(P)
```

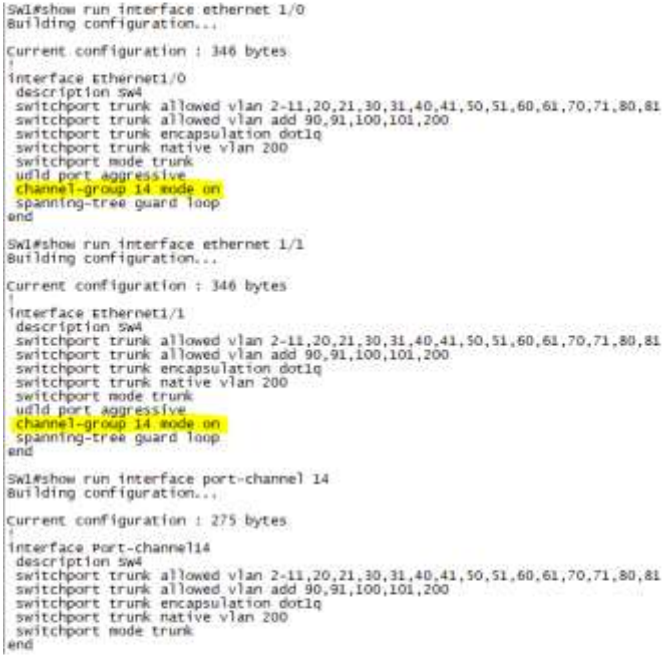


```
SW3#show etherchannel 13 protocol
Protocol:  PAgP

SW3#show etherchannel 13 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 4
Number of aggregators:           4

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
13     Po13(SU)         PAgP      Et0/2(P)     Et0/3(P)
```

SW1 to SW4 channel group ON

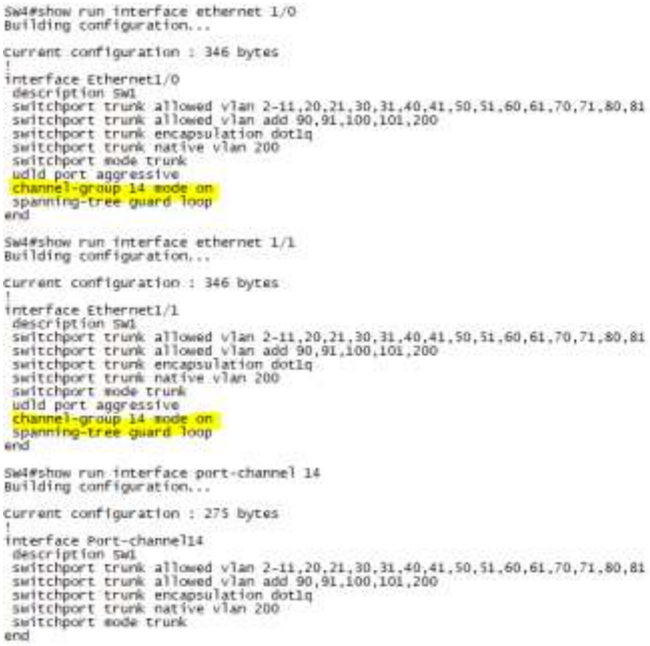


SW1 configuration :



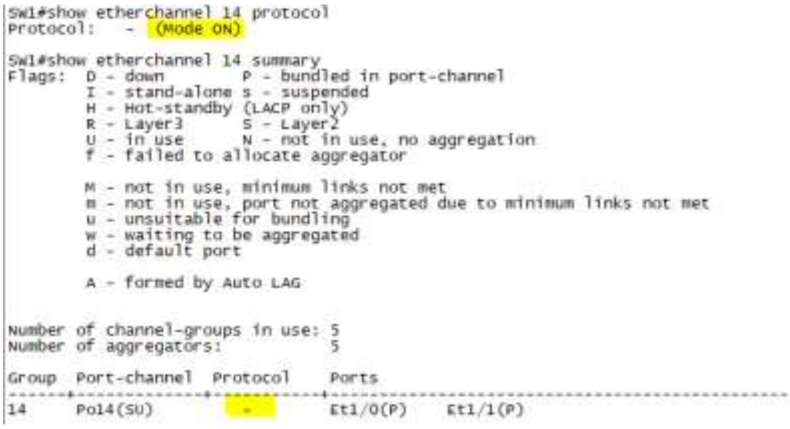
```
SW1#show run interface ethernet 1/0
Building configuration...

Current configuration : 346 bytes
!
interface Ethernet1/0
 description SW4
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 14 mode on
 spanning-tree guard loop
end

SW1#show run interface ethernet 1/1
Building configuration...

Current configuration : 346 bytes
!
interface Ethernet1/1
 description SW4
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 14 mode on
 spanning-tree guard loop
end

SW1#show run interface port-channel 14
Building configuration...

Current configuration : 275 bytes
!
interface Port-channel14
 description SW4
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
end
```

SW4 configuration:



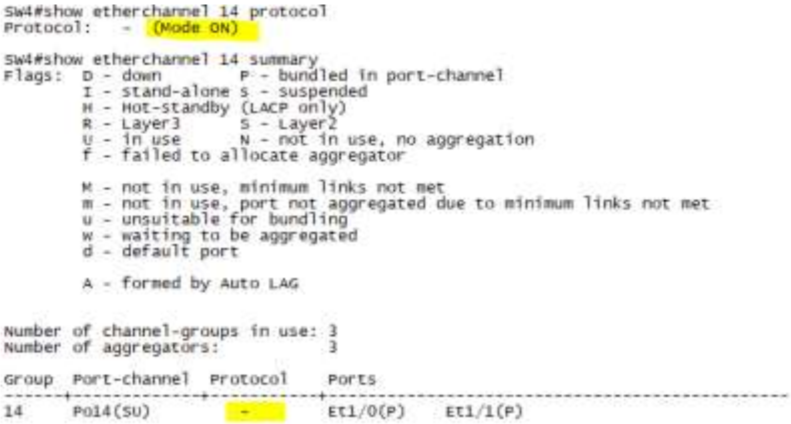
```
SW4#show run interface ethernet 1/0
Building configuration...

Current configuration : 346 bytes
!
interface Ethernet1/0
 description SW1
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 14 mode on
 spanning-tree guard loop
end

SW4#show run interface ethernet 1/1
Building configuration...

Current configuration : 346 bytes
!
interface Ethernet1/1
 description SW1
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 14 mode on
 spanning-tree guard loop
end

SW4#show run interface port-channel 14
Building configuration...

Current configuration : 275 bytes
!
interface Port-channel14
 description SW1
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
end
```

Verifications SW1



```
SW1#show etherchannel 14 protocol
Protocol:   -  (Mode ON)

SW1#show etherchannel 14 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 5
Number of aggregators:           5

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
14     Po14(SU)         -        Et1/0(P)    Et1/1(P)
```

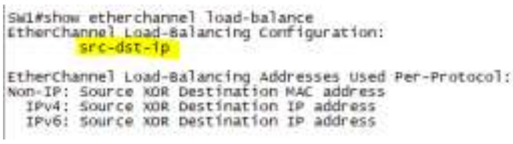
Verifications SW4



```
SW4#show etherchannel 14 protocol
Protocol:   -  (Mode ON)

SW4#show etherchannel 14 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 3
Number of aggregators:           3

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
14     Po14(SU)         -        Et1/0(P)    Et1/1(P)
```

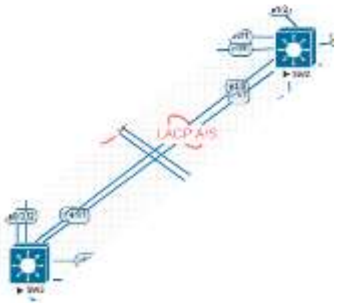
Other Port-channel Verification for Diagnosis Purpose:

show logging – always check the logs, if you have any issues
show etherchannel load-balance
show pagp internal
show pagp counters
show pagp neighbor
show lacp sys-id
show lacp counters
show lacp internal
show lacp neighbor
show etherchannel detail
show etherchannel summary

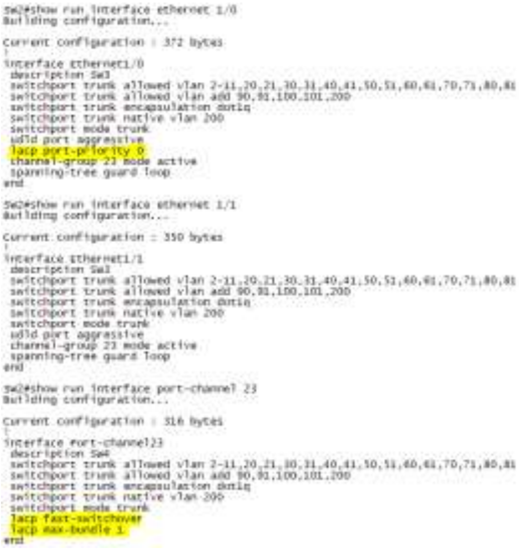
Load balancing methods	Default :
– dst-ip – dst-mac – src-dst-ip – src-dst-mac – src-ip – src-mac	 SW1#show etherchannel load-balance EtherChannel Load-Balancing Configuration: src-dst-ip EtherChannel Load-Balancing Addresses Used Per-Protocol: Non-IP: Source XOR Destination MAC address IPv4: Source XOR Destination IP address IPv6: Source XOR Destination IP address

LACP Active / Standby

Basically LACP fast-switchover is utilizing LACP for an active and standby link (by default it's active active).
Only time you would want to do this is if somehow the load balancing was not favoring the application correctly, or you were having issues with the traffic/flows being hashed.
This gives you the redundancy of LACP (without utilizing the redundancy of STP) in an active standby pair. This fast-switchover commands basically let's us recover an active/ standby link faster than just having the max-bundle 1 command, if we just had max-bundle in the config, the port-channel would flap and take longer to recover.

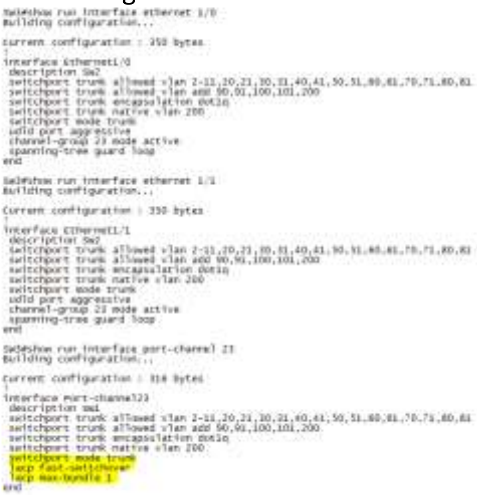


SW2 config



```
SW2#show run interface ethernet 1/0
Building configuration...

Current configuration : 372 bytes
!
interface Ethernet1/0
 description Sw3
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 lacp port-priority 0
 channel-group 23 mode active
 spanning-tree guard loop
end

SW2#show run interface ethernet 1/1
Building configuration...

Current configuration : 350 bytes
!
interface Ethernet1/1
 description Sw3
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 23 mode active
 spanning-tree guard loop
end

SW2#show run interface port-channel 23
Building configuration...

Current configuration : 316 bytes
!
interface Port-channel23
 description Sw4
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 lacp fast-switchover
 lacp max-bundle 1
end
```

SW3 config

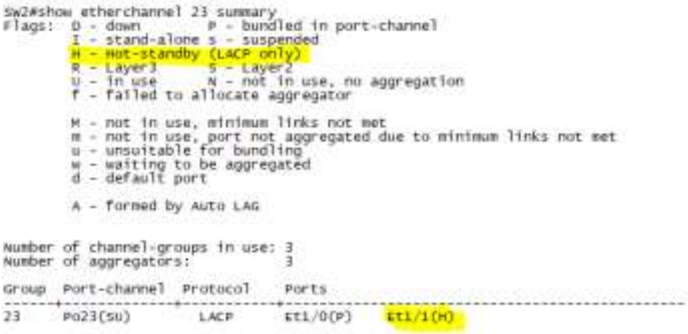


```
SW3#show run interface ethernet 1/0
Building configuration...

Current configuration : 350 bytes
!
interface Ethernet1/0
 description Sw2
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 23 mode active
 spanning-tree guard loop
end

SW3#show run interface ethernet 1/1
Building configuration...

Current configuration : 350 bytes
!
interface Ethernet1/1
 description Sw2
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 udld port aggressive
 channel-group 23 mode active
 spanning-tree guard loop
end

SW3#show run interface port-channel 23
Building configuration...

Current configuration : 316 bytes
!
interface Port-channel23
 description Sw1
 switchport trunk allowed vlan 2-11,20,21,30,31,40,41,50,51,60,61,70,71,80,81
 switchport trunk allowed vlan add 90,91,100,101,200
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 200
 switchport mode trunk
 lacp fast-switchover
 lacp max-bundle 1
end
```

By configuring interface config - **lacp port-priority 0**

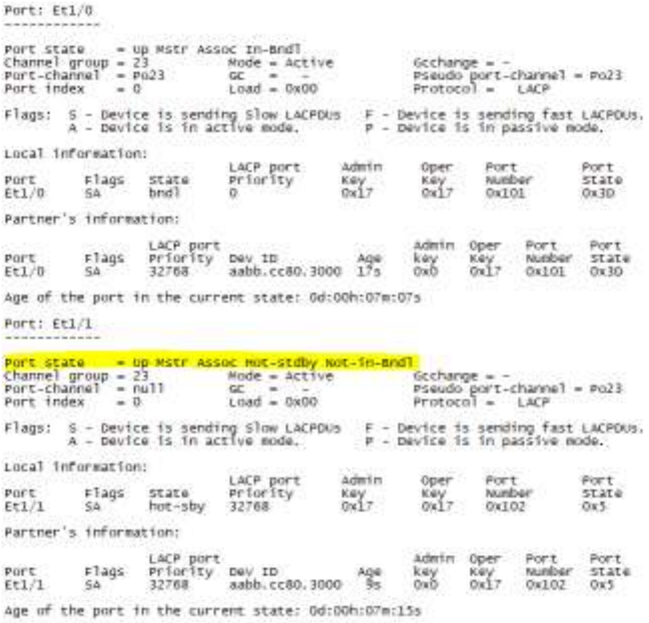
Now we configured SW2 to be the LACP master, and it will now choose which port becomes standby.

Let’s do some verifications.



```
SW2#show etherchannel 23 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 3
Number of aggregators:           3

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
23     Po23(SU)        LACP      Et1/0(P)    Et1/1(H)
```

SW2#show etherchannel 23 detail



```
Port: Et1/0
------------

Port state    = Up Mstr Assoc In-Bndl
Channel group = 23          Mode = Active          Gcchange = -
Port-channel  = Po23        GC   =   -             Pseudo port-channel = Po23
Port index    = 0           Load = 0x00            Protocol =   LACP

Flags:  S - Device is sending Slow LACPDUs   F - Device is sending fast LACPDUs.
        A - Device is in active mode.        P - Device is in passive mode.

Local information:
                            LACP port     Admin     Oper    Port        Port
Port      Flags   State     Priority      Key       Key     Number      State
Et1/0     SA      bndl      0             0x17      0x17    0x101       0x3D

Partner's information:

                  LACP port                        Admin  Oper   Port    Port
Port      Flags   Priority  Dev ID          Age    key    Key    Number  State
Et1/0     SA      32768     aabb.cc80.3000  17s    0x0    0x17   0x101   0x3D

Age of the port in the current state: 0d:00h:07m:07s

Port: Et1/1
------------

Port state    = Up Mstr Assoc Hot-stdby Not-in-Bndl
Channel group = 23          Mode = Active          Gcchange = -
Port-channel  = null        GC   =   -             Pseudo port-channel = Po23
Port index    = 0           Load = 0x00            Protocol =   LACP

Flags:  S - Device is sending Slow LACPDUs   F - Device is sending fast LACPDUs.
        A - Device is in active mode.        P - Device is in passive mode.

Local information:
                            LACP port     Admin     Oper    Port        Port
Port      Flags   State     Priority      Key       Key     Number      State
Et1/1     SA      hot-sby   32768         0x17      0x17    0x102       0x5

Partner's information:

                  LACP port                        Admin  Oper   Port    Port
Port      Flags   Priority  Dev ID          Age    key    Key    Number  State
Et1/1     SA      32768     aabb.cc80.3000   9s    0x0    0x17   0x102   0x5

Age of the port in the current state: 0d:00h:07m:15s
```

1.1.e [ii] layer 2, layer 3

Layer 2 EtherChannel

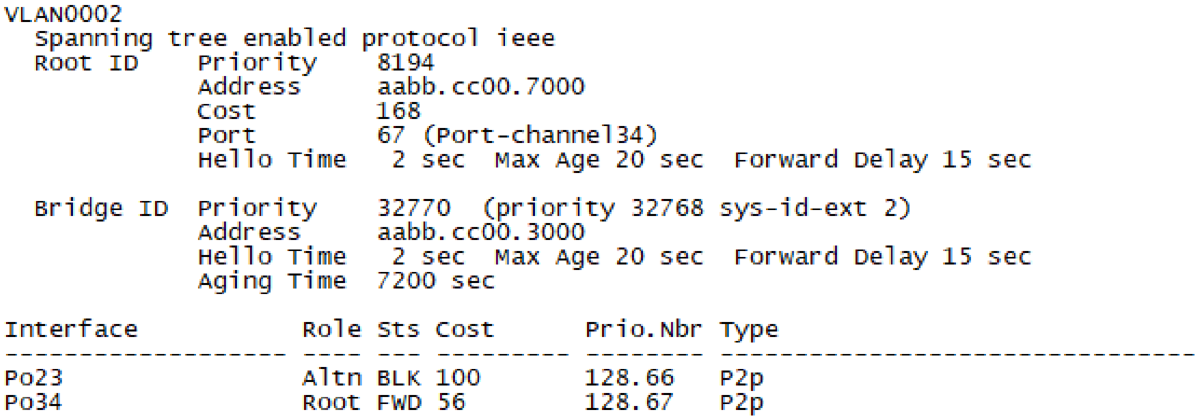
To create an L2 etherchannel, simply go to your physical interface(s) and use the **channel-group X mode** *mode* command.

It is best to configure physical interfaces while shut down and to bring up the virtual and physical interfaces all together.

After bringing up an L2 EtherChannel, you should verify its operation with traditional spanning-tree verification commands. Instead of seeing physical links, you should see your EtherChannel in the Spanning-tree show commands.

Example :

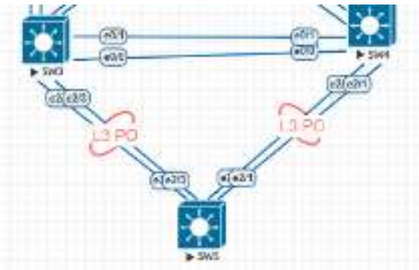
SW3#show spanning-tree



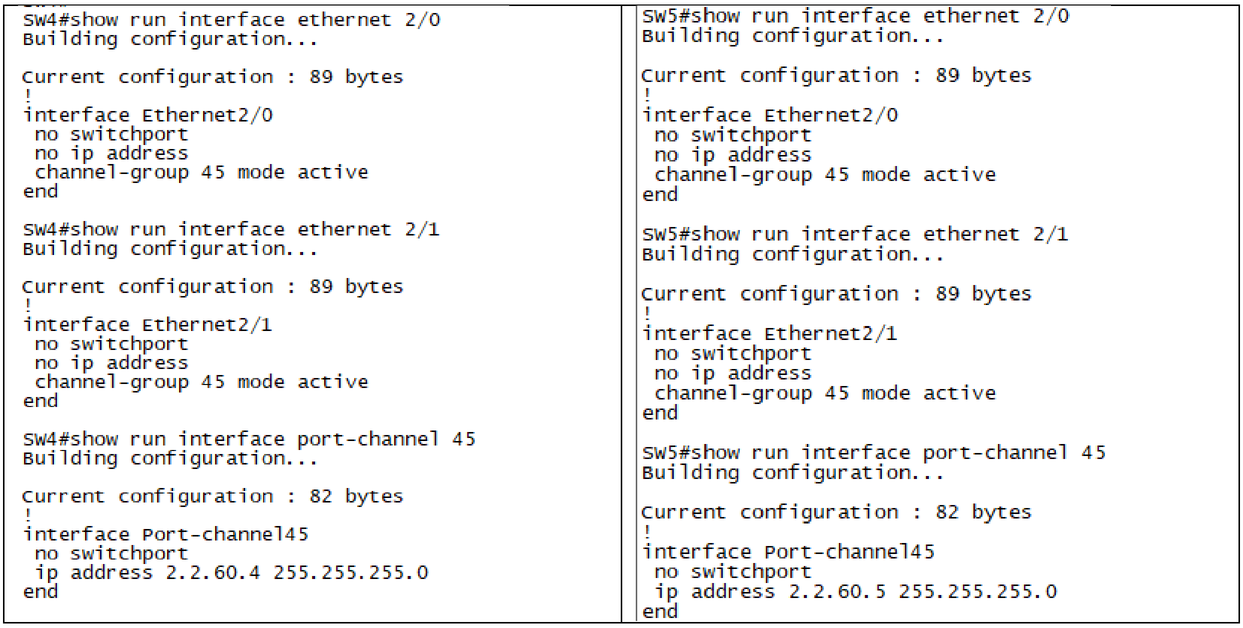
```
VLAN0002
  Spanning tree enabled protocol ieee
  Root ID    Priority    8194
             Address     aabb.cc00.7000
             Cost        168
             Port        67 (Port-channel34)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32770  (priority 32768 sys-id-ext 2)
             Address     aabb.cc00.3000
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  7200 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Po23                Altn BLK 100       128.66   P2p
Po34                Root FWD 56        128.67   P2p
```

Layer 3 EtherChannel:

-combines multiple logical interfaces such as subinterfaces.
-L3 Port-channel order of operation is important
- configure no switch port
- then configure channel group.

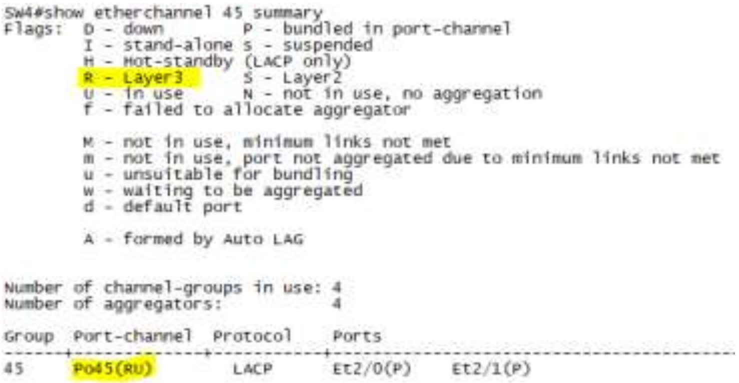


Let’s configure SW4 and SW5 L3 Port-channel

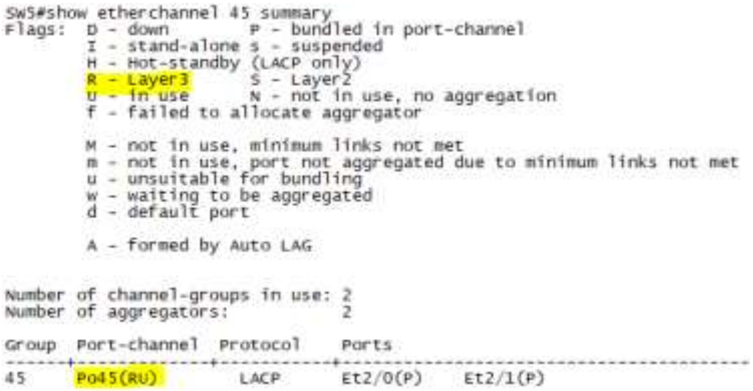


```
SW4#show run interface ethernet 2/0
Building configuration...

Current configuration : 89 bytes
!
interface Ethernet2/0
 no switchport
 no ip address
 channel-group 45 mode active
end

SW4#show run interface ethernet 2/1
Building configuration...

Current configuration : 89 bytes
!
interface Ethernet2/1
 no switchport
 no ip address
 channel-group 45 mode active
end

SW4#show run interface port-channel 45
Building configuration...

Current configuration : 82 bytes
!
interface Port-channel45
 no switchport
 ip address 2.2.60.4 255.255.255.0
end
```

```
SW5#show run interface ethernet 2/0
Building configuration...

Current configuration : 89 bytes
!
interface Ethernet2/0
 no switchport
 no ip address
 channel-group 45 mode active
end

SW5#show run interface ethernet 2/1
Building configuration...

Current configuration : 89 bytes
!
interface Ethernet2/1
 no switchport
 no ip address
 channel-group 45 mode active
end

SW5#show run interface port-channel 45
Building configuration...

Current configuration : 82 bytes
!
interface Port-channel45
 no switchport
 ip address 2.2.60.5 255.255.255.0
end
```

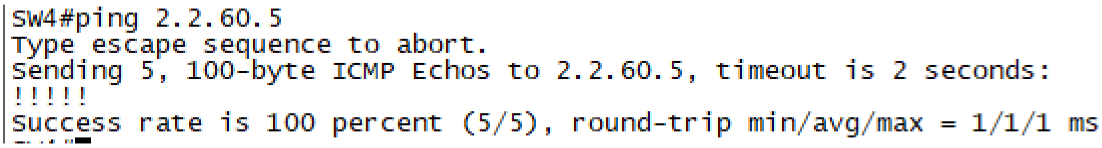


```
SW4#show etherchannel 45 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 4
Number of aggregators:           4

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
45     Po45(RU)         LACP      Et2/0(P)    Et2/1(P)
```

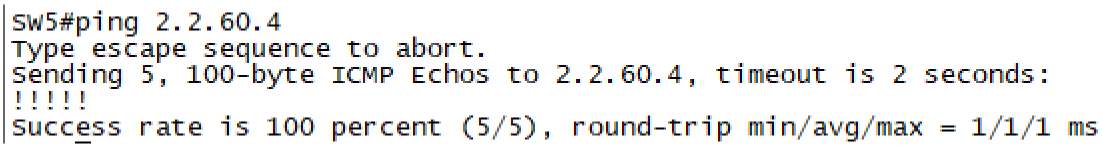


```
SW5#show etherchannel 45 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      N - not in use, no aggregation
        f - failed to allocate aggregator

        M - not in use, minimum links not met
        m - not in use, port not aggregated due to minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 2
Number of aggregators:           2

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
45     Po45(RU)         LACP      Et2/0(P)    Et2/1(P)
```

Ping Test :



```
SW4#ping 2.2.60.5
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.60.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```



```
SW5#ping 2.2.60.4
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.60.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

1.1.e [iii] load-balancing

Flows can become polarized (like CEF polarization) to a specific link if their flow is not classified as diverse. So depending on the etherchannel load balancing hashing algorithm, we might get polarization. By default, Cisco devices just hash the flow based on the source mac address, which is susceptible to polarization, thus this should be changed to something like src+dst ip and src + dst ports.

Note: changing this value is only locally significant and does not need to match. In fact, some people choose to have it not match when they don't have any good load balancing options.

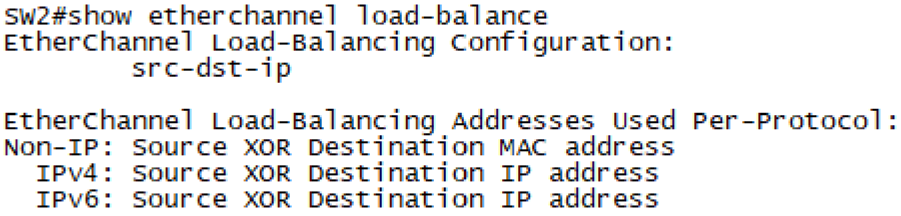
The LB method is based on a flow-by-flow behavior.

Example: If one of your port channels is configure as an access PC towards a server. The upstream switch receiving the frames from the server will see only one MAC address – the one assigned to the NIC of the Server.
The Switch will then say – for all frames coming from the server (for Example) use the first link in the port-channel to forward. The result is the first link will constantly have high-utilization.

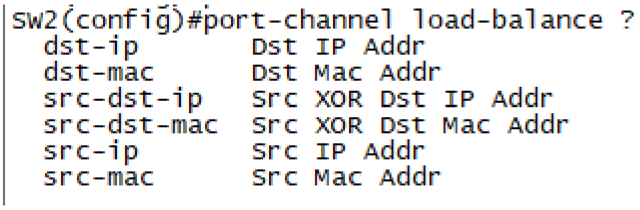
Here’s how we can change that algorithm.config:

port-channel load-balance <method>
port-channel load-balance src-ip
port-channel load-balance dst-ip
port-channel load-balance src-mac
port-channel load-balance dst-mac
port-channel load-balance src-dst-ip
port-channel load-balance src-dst-mac
verify: show etherchannel load-balance
We can test our load balancing further via a test command:
test etherchannel load-balance int port-channl <num> <ip or mac>

basically CEF handles L3 load balancing, and a different hashing bucket handles the L2 load balancing



```
SW2#show etherchannel load-balance
EtherChannel Load-Balancing Configuration:
        src-dst-ip

EtherChannel Load-Balancing Addresses Used Per-Protocol:
Non-IP: Source XOR Destination MAC address
  IPv4: Source XOR Destination IP address
  IPv6: Source XOR Destination IP address
```



```
SW2(config)#port-channel load-balance ?
  dst-ip       Dst IP Addr
  dst-mac      Dst Mac Addr
  src-dst-ip   Src XOR Dst IP Addr
  src-dst-mac  Src XOR Dst Mac Addr
  src-ip       Src IP Addr
  src-mac      Src Mac Addr
```

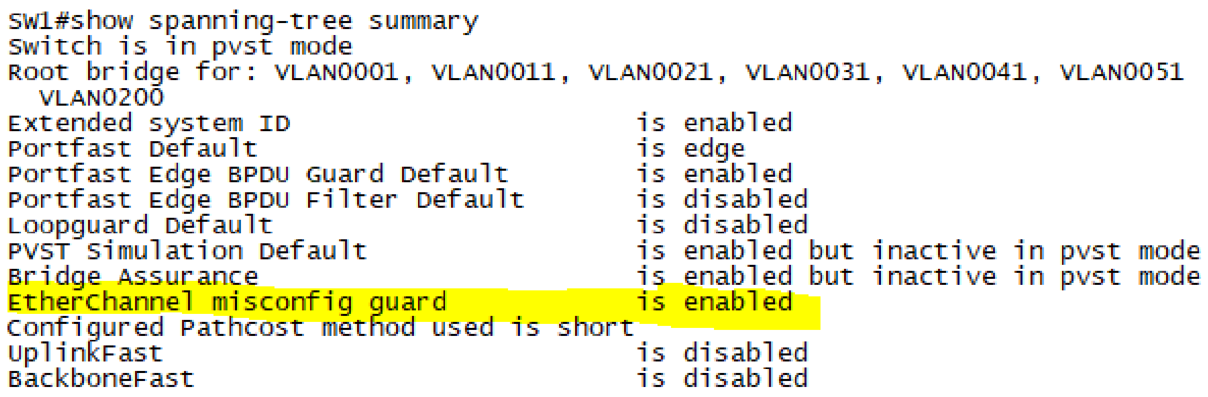
1.1.e [iv] etherchannel misconfiguration guard

EtherChannel Guard

You can use EtherChannel guard to detect an EtherChannel misconfiguration between the switch and a connected device.
A misconfiguration can occur if the switch interfaces are configured in an EtherChannel, but the interfaces on the other device are not.
A misconfiguration can also occur if the channel parameters are not the same at both ends of the EtherChannel.

If the switch detects a misconfiguration on the other device, EtherChannel guard places the switch interfaces in the error-disabled state, and displays an error message.

You can enable this feature by using the global configuration command:
spanning-tree etherchannel guard misconfig



```
SW1#show spanning-tree summary
Switch is in pvst mode
Root bridge for: VLAN0001, VLAN0011, VLAN0021, VLAN0031, VLAN0041, VLAN0051
  VLAN0200
Extended system ID                      is enabled
Portfast Default                        is edge
Portfast Edge BPDU Guard Default        is enabled
Portfast Edge BPDU Filter Default       is disabled
Loopguard Default                       is disabled
PVST Simulation Default                 is enabled but inactive in pvst mode
Bridge Assurance                        is enabled but inactive in pvst mode
EtherChannel misconfig guard            is enabled
Configured Pathcost method used is short
UplinkFast                              is disabled
BackboneFast                            is disabled
```

Use case:
You should use care when using the on mode. This is a manual configuration, and ports on both ends of the EtherChannel must have the same configuration. If the group is misconfigured, packet loss or spanning-tree loops can occur.

1.1.f Implement and troubleshoot spanning-tree

Spanning Tree Protocol

Spanning Tree Protocol (STP) is a Layer 2 link management protocol that provides path redundancy while preventing loops in the network. For a Layer 2 Ethernet network to function properly, only one active path can exist between any two stations. Multiple active paths among end stations cause loops in the network. If a loop exists in the network, end stations might receive duplicate messages. Switches might also learn end-station MAC addresses on multiple Layer 2 interfaces. These conditions result in an unstable network. Spanning-tree operation is transparent to end stations, which cannot detect whether they are connected to a single LAN segment or a switched LAN of multiple segments. The STP uses a spanning-tree algorithm to select one switch of a redundantly connected network as the root of the spanning tree. The algorithm calculates the best loop-free path through a switched Layer 2 network by assigning a role to each port based on the role of the port in the active topology

Root—A forwarding port elected for the spanning-tree topology
Designated—A forwarding port elected for every switched LAN segment
Alternate—A blocked port providing an alternate path to the root bridge in the spanningtree
Backup—A blocked port in a loopback configuration

1.1.f [i] PVST+/RPVST+/MST

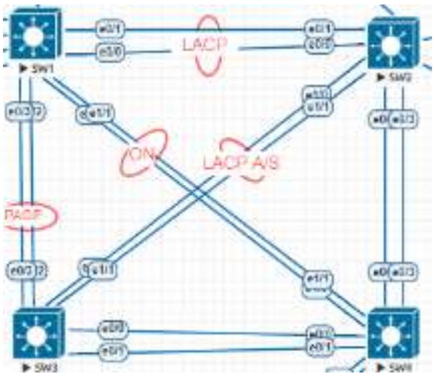
PVST+
– IEEE 802.1D
– runs a spanning-tree instance for each VLAN
– uses an aging-timer
– limited to 128 spanning-tree instances

rapid-PVST+
– IEEE 802.1w
– to provide rapid convergence, all dynamically learned MAC address entries are deleted
  when a topology change is received
– limited to 128 spanning-tree instances

MSTP

– IEEE 802.1s
– can map multiple VLANs to the same spanning-tree instance
– runs on top of rapid-PVST+
– limited to 65 MST instances
– the number of VLANs that can be mapped to an MST instance is unlimited

Here is the example configuration and topology :



SW1 will be root bridge for 11,21,31,41,51 with an priorty 24576

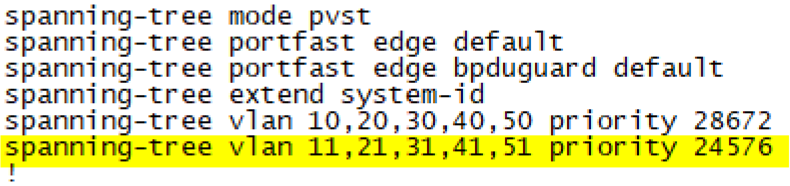
SW2 will be root bridge for 10,20,30,40,50 with an priorty 28672

SW1 will be root bridge for 61,71,81,91,101 with an priorty 24576

SW2 will be root bridge for 60,70,80,90,100 with an priorty 28672

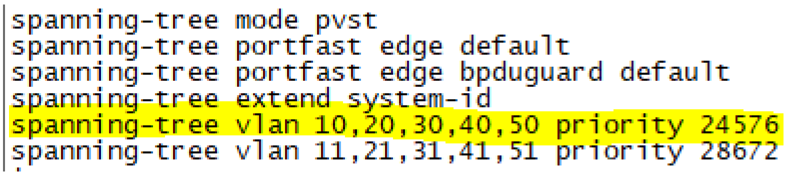
Configuration:

SW1



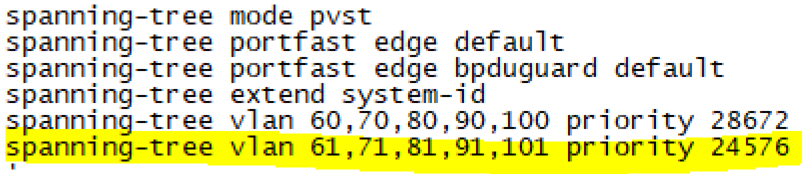
```
spanning-tree mode pvst
spanning-tree portfast edge default
spanning-tree portfast edge bpduguard default
spanning-tree extend system-id
spanning-tree vlan 10,20,30,40,50 priority 28672
spanning-tree vlan 11,21,31,41,51 priority 24576
!
```

SW2



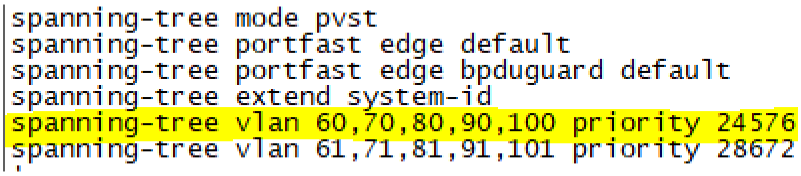
```
spanning-tree mode pvst
spanning-tree portfast edge default
spanning-tree portfast edge bpduguard default
spanning-tree extend system-id
spanning-tree vlan 10,20,30,40,50 priority 24576
spanning-tree vlan 11,21,31,41,51 priority 28672
```

SW3



```
spanning-tree mode pvst
spanning-tree portfast edge default
spanning-tree portfast edge bpduguard default
spanning-tree extend system-id
spanning-tree vlan 60,70,80,90,100 priority 28672
spanning-tree vlan 61,71,81,91,101 priority 24576
```

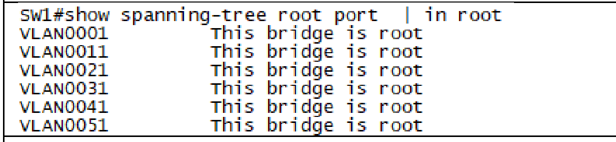
SW4



```
spanning-tree mode pvst
spanning-tree portfast edge default
spanning-tree portfast edge bpduguard default
spanning-tree extend system-id
spanning-tree vlan 60,70,80,90,100 priority 24576
spanning-tree vlan 61,71,81,91,101 priority 28672
```

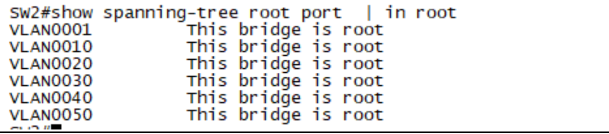
Verifications:

SW1



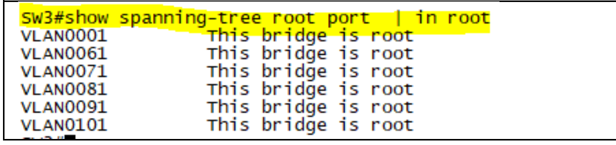
```
SW1#show spanning-tree root port   | in root
VLAN0001            This bridge is root
VLAN0011            This bridge is root
VLAN0021            This bridge is root
VLAN0031            This bridge is root
VLAN0041            This bridge is root
VLAN0051            This bridge is root
```

SW2



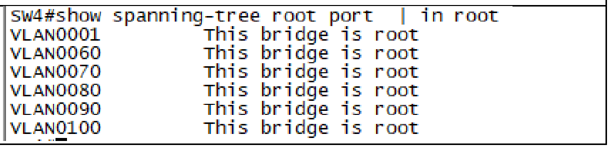
```
SW2#show spanning-tree root port   | in root
VLAN0001            This bridge is root
VLAN0010            This bridge is root
VLAN0020            This bridge is root
VLAN0030            This bridge is root
VLAN0040            This bridge is root
VLAN0050            This bridge is root
```

SW3



```
SW3#show spanning-tree root port   | in root
VLAN0001            This bridge is root
VLAN0061            This bridge is root
VLAN0071            This bridge is root
VLAN0081            This bridge is root
VLAN0091            This bridge is root
VLAN0101            This bridge is root
```

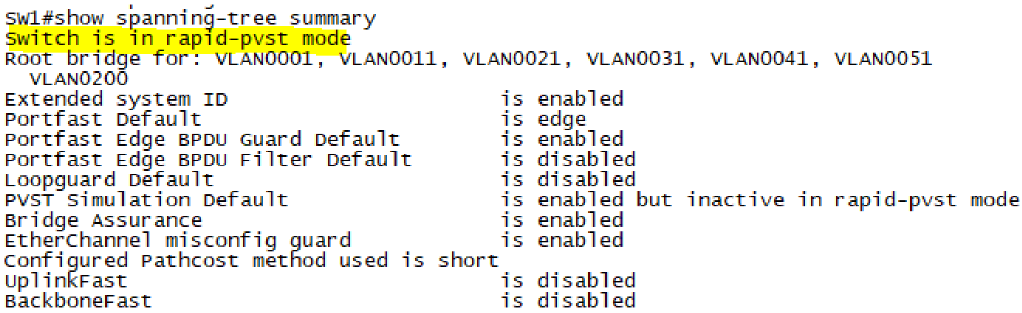
SW4



```
SW4#show spanning-tree root port   | in root
VLAN0001            This bridge is root
VLAN0060            This bridge is root
VLAN0070            This bridge is root
VLAN0080            This bridge is root
VLAN0090            This bridge is root
VLAN0100            This bridge is root
```

Now change spanning tree mode to rapid-pvst on SW1, SW2, SW3, SW4

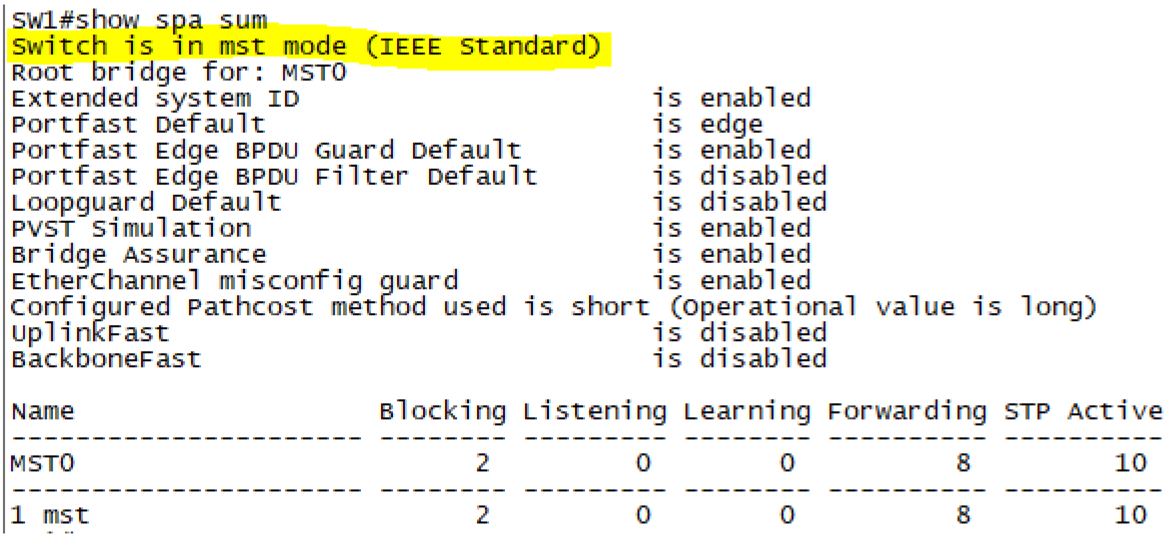
Output show in all switches same like SW1 (other switches will show vlan as per the configuration).



```
SW1#show spanning-tree summary
Switch is in rapid-pvst mode
Root bridge for: VLAN0001, VLAN0011, VLAN0021, VLAN0031, VLAN0041, VLAN0051
  VLAN0200
Extended system ID                      is enabled
Portfast Default                        is edge
Portfast Edge BPDU Guard Default        is enabled
Portfast Edge BPDU Filter Default       is disabled
Loopguard Default                       is disabled
PVST Simulation Default                 is enabled but inactive in rapid-pvst mode
Bridge Assurance                        is enabled
EtherChannel misconfig guard            is enabled
Configured Pathcost method used is short
UplinkFast                              is disabled
BackboneFast                            is disabled
```

Now change spanning tree mode to MST on SW1, SW2, SW3, SW4

Output show in all switches same like SW1 (other switches will show vlan as per the configuration).



```
SW1#show spa sum
Switch is in mst mode (IEEE Standard)
Root bridge for: MST0
Extended system ID                      is enabled
Portfast Default                        is edge
Portfast Edge BPDU Guard Default        is enabled
Portfast Edge BPDU Filter Default       is disabled
Loopguard Default                       is disabled
PVST Simulation                         is enabled
Bridge Assurance                        is enabled
EtherChannel misconfig guard            is enabled
Configured Pathcost method used is short (Operational value is long)
UplinkFast                              is disabled
BackboneFast                            is disabled

Name                      Blocking Listening Learning Forwarding STP Active
---------------------- -------- --------- -------- ---------- ----------
MST0                          2         0        0          8         10
---------------------- -------- --------- -------- ---------- ----------
1 mst                         2         0        0          8         10
```

1.1.f [ii] switch priority, port priority, path cost, STP timers

Port Priority

If a loop occurs, spanning tree uses the port priority when selecting an interface to put into the forwarding state. You can assign higher priority values (lower numerical values) to interfaces that you want to select first and lower priority values (higher numerical values) that you want to select last. If all interfaces have the same priority value, spanning tree puts the interface with the lowest interface number in the forwarding state and blocks the other interfaces.

The default switch priority is 32768

default port priority is 128

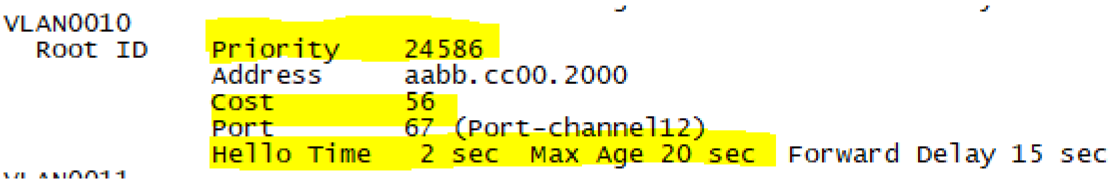
Cost Values:

Traffic to root will always follow the lowest cost path to reach root. The cost is a cumulative over all links to the root and based on link type. The links with the lowest cost will be elected the designated port. If there ends up a tie – then the switches will fall back to their own bridge ID

Bandwidth	OLD STP Value	New STP Value
10 MB	100	2,000,000
100MMbps	19	200,000
1GB	4	20,000
10GB	2	2,000
100GB	NA	2000
1Tbbs	NA	20
Port-channels		12 (depends on how many members are in channel (this example reflects 200Mb or 2 100MB links)

STP Timers:

Variable	Description
Hello timer	Controls how often the switch broadcasts hello messages to other switches.
Forward-delay timer	Controls how long each of the listening and learning states last before the interface begins forwarding.
Maximum-age timer	Controls the amount of time the switch stores protocol information received on an interface.
Transmit hold count	Controls the number of BPDUs that can be sent before pausing for 1 second.



```
VLAN0010
  Root ID    Priority    24586
             Address     aabb.cc00.2000
             Cost        56
             Port        67 (Port-channel12)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
VLAN0011
```

Default timers:
hello time 2,
max age time 20 (dead interval)
forward delay time 15 (how long to wait between listening and learning)
Changing these per switch has no meaning.
Only the root bridge can set these and propagate them down.

Changing timers:

conf t
spanning-tree vlan 1 hello-time 1
spanning-tree vlan 1 forward-time 8
spanning-tee vlan 1 max-age 10

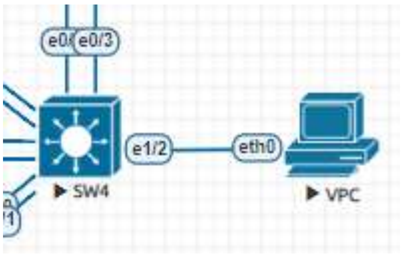
1.1.f [iii] port fast, BPDUguard, BPDUfilter

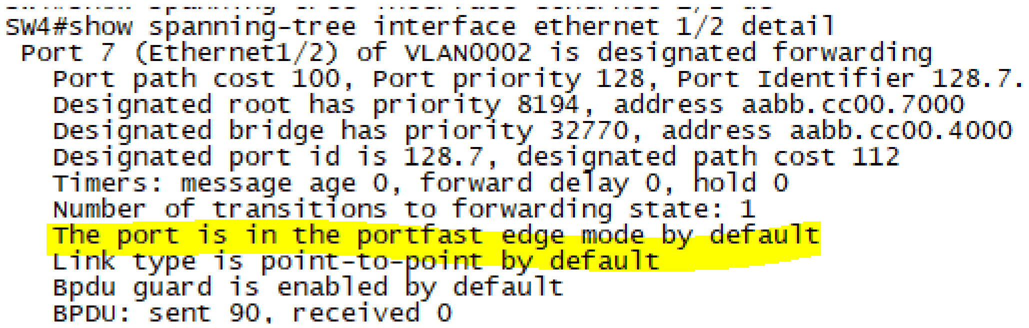
PortFast
With RSTP this is now an open standard and is called an EDGE port, however the configuration is still very similar to 802.1d portfast

Allows a port running STP to go from blocking to forwarding immediately. This skips listening and learning (15 + 15 sec)
Should only be enabled on access switches, edge ports. Requires extra parameters to be configured On a trunk port. Can still prevent loops.

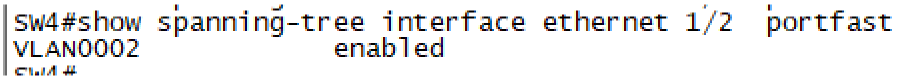
This also prevents topology change notifications from being sent when a port goes up or down.

Here is example topology: SW4 interface Ethernet 1 / 4 connected to PC with access port.





```
SW4#show spanning-tree interface ethernet 1/2 detail
 Port 7 (Ethernet1/2) of VLAN0002 is designated forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.7.
   Designated root has priority 8194, address aabb.cc00.7000
   Designated bridge has priority 32770, address aabb.cc00.4000
   Designated port id is 128.7, designated path cost 112
   Timers: message age 0, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   The port is in the portfast edge mode by default
   Link type is point-to-point by default
   Bpdu guard is enabled by default
   BPDU: sent 90, received 0
```



```
SW4#show spanning-tree interface ethernet 1/2  portfast
VLAN0002                enabled
SW4#
```

Note: Configure PortFast edge only on ports that connect to end stations; otherwise, an accidental topology loop could cause a data packet loop and disrupt switch and network operation.

BPDU Guard

The Bridge Protocol Data Unit (BPDU) guard feature can be globally enabled on the switch or can be enabled per port, but the feature operates with some differences.

When you enable BPDU guard at the global level on PortFast edge-enabled ports, spanning tree shuts down ports that are in a PortFast edge-operational state if any BPDU is received on them. In a valid configuration, PortFast edge-enabled ports do not receive BPDUs. Receiving a BPDU on a Port Fast edge-enabled port means an invalid configuration, such as the connection of an unauthorized device, and the BPDU guard feature puts the port in the error-disabled state. When this happens, the switch shuts down the entire port on which the violation occurred.

When you enable BPDU guard at the interface level on any port without also enabling the PortFast edge feature, and the port receives a BPDU, it is put in the error-disabled state.

The BPDU guard feature provides a secure response to invalid configurations because you must manually put the interface back in service. Use the BPDU guard feature in a service-provider network to prevent an access port from participating in the spanning tree.

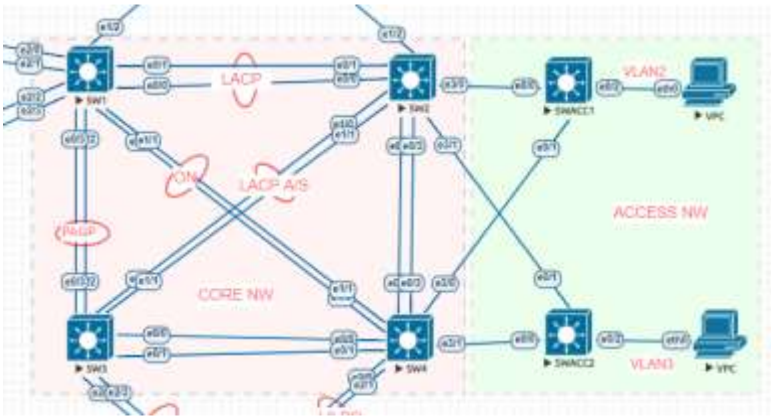
BPDU Filter

BPDU Filter prevents ANY BPDUs from leaving or being received on portfast enabled ports. Can be enabled globally, or per port.

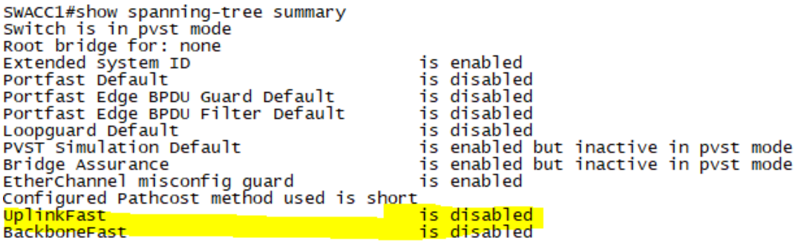
Configuration:
# spanning-tree portfast edge bpdufilter default
# spanning-tree bpdufilter enable

Note : Enabling BPDU filtering on an interface is the same as disabling spanning tree on it and can result in spanning-tree loops.

UplinkFast – Fast uplink recovery at the access layer where a switch has a backup/alternate root port in blocked state, enables port to immediately move to forwarding state. Alternate port cost with UplinkFast is very high, preventing access switch from becoming a transit point.



We use SWACC1 for testing, default spanning uplinkfast not enable, so it will take time for recovery as below tests.

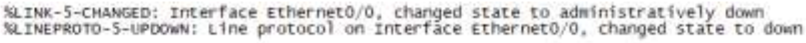


```
SWACC1#show spanning-tree summary
Switch is in pvst mode
Root bridge for: none
Extended system ID                       is enabled
Portfast Default                         is disabled
Portfast Edge BPDU Guard Default         is disabled
Portfast Edge BPDU Filter Default        is disabled
Loopguard Default                        is disabled
PVST Simulation Default                  is enabled but inactive in pvst mode
Bridge Assurance                         is enabled but inactive in pvst mode
EtherChannel misconfig guard             is enabled
Configured Pathcost method used is short
UplinkFast                               is disabled
BackboneFast                             is disabled
```

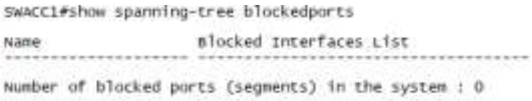
SWACC1 shutdown the Ether0/0 towards SW2 since Ether 0/1 in blocking mode.

```
SWACC1#show spanning-tree blockedports

Name                 Blocked Interfaces List
-------------------- ------------------------------------
VLAN0002             Et0/1
```

SWACC1(config)#interface ethernet 0/0
SWACC1(config-if)#shutdown

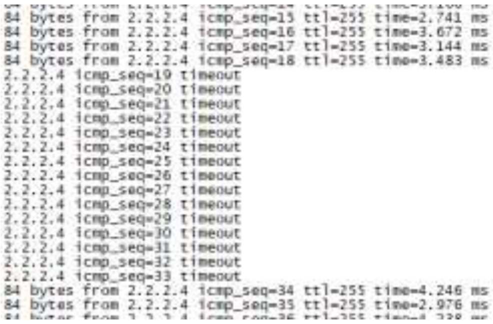


```
%LINK-5-CHANGED: Interface Ethernet0/0, changed state to administratively down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0, changed state to down
```



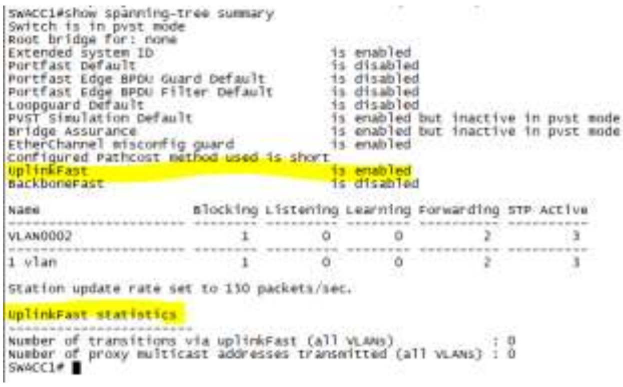
```
SWACC1#show spanning-tree blockedports

Name                 Blocked Interfaces List
-------------------- ------------------------------------

Number of blocked ports (segments) in the system : 0
```

From PC 1 ping continuous for observing the recovery time.



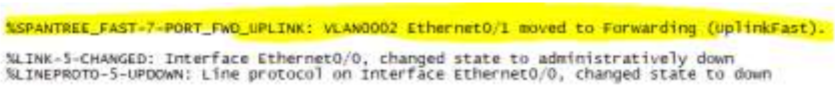
```
84 bytes from 2.2.2.4 icmp_seq=15 ttl=255 time=2.741 ms
84 bytes from 2.2.2.4 icmp_seq=16 ttl=255 time=3.672 ms
84 bytes from 2.2.2.4 icmp_seq=17 ttl=255 time=3.144 ms
84 bytes from 2.2.2.4 icmp_seq=18 ttl=255 time=3.483 ms
2.2.2.4 icmp_seq=19 timeout
2.2.2.4 icmp_seq=20 timeout
2.2.2.4 icmp_seq=21 timeout
2.2.2.4 icmp_seq=22 timeout
2.2.2.4 icmp_seq=23 timeout
2.2.2.4 icmp_seq=24 timeout
2.2.2.4 icmp_seq=25 timeout
2.2.2.4 icmp_seq=26 timeout
2.2.2.4 icmp_seq=27 timeout
2.2.2.4 icmp_seq=28 timeout
2.2.2.4 icmp_seq=29 timeout
2.2.2.4 icmp_seq=30 timeout
2.2.2.4 icmp_seq=31 timeout
2.2.2.4 icmp_seq=32 timeout
2.2.2.4 icmp_seq=33 timeout
84 bytes from 2.2.2.4 icmp_seq=34 ttl=255 time=4.246 ms
84 bytes from 2.2.2.4 icmp_seq=35 ttl=255 time=2.976 ms
84 bytes from 2.2.2.4 icmp_seq=36 ttl=255 time=4.238 ms
```

Enable spanningtree uplinkfast



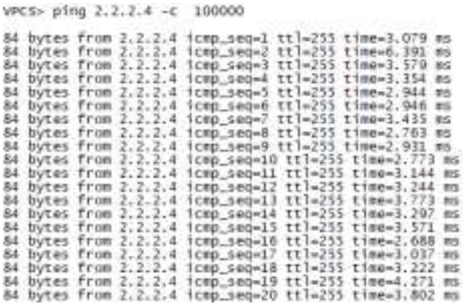
```
SWACC1#show spanning-tree summary
Switch is in pvst mode
Root bridge for: none
Extended system ID                      is enabled
Portfast Default                        is disabled
Portfast Edge BPDU Guard Default        is disabled
Portfast Edge BPDU Filter Default       is disabled
Loopguard Default                       is disabled
PVST Simulation Default                 is enabled but inactive in pvst mode
Bridge Assurance                        is enabled but inactive in pvst mode
EtherChannel misconfig guard            is enabled
Configured Pathcost method used is short
UplinkFast                              is enabled
BackboneFast                            is disabled

Name                   Blocking Listening Learning Forwarding STP Active
---------------------- -------- --------- -------- ---------- ----------
VLAN0002                      1         0        0          2          3
---------------------- -------- --------- -------- ---------- ----------
1 vlan                        1         0        0          2          3

Station update rate set to 150 packets/sec.

UplinkFast statistics
-----------------------
Number of transitions via uplinkFast (all VLANs)            : 0
Number of proxy multicast addresses transmitted (all VLANs) : 0
SWACC1# █
```

SWACC1(config)#interface ethernet 0/0
SWACC1(config-if)#shutdown



```
%SPANTREE_FAST-7-PORT_FWD_UPLINK: VLAN0002 Ethernet0/1 moved to Forwarding (UplinkFast).

%LINK-5-CHANGED: Interface Ethernet0/0, changed state to administratively down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0, changed state to down
```

PC side no ping loss

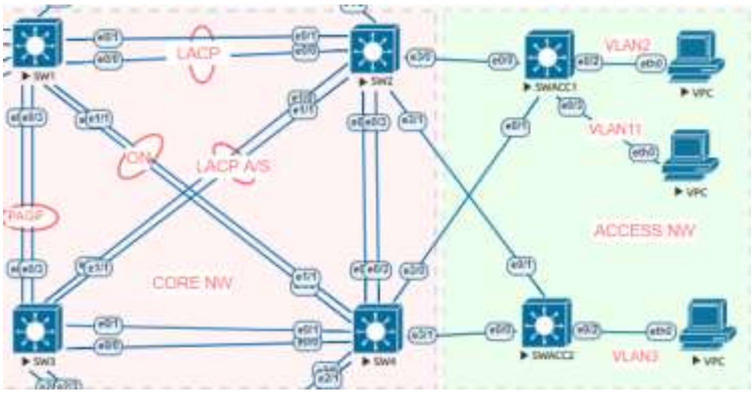


```
VPCS> ping 2.2.2.4 -c  100000

84 bytes from 2.2.2.4 icmp_seq=1 ttl=255 time=3.079 ms
84 bytes from 2.2.2.4 icmp_seq=2 ttl=255 time=6.391 ms
84 bytes from 2.2.2.4 icmp_seq=3 ttl=255 time=3.579 ms
84 bytes from 2.2.2.4 icmp_seq=4 ttl=255 time=3.354 ms
84 bytes from 2.2.2.4 icmp_seq=5 ttl=255 time=2.944 ms
84 bytes from 2.2.2.4 icmp_seq=6 ttl=255 time=2.946 ms
84 bytes from 2.2.2.4 icmp_seq=7 ttl=255 time=3.435 ms
84 bytes from 2.2.2.4 icmp_seq=8 ttl=255 time=2.763 ms
84 bytes from 2.2.2.4 icmp_seq=9 ttl=255 time=2.931 ms
84 bytes from 2.2.2.4 icmp_seq=10 ttl=255 time=2.773 ms
84 bytes from 2.2.2.4 icmp_seq=11 ttl=255 time=3.144 ms
84 bytes from 2.2.2.4 icmp_seq=12 ttl=255 time=3.244 ms
84 bytes from 2.2.2.4 icmp_seq=13 ttl=255 time=3.773 ms
84 bytes from 2.2.2.4 icmp_seq=14 ttl=255 time=3.297 ms
84 bytes from 2.2.2.4 icmp_seq=15 ttl=255 time=3.571 ms
84 bytes from 2.2.2.4 icmp_seq=16 ttl=255 time=2.688 ms
84 bytes from 2.2.2.4 icmp_seq=17 ttl=255 time=3.037 ms
84 bytes from 2.2.2.4 icmp_seq=18 ttl=255 time=3.222 ms
84 bytes from 2.2.2.4 icmp_seq=19 ttl=255 time=4.271 ms
84 bytes from 2.2.2.4 icmp_seq=20 ttl=255 time=3.802 ms
```

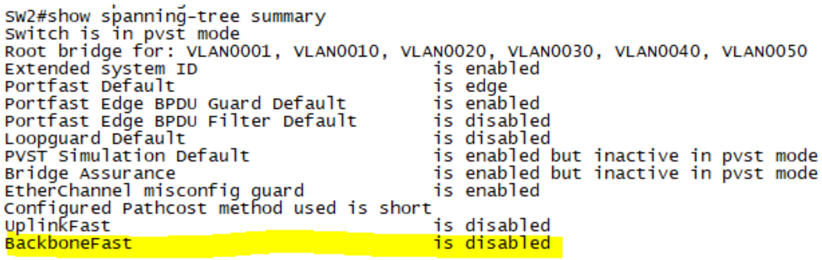
Checking spanningtree uplinkfast :

```
SWACC1#show spanning-tree uplinkfast
UplinkFast is enabled

Station update rate set to 150 packets/sec.

UplinkFast statistics
-----------------------
Number of transitions via uplinkFast (all VLANs)            : 2
Number of proxy multicast addresses transmitted (all VLANs) : 0

Name                 Interface List
-------------------- ------------------------------------
VLAN0002             Et0/0(fwd), Et0/1
```

BackboneFast – Enables a switch to recognize and respond to an indirect link failure by transitioning a blocking port to forwarding without waiting for the max-aging timer to expire, it moves straight to listening mode.

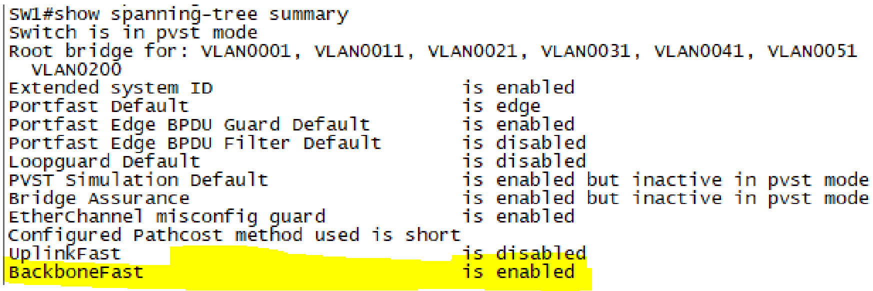


We are testing VLAN 11, the root bridge for the VLAN 11 is SW1, below tests before backbonefast enabled.



```
SW2#show spanning-tree summary
Switch is in pvst mode
Root bridge for: VLAN0001, VLAN0010, VLAN0020, VLAN0030, VLAN0040, VLAN0050
Extended system ID                      is enabled
Portfast Default                        is edge
Portfast Edge BPDU Guard Default        is enabled
Portfast Edge BPDU Filter Default       is disabled
Loopguard Default                       is disabled
PVST Simulation Default                 is enabled but inactive in pvst mode
Bridge Assurance                        is enabled but inactive in pvst mode
EtherChannel misconfig guard            is enabled
Configured Pathcost method used is short
UplinkFast                              is disabled
BackboneFast                            is disabled
```

You need to enable all backbone switches SW1, SW2, SW3, SW4 , if any alternative link fails, it will be routed to other route.
Config t
SW1(config)#spanning-tree backbonefast



```
SW1#show spanning-tree summary
Switch is in pvst mode
Root bridge for: VLAN0001, VLAN0011, VLAN0021, VLAN0031, VLAN0041, VLAN0051
  VLAN0200
Extended system ID                      is enabled
Portfast Default                        is edge
Portfast Edge BPDU Guard Default        is enabled
Portfast Edge BPDU Filter Default       is disabled
Loopguard Default                       is disabled
PVST Simulation Default                 is enabled but inactive in pvst mode
Bridge Assurance                        is enabled but inactive in pvst mode
EtherChannel misconfig guard            is enabled
Configured Pathcost method used is short
UplinkFast                              is disabled
BackboneFast                            is enabled
```

1.1.f [iv] loopguard, rootguard

LoopGuard – LoopGuard enables detection of uni-directional link failures, where an interface is only able to send or receive.

UDLD – can be enabled on interfaces that are not running STP. The loop guard feature in inherit to STP and must be configured on an L2 interface only.

Loop Guard
You can use loop guard to prevent alternate or root ports from becoming designated ports because of a failure that leads to a unidirectional link. This feature is most effective when it is enabled on the entire

switched network. Loop guard prevents alternate and root ports from becoming designated ports, and spanning tree does not send BPDUs on root or alternate ports.When the switch is operating in PVST+ or rapid-PVST+ mode, loop guard prevents alternate and root ports from becoming designated ports, and spanning tree does not send BPDUs on root or alternate ports.
When the switch is operating in MST mode, BPDUs are not sent on nonboundary ports only if the interface is blocked by loop guard in all MST instances. On a boundary port, loop guard blocks the interface in all MST instances.

Config-
Conf# spanning-tree loopguard default
Conf-if# spanning-tree guard loop

RootGuard

Root guard makes sure that if we receive a superior BPDU on certain ports that we don't trust them, and we can put the port into blocking state. Root guard should be enabled on "edge" or access ports. It's usually accompanied by portfast. When a superior BPDU comes in, our ports go into root inconsistent-state. If inferior BPDUs come to that port, the port should come back up.

Configure RootGuard:

From the root switch – on the downstream facing interfaces.
# spanning-tree guard root

1.1.g Implement and troubleshoot other LAN switching technologies

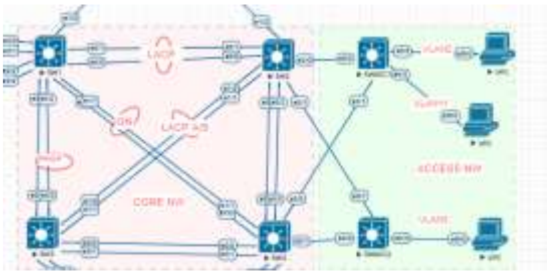
1.1.g [i] SPAN, RSPAN, ERSPAN

SPAN

Switchport analyzer (SPAN)
We need visibility on the traffic coming into our LAN. We have a network monitor device that we want to send copies of the frames to. SPAN ports allow you to mirror a port to another one so you can capture unicast data on the side.

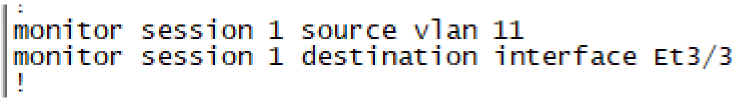
In span you define the source and destination ports, the source being the original port you want to clone.

You can choose to copy ingress, egress or both.
Local SPAN – the source and destination ports all belong to the same switch



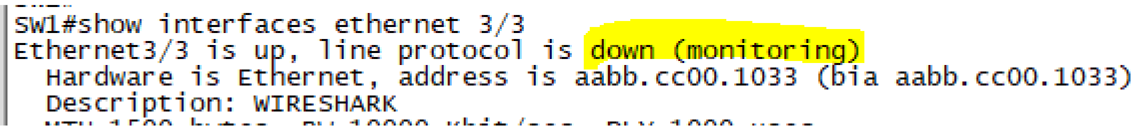
Above diagram SW1 have interface VLAN 11 and we going to SPAN the traffic to Interface 3/3 and run Wireshark and see the traffic.

Here is the configuration on SW1:



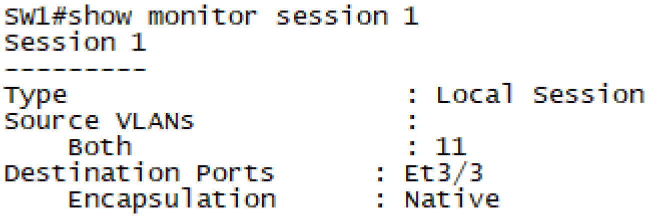
```
monitor session 1 source vlan 11
monitor session 1 destination interface Et3/3
!
```

You can observe now that interface is in Monitoring.



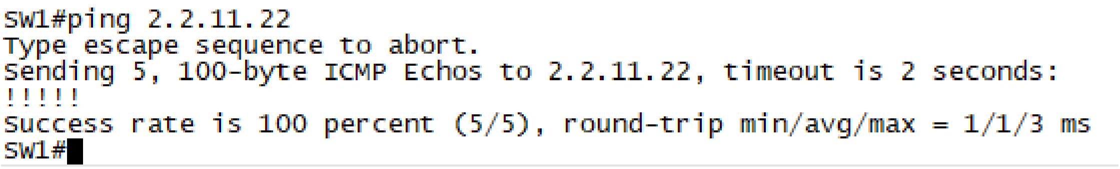
```
SW1#show interfaces ethernet 3/3
Ethernet3/3 is up, line protocol is down (monitoring)
  Hardware is Ethernet, address is aabb.cc00.1033 (bia aabb.cc00.1033)
  Description: WIRESHARK
```

Verify the session:

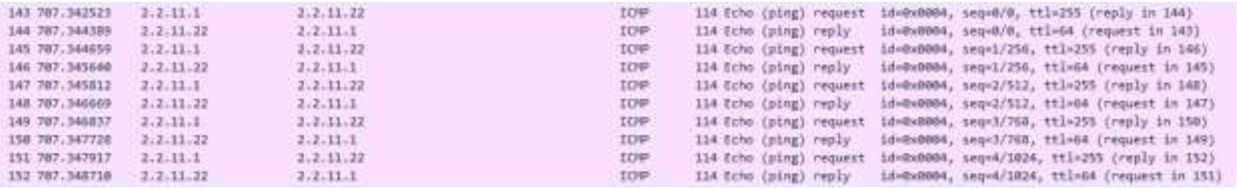


```
SW1#show monitor session 1
Session 1
---------
Type                     : Local Session
Source VLANs             :
    Both                 : 11
Destination Ports      : Et3/3
    Encapsulation      : Native
```

Now initiate the ping from SW1 to PC with IP: from 2.2.11.1 to 2.2.11.22 and observe the Wireshark output which was connected to Ethernet 3/3.



```
SW1#ping 2.2.11.22
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.11.22, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/3 ms
SW1#
```



```
143 707.342523  2.2.11.1     2.2.11.22    ICMP   114 Echo (ping) request  id=0x0004, seq=0/0, ttl=255 (reply in 144)
144 707.344389  2.2.11.22    2.2.11.1     ICMP   114 Echo (ping) reply    id=0x0004, seq=0/0, ttl=64 (request in 143)
145 707.344659  2.2.11.1     2.2.11.22    ICMP   114 Echo (ping) request  id=0x0004, seq=1/256, ttl=255 (reply in 146)
146 707.345640  2.2.11.22    2.2.11.1     ICMP   114 Echo (ping) reply    id=0x0004, seq=1/256, ttl=64 (request in 145)
147 707.345812  2.2.11.1     2.2.11.22    ICMP   114 Echo (ping) request  id=0x0004, seq=2/512, ttl=255 (reply in 148)
148 707.346669  2.2.11.22    2.2.11.1     ICMP   114 Echo (ping) reply    id=0x0004, seq=2/512, ttl=64 (request in 147)
149 707.346837  2.2.11.1     2.2.11.22    ICMP   114 Echo (ping) request  id=0x0004, seq=3/768, ttl=255 (reply in 150)
150 707.347728  2.2.11.22    2.2.11.1     ICMP   114 Echo (ping) reply    id=0x0004, seq=3/768, ttl=64 (request in 149)
151 707.347917  2.2.11.1     2.2.11.22    ICMP   114 Echo (ping) request  id=0x0004, seq=4/1024, ttl=255 (reply in 152)
152 707.348710  2.2.11.22    2.2.11.1     ICMP   114 Echo (ping) reply    id=0x0004, seq=4/1024, ttl=64 (request in 151)
```

RSPAN

Remote SPAN – The source and destination ports are not on the same switch.

Remote SPAN requires a trunk since RSPAN traffic is on a different vlan.

ERSPAN

ERSPAN (enhanced). Encapsulating frames into GRE then IP so it can be routed to a destination

1.2 Layer 2 Multicast

1.2.a Implement and troubleshoot IGMP

IGMP is used to dynamically register individual hosts in a multicast group on a particular LAN. Enabling PIM on an interface also enables IGMP. IGMP provides a means to automatically control and limit the flow of multicast traffic throughout your network with the use of special multicast queriers and hosts.

A querier is a network device, such as a router, that sends query messages to discover which network devices are members of a given multicast group.

A host is a receiver, including routers, that sends report messages (in response to query messages) to inform the querier of a host membership. Hosts use IGMP messages to join and leave multicast groups.

Hosts identify group memberships by sending IGMP messages to their local multicast device. Under IGMP, devices listen to IGMP messages and periodically send out queries to discover which groups are active or inactive on a particular subnet.

IGMP Multicast Addresses

IP multicast traffic uses group addresses, which are Class D IP addresses. The high-order four bits of a Class D address are 1110. Therefore, host group addresses can be in the range 224.0.0.0 to 239.255.255.255.

Multicast addresses in the range 224.0.0.0 to 224.0.0.255 are reserved for use by routing protocols and other network control traffic. The address 224.0.0.0 is guaranteed not to be assigned to any group.

IGMP packets are transmitted using IP multicast group addresses as follows:

- IGMP general queries are destined to the address 224.0.0.1 (all systems on a subnet).
- IGMP group-specific queries are destined to the group IP address for which the device is querying.
- IGMP group membership reports are destined to the group IP address for which the device is reporting.
- IGMPv2 leave-group messages are destined to the address 224.0.0.2 (all devices on a subnet).
- IGMPv3 membership reports are destined to the address 224.0.0.22; all IGMPv3-capable multicast devices must listen to this address.

1.2.a [I] IGMPv1, IGMPv2, IGMPv3

IGMP Versions

The switch supports IGMP version 1, IGMP version 2, and IGMP version 3.

IGMP Version 1

IGMP version 1 (IGMPv1) primarily uses a query-response model that enables the multicast router and multilayer switch to find which multicast groups are active (have one or more hosts interested in a multicast group) on the local subnet. IGMPv1 has other processes that enable a host to join and leave a multicast group.

IGMP Version 2

IGMPv2 extends IGMP functionality by providing such features as the IGMP leave process to reduce leave latency, group-specific queries, and an explicit maximum query response time. IGMPv2 also adds the capability for routers to elect the IGMP querier without depending on the multicast protocol to perform this task.

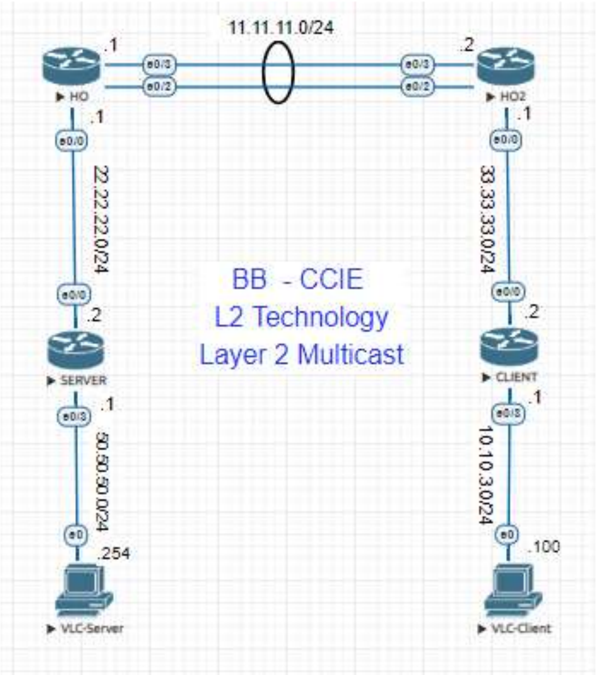
Note : IGMP version 2 is the default version for the switch.

IGMP Version 3

The switch supports IGMP version 3.

An IGMPv3 switch supports Basic IGMPv3 Snooping Support (BISS), which includes support for the snooping features on IGMPv1 and IGMPv2 switches and for IGMPv3 membership report messages. BISS constrains the flooding of multicast traffic when your network includes IGMPv3 hosts. It constrains traffic to approximately the same set of ports as the IGMP snooping feature on IGMPv2 or IGMPv1 hosts.

An IGMPv3 switch can receive messages from and forward messages to a device running the Source Specific Multicast (SSM) feature.



Enable multicast routing on H0, HO2, SERVER, CLIENT start with Dense mode.

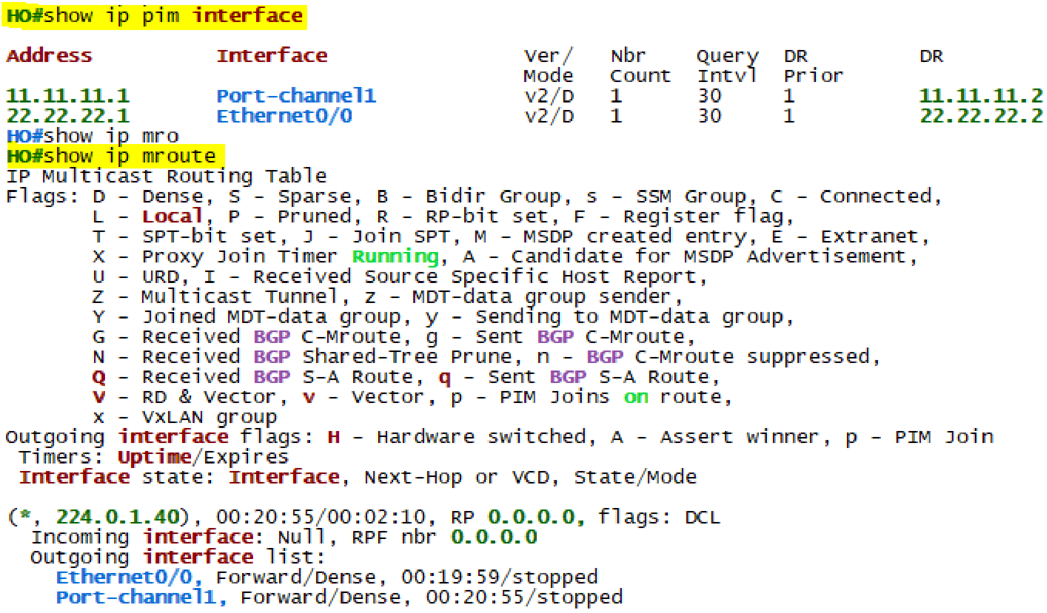
Here VLC Server is Streaming Video and VLC Client receiving

Dense mode uses a push model, meaning that the multicast traffic is flooded throughout the network. This is called a flood and prune model because initially the traffic is flooded to the network, but after the initial flood the routers with no receivers are pruned.

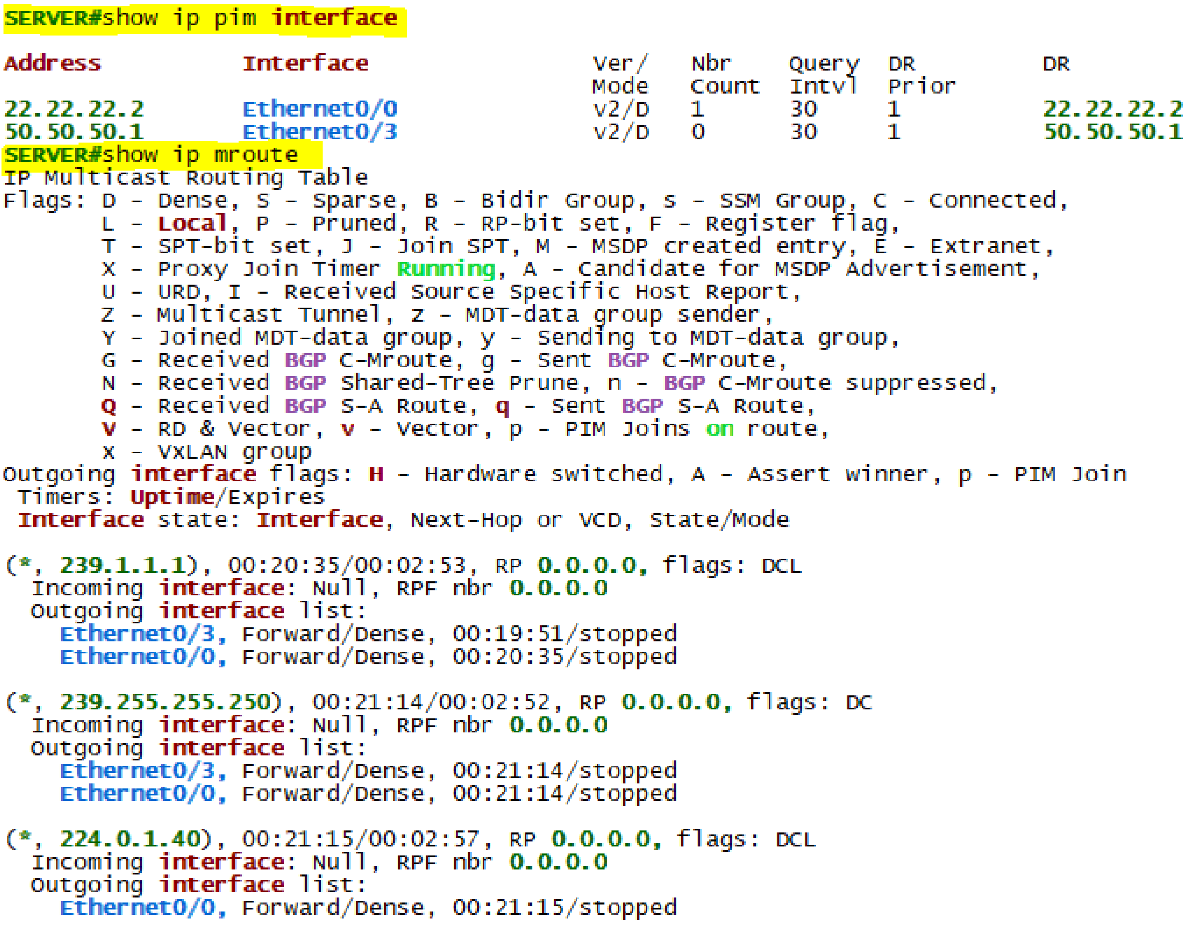
Below configuration for all the device configured:

HO	HO2	SERVER	Client
hostname HO ! ip multicast-routing ! ! ip cef no ipv6 cef ! ! ! interface Port-channel1 no switchport ip address 11.11.11.1 255.255.255.0 ip pim dense-mode ! interface Ethernet0/0 no switchport ip address 22.22.22.1 255.255.255.0 ip pim dense-mode ! interface Ethernet0/2 no switchport no ip address channel-group 1 mode active ! interface Ethernet0/3 no switchport no ip address channel-group 1 mode active	hostname HO2 ! ip multicast-routing ! ! ip cef no ipv6 cef ! ! ! interface Port-channel1 no switchport ip address 11.11.11.2 255.255.255.0 ip pim dense-mode ! interface Ethernet0/0 no switchport ip address 33.33.33.1 255.255.255.0 ip pim dense-mode ! interface Ethernet0/2 no switchport no ip address channel-group 1 mode active ! interface Ethernet0/3 no switchport no ip address channel-group 1 mode active	hostname SERVER ! ip multicast-routing ! ! ip cef no ipv6 cef ! interface Ethernet0/0 no switchport ip address 22.22.22.2 255.255.255.0 ip pim dense-mode ! interface Ethernet0/3 no switchport ip address 50.50.50.1 255.255.255.0 ip pim dense-mode ip igmp join-group 239.1.1.1	hostname CLIENT ! ip multicast-routing ! ! ip cef no ipv6 cef ! interface Ethernet0/0 no switchport ip address 33.33.33.2 255.255.255.0 ip pim dense-mode ! interface Ethernet0/3 no switchport ip address 10.10.3.1 255.255.255.0 ip pim dense-mode !

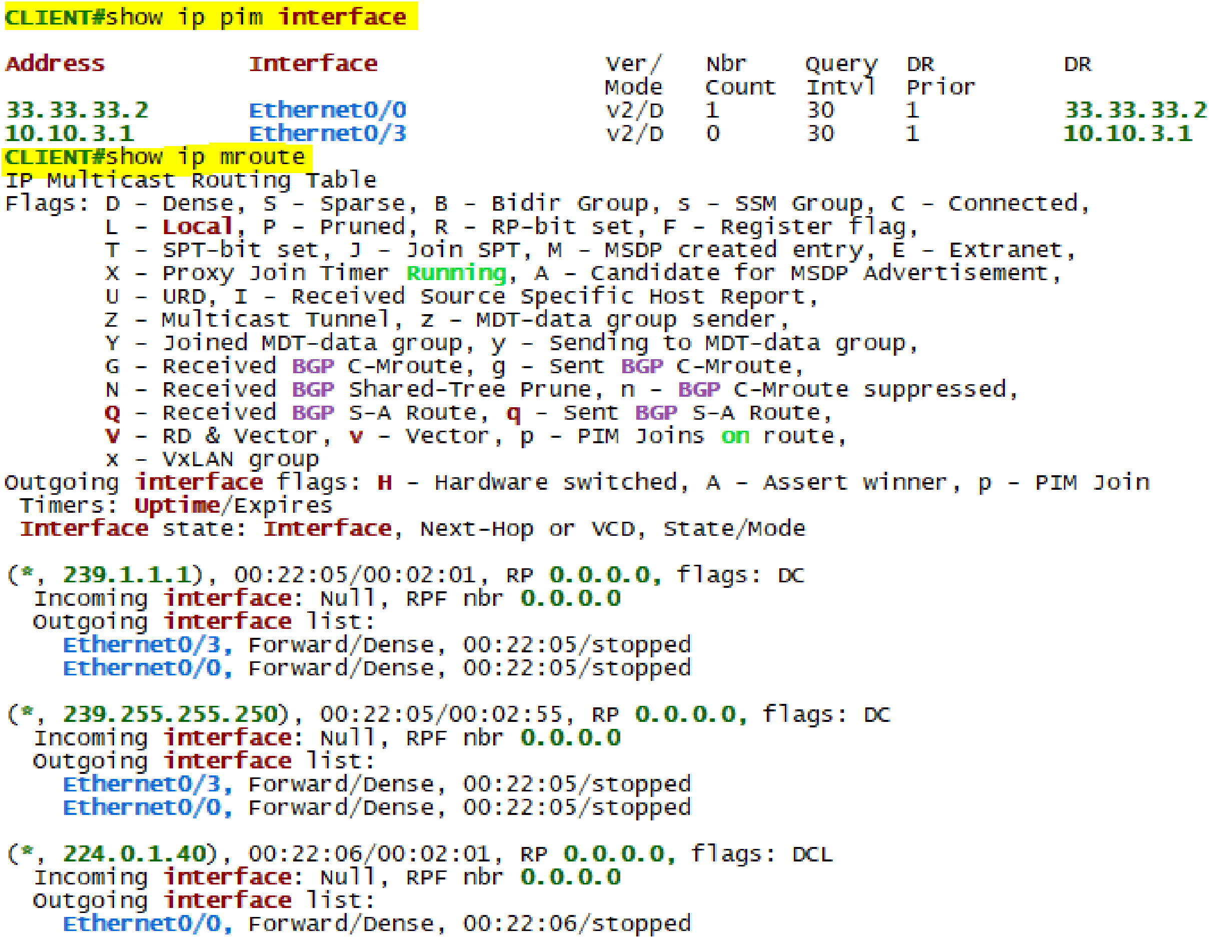
Verify the pim interface:



```
HO#show ip pim interface

Address          Interface                Ver/   Nbr    Query  DR         DR
                                          Mode   Count  Intvl  Prior
11.11.11.1       Port-channel1            v2/D   1      30     1          11.11.11.2
22.22.22.1       Ethernet0/0              v2/D   1      30     1          22.22.22.2
HO#show ip mro
HO#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 224.0.1.40), 00:20:55/00:02:10, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Dense, 00:19:59/stopped
    Port-channel1, Forward/Dense, 00:20:55/stopped
```



```
SERVER#show ip pim interface

Address          Interface                Ver/   Nbr    Query  DR         DR
                                          Mode   Count  Intvl  Prior
22.22.22.2       Ethernet0/0              v2/D   1      30     1          22.22.22.2
50.50.50.1       Ethernet0/3              v2/D   0      30     1          50.50.50.1
SERVER#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:20:35/00:02:53, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/3, Forward/Dense, 00:19:51/stopped
    Ethernet0/0, Forward/Dense, 00:20:35/stopped

(*, 239.255.255.250), 00:21:14/00:02:52, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/3, Forward/Dense, 00:21:14/stopped
    Ethernet0/0, Forward/Dense, 00:21:14/stopped

(*, 224.0.1.40), 00:21:15/00:02:57, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Dense, 00:21:15/stopped
```



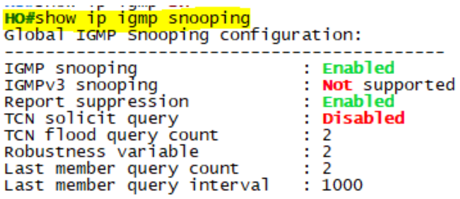
```
CLIENT#show ip pim interface

Address          Interface                Ver/   Nbr    Query  DR         DR
                                          Mode   Count  Intvl  Prior
33.33.33.2       Ethernet0/0              v2/D   1      30     1          33.33.33.2
10.10.3.1        Ethernet0/3              v2/D   0      30     1          10.10.3.1
CLIENT#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:22:05/00:02:01, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/3, Forward/Dense, 00:22:05/stopped
    Ethernet0/0, Forward/Dense, 00:22:05/stopped

(*, 239.255.255.250), 00:22:05/00:02:55, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/3, Forward/Dense, 00:22:05/stopped
    Ethernet0/0, Forward/Dense, 00:22:05/stopped

(*, 224.0.1.40), 00:22:06/00:02:01, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Dense, 00:22:06/stopped
```

1.2.a [ii] IGMP snooping

- Allows switch to listen to IGMP conversations between host and router
- When switch hears an report for host to multicast group it adds a host port number to the GDA list
- When it hears a IGMP leave it removed the hosts port from the CAM table entry
- Learning Router Port
  - IGMP Membership query send to 01-00-5e-00-00-01 or 224.0.0.1
  - PIMv1 hello send to 01-00-5e-00-00-02 or 224.0.0.2
  - PIMv2 hello send to 01-00-5e-00-00-0d or 224.0.0.13
  - DVMRP probes send to 01-00-5e-00-04 or or 224.0.0.4
  - MOSPF message send to 01-00-5e-00-05 or 06 or or 224.0.0.5 / 06
- Enabled on per-VLAN basis

By default, when PIM is configured on an interface, IGMPv2 is also enabled; the version of IGMP can be changed using the ip igmp version command.

IGMP is enabled on interface
Current IGMP host version is 2



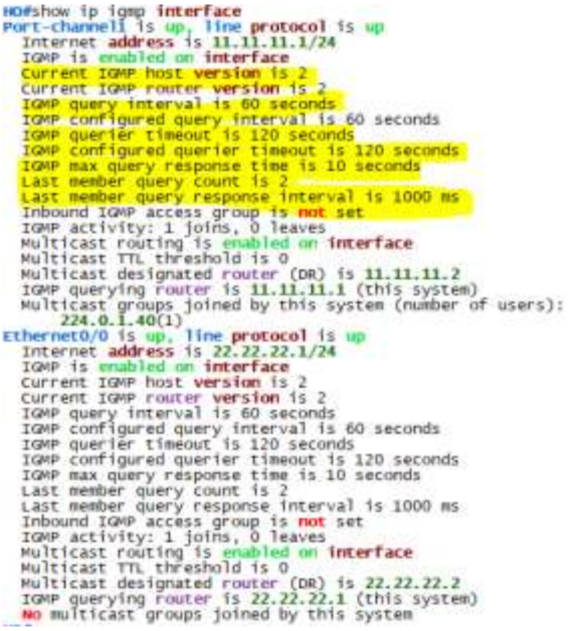
```
HO#show ip igmp snooping
Global IGMP Snooping configuration:
-------------------------------------------
IGMP snooping                : Enabled
IGMPv3 snooping              : Not supported
Report suppression           : Enabled
TCN solicit query            : Disabled
TCN flood query count        : 2
Robustness variable          : 2
Last member query count      : 2
Last member query interval   : 1000
```

1.2.a [iii] IGMP querier

Sends periodic IGMP queries that trigger IGMP report messages from hosts that want to receive IP multicast traffic.

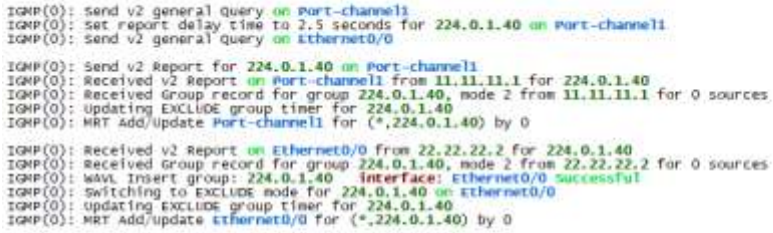
Default:

IGMP query interval is 60 seconds
IGMP querier timeout is 120 seconds



```
HO#show ip igmp interface
Port-channel1 is up, line protocol is up
  Internet address is 11.11.11.1/24
  IGMP is enabled on interface
  Current IGMP host version is 2
  Current IGMP router version is 2
  IGMP query interval is 60 seconds
  IGMP configured query interval is 60 seconds
  IGMP querier timeout is 120 seconds
  IGMP configured querier timeout is 120 seconds
  IGMP max query response time is 10 seconds
  Last member query count is 2
  Last member query response interval is 1000 ms
  Inbound IGMP access group is not set
  IGMP activity: 1 joins, 0 leaves
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 11.11.11.2
  IGMP querying router is 11.11.11.1 (this system)
  Multicast groups joined by this system (number of users):
      224.0.1.40(1)
Ethernet0/0 is up, line protocol is up
  Internet address is 22.22.22.1/24
  IGMP is enabled on interface
  Current IGMP host version is 2
  Current IGMP router version is 2
  IGMP query interval is 60 seconds
  IGMP configured query interval is 60 seconds
  IGMP querier timeout is 120 seconds
  IGMP configured querier timeout is 120 seconds
  IGMP max query response time is 10 seconds
  Last member query count is 2
  Last member query response interval is 1000 ms
  Inbound IGMP access group is not set
  IGMP activity: 1 joins, 0 leaves
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 22.22.22.2
  IGMP querying router is 22.22.22.1 (this system)
  No multicast groups joined by this system
```

Ip debug igmp :



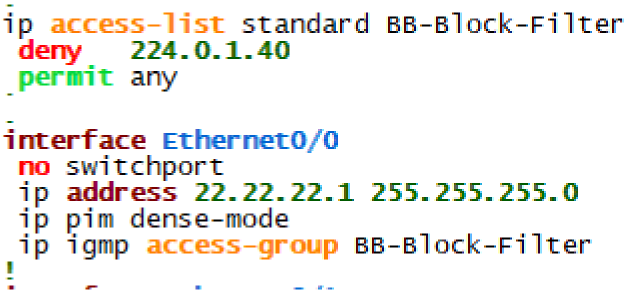
```
IGMP(0): Send v2 general Query on Port-channel1
IGMP(0): Set report delay time to 2.5 seconds for 224.0.1.40 on Port-channel1
IGMP(0): Send v2 general Query on Ethernet0/0

IGMP(0): Send v2 Report for 224.0.1.40 on Port-channel1
IGMP(0): Received v2 Report on Port-channel1 from 11.11.11.1 for 224.0.1.40
IGMP(0): Received Group record for group 224.0.1.40, mode 2 from 11.11.11.1 for 0 sources
IGMP(0): Updating EXCLUDE group timer for 224.0.1.40
IGMP(0): MRT Add/Update Port-channel1 for (*,224.0.1.40) by 0

IGMP(0): Received v2 Report on Ethernet0/0 from 22.22.22.2 for 224.0.1.40
IGMP(0): Received Group record for group 224.0.1.40, mode 2 from 22.22.22.2 for 0 sources
IGMP(0): WAVL Insert group: 224.0.1.40   interface: Ethernet0/0 Successful
IGMP(0): Switching to EXCLUDE mode for 224.0.1.40 on Ethernet0/0
IGMP(0): Updating EXCLUDE group timer for 224.0.1.40
IGMP(0): MRT Add/update Ethernet0/0 for (*,224.0.1.40) by 0
```

1.2.a [iv] IGMP filter

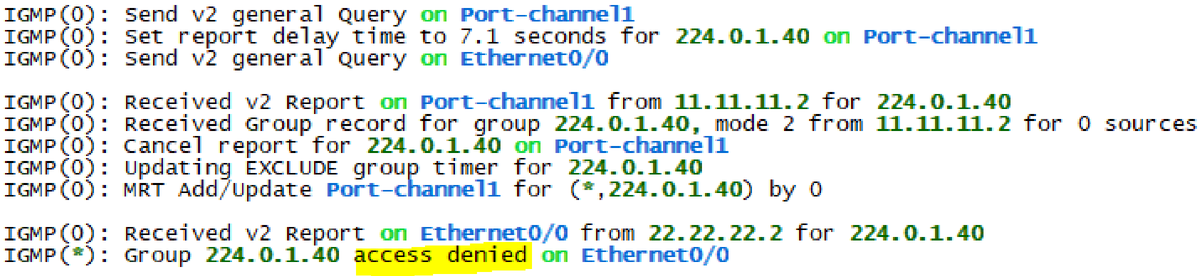
Multicast IGMP membership report messages include the multicast group addresses that our receivers want to join. By default, all multicast groups will be accepted.

It is possible to filter certain multicast groups. We can configure IGMP filtering on a multicast router or on a switch where IGMP snooping is enabled.

Let's create access-list to test :



```
ip access-list standard BB-Block-Filter
 deny   224.0.1.40
 permit any

interface Ethernet0/0
 no switchport
 ip address 22.22.22.1 255.255.255.0
 ip pim dense-mode
 ip igmp access-group BB-Block-Filter
!
```



```
IGMP(0): Send v2 general Query on Port-channel1
IGMP(0): Set report delay time to 7.1 seconds for 224.0.1.40 on Port-channel1
IGMP(0): Send v2 general Query on Ethernet0/0

IGMP(0): Received v2 Report on Port-channel1 from 11.11.11.2 for 224.0.1.40
IGMP(0): Received Group record for group 224.0.1.40, mode 2 from 11.11.11.2 for 0 sources
IGMP(0): Cancel report for 224.0.1.40 on Port-channel1
IGMP(0): Updating EXCLUDE group timer for 224.0.1.40
IGMP(0): MRT Add/Update Port-channel1 for (*,224.0.1.40) by 0

IGMP(0): Received v2 Report on Ethernet0/0 from 22.22.22.2 for 224.0.1.40
IGMP(*): Group 224.0.1.40 access denied on Ethernet0/0
```

1.2.a [v] IGMP proxy

Enables hosts in unidirectional link routing (UDLP) environment that are not directly connected to a downstream router to join a multicast group sources from an upstream network

Besides adding PIM, we use two important commands:

- **ip pim proxy-server**: this command enables the actual proxying of IGMP membership reports.
- **ip igmp helper-address udl**: this command tells the router to send IGMP membership reports to an upstream router that is connected on our UDL interface.

1.3 Layer 2 WAN circuit technologies

1.3.a Implement and troubleshoot HDLC

HDLC is a bit -oriented synchronous data link layer protocol. It supports various layer 3 protocols in addition to IP.

Cisco High-Level Data Link Controller (HDLC) is the Cisco proprietary protocol for sending data over synchronous serial links using HDLC. Cisco HDLC also provides a simple control protocol called Serial Line Address Resolution Protocol (SLARP) to maintain serial link keepalives. Cisco HDLC is the default for data encapsulation at Layer 2 (data link) of the Open System Interconnection (OSI) stack for efficient packet delineation and error control.

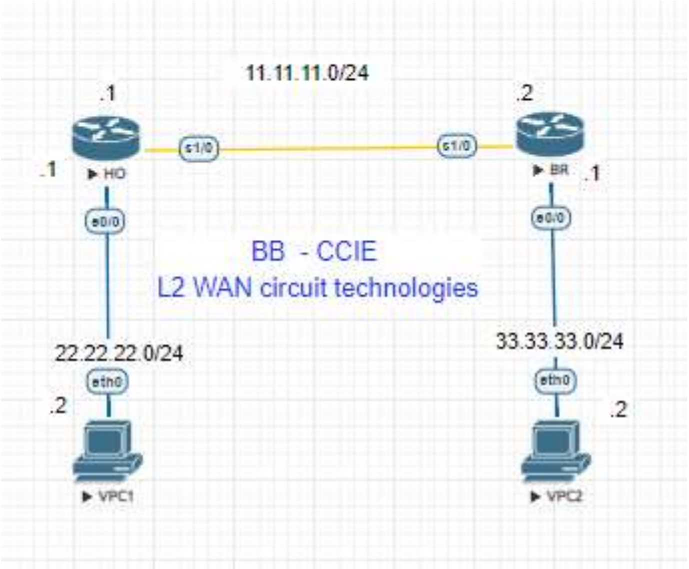
Note: Cisco HDLC is the default encapsulation type for the serial interfaces.

Cisco HDLC control protocol – Serial Line Address Resolution Protocol (SLARP) to maintain serial link keepalives.

DTE-DCE

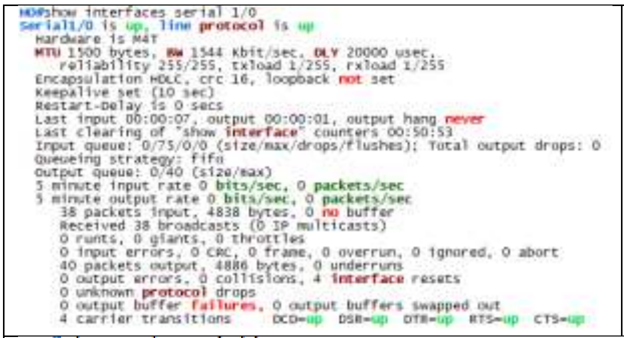
The communication through the WAN is through the DTE and the DCE device. A serial link is usually made up of two DCE devices at each end. The DCEs connect to DTEs in the remote LAN networks.

The DTE is usually a router or similar device. This is usually the source of the information at a layer 2 perspective. The DTE usually sends the data to the DCE.

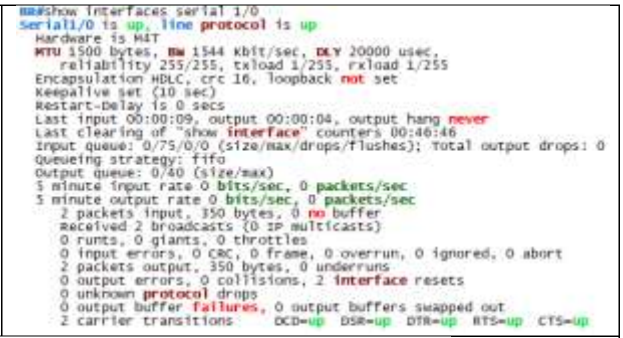


Default encapsulation HDLC.

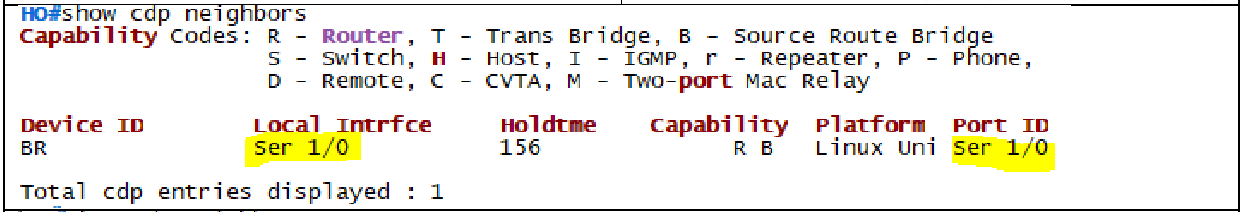
HO	BR
interface Serial1/0 serial restart-delay 0 ip address 11.11.11.1 255.255.255.0	interface Serial1/0 serial restart-delay 0 ip address 11.11.11.2 255.255.255.0



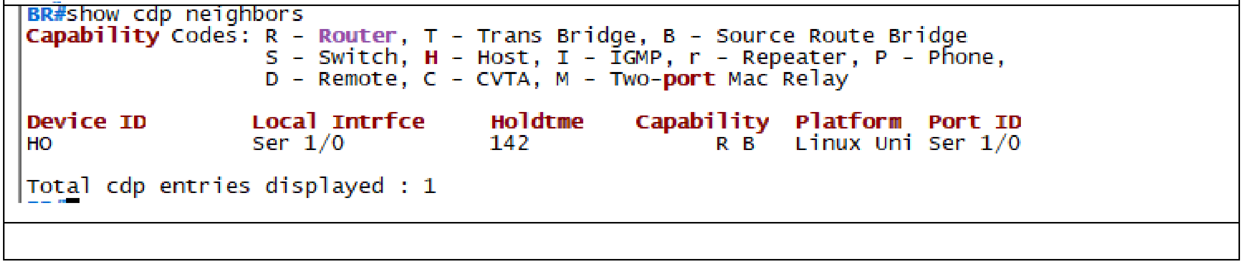
```
HO#show interfaces serial 1/0
Serial1/0 is up, line protocol is up
  Hardware is M4T
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  Last input 00:00:07, output 00:00:01, output hang never
  Last clearing of "show interface" counters 00:50:53
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     38 packets input, 4838 bytes, 0 no buffer
     Received 38 broadcasts (0 IP multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     40 packets output, 4886 bytes, 0 underruns
     0 output errors, 0 collisions, 4 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     4 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```



```
BR#show interfaces serial 1/0
Serial1/0 is up, line protocol is up
  Hardware is M4T
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  Last input 00:00:09, output 00:00:04, output hang never
  Last clearing of "show interface" counters 00:46:46
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     2 packets input, 350 bytes, 0 no buffer
     Received 2 broadcasts (0 IP multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     2 packets output, 350 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```



```
HO#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone,
                  D - Remote, C - CVTA, M - Two-port Mac Relay

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
BR               Ser 1/0           156              R B   Linux Uni Ser 1/0

Total cdp entries displayed : 1
```



```
BR#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone,
                  D - Remote, C - CVTA, M - Two-port Mac Relay

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
HO               Ser 1/0           142              R B   Linux Uni Ser 1/0

Total cdp entries displayed : 1
```

1.3.b Implement and troubleshoot PPP

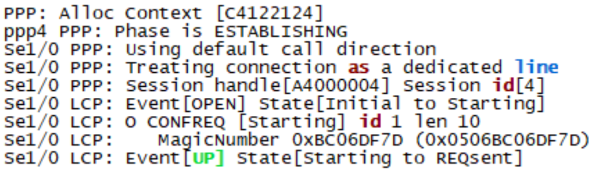
PPP, while more detailed than HDLC, is still very easy to use and troubleshoot. It was designed by simply taking HDLC and adding some fields in order to allow it to specify protocols carried within the L2 frame. This protocol field allows PPP to use many additional features such as authentication and address assignment. Activating PPP itself on an interface is as simple as using the interface **encapsulation ppp** command. While you could stop at this point and you would have a fully functional link, the basic reason to do so is to use some of the added features that PPP gives.

PPP (point-to-point protocol)

Media independent encapsulation – also very lightweight
-serial, Ethernet, Frame Relay, ATM.
-encapsulation PPP
Adds features that other layer 2 medias don't natively support.
-authentication
-Multilink
-fragmentation
-reliability

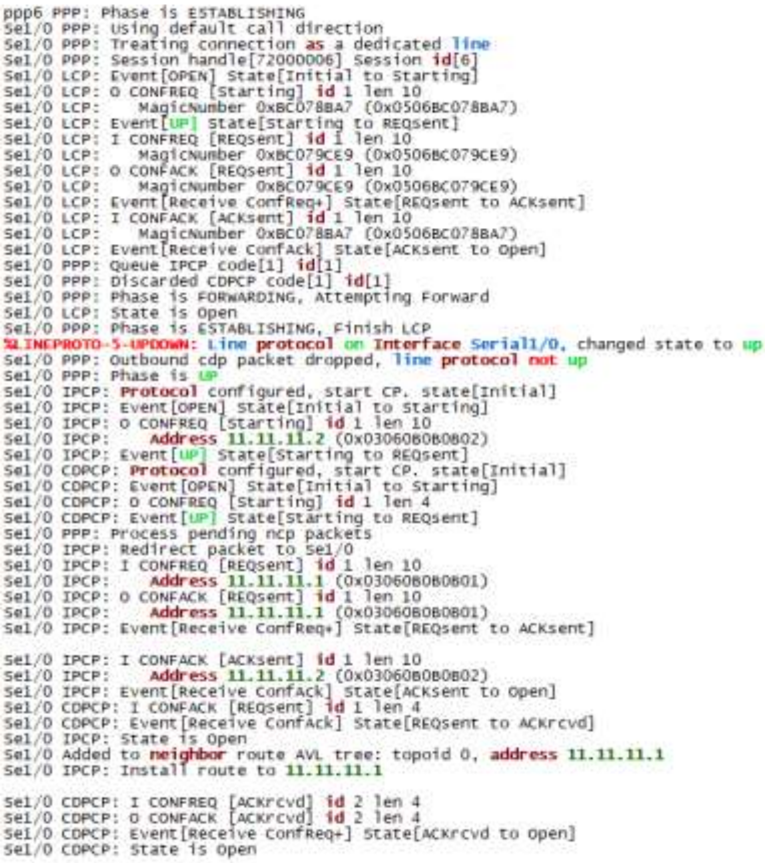
HO	BR
interface Serial1/0 encapsulation ppp serial restart-delay 0 ip address 11.11.11.1 255.255.255.0	interface Serial1/0 encapsulation ppp serial restart-delay 0 ip address 11.11.11.2 255.255.255.0

Debug ppp negotiation will give debug



```
PPP: Alloc Context [C4122124]
ppp4 PPP: Phase is ESTABLISHING
Se1/0 PPP: Using default call direction
Se1/0 PPP: Treating connection as a dedicated line
Se1/0 PPP: Session handle[A4000004] Session id[4]
Se1/0 LCP: Event[OPEN] State[Initial to Starting]
Se1/0 LCP: O CONFREQ [Starting] id 1 len 10
Se1/0 LCP:    MagicNumber 0xBC06DF7D (0x0506BC06DF7D)
Se1/0 LCP: Event[UP] State[Starting to REQsent]
```

As soon as other found PPP



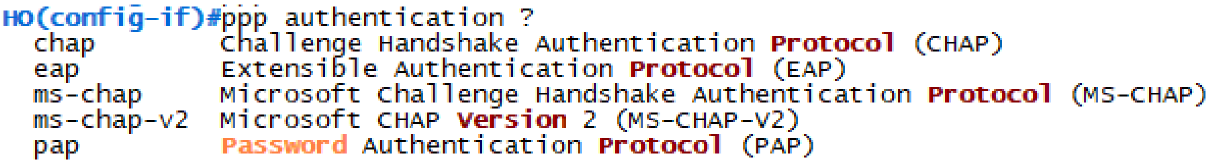
```
ppp6 PPP: Phase is ESTABLISHING
Se1/0 PPP: Using default call direction
Se1/0 PPP: Treating connection as a dedicated line
Se1/0 PPP: Session handle[72000006] Session id[6]
Se1/0 LCP: Event[OPEN] State[Initial to Starting]
Se1/0 LCP: O CONFREQ [Starting] id 1 len 10
Se1/0 LCP:    MagicNumber 0xBC078BA7 (0x0506BC078BA7)
Se1/0 LCP: Event[UP] State[Starting to REQsent]
Se1/0 LCP: I CONFREQ [REQsent] id 1 len 10
Se1/0 LCP:    MagicNumber 0xBC079CE9 (0x0506BC079CE9)
Se1/0 LCP: O CONFACK [REQsent] id 1 len 10
Se1/0 LCP:    MagicNumber 0xBC079CE9 (0x0506BC079CE9)
Se1/0 LCP: Event[Receive ConfReq+] State[REQsent to ACKsent]
Se1/0 LCP: I CONFACK [ACKsent] id 1 len 10
Se1/0 LCP:    MagicNumber 0xBC078BA7 (0x0506BC078BA7)
Se1/0 LCP: Event[Receive ConfAck] State[ACKsent to Open]
Se1/0 PPP: Queue IPCP code[1] id[1]
Se1/0 PPP: Discarded CDPCP code[1] id[1]
Se1/0 PPP: Phase is FORWARDING, Attempting Forward
Se1/0 LCP: State is Open
Se1/0 PPP: Phase is ESTABLISHING, Finish LCP
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0, changed state to up
Se1/0 PPP: Outbound cdp packet dropped, line protocol not up
Se1/0 PPP: Phase is UP
Se1/0 IPCP: Protocol configured, start CP. state[Initial]
Se1/0 IPCP: Event[OPEN] State[Initial to Starting]
Se1/0 IPCP: O CONFREQ [Starting] id 1 len 10
Se1/0 IPCP:    Address 11.11.11.2 (0x03060B0B0B02)
Se1/0 IPCP: Event[UP] State[Starting to REQsent]
Se1/0 CDPCP: Protocol configured, start CP. state[Initial]
Se1/0 CDPCP: Event[OPEN] State[Initial to Starting]
Se1/0 CDPCP: O CONFREQ [Starting] id 1 len 4
Se1/0 CDPCP: Event[UP] State[Starting to REQsent]
Se1/0 PPP: Process pending ncp packets
Se1/0 IPCP: Redirect packet to Se1/0
Se1/0 IPCP: I CONFREQ [REQsent] id 1 len 10
Se1/0 IPCP:    Address 11.11.11.1 (0x03060B0B0B01)
Se1/0 IPCP: O CONFACK [REQsent] id 1 len 10
Se1/0 IPCP:    Address 11.11.11.1 (0x03060B0B0B01)
Se1/0 IPCP: Event[Receive ConfReq+] State[REQsent to ACKsent]

Se1/0 IPCP: I CONFACK [ACKsent] id 1 len 10
Se1/0 IPCP:    Address 11.11.11.2 (0x03060B0B0B02)
Se1/0 IPCP: Event[Receive ConfAck] State[ACKsent to Open]
Se1/0 CDPCP: I CONFACK [REQsent] id 1 len 4
Se1/0 CDPCP: Event[Receive ConfAck] State[REQsent to ACKrcvd]
Se1/0 IPCP: State is Open
Se1/0 Added to neighbor route AVL tree: topoid 0, address 11.11.11.1
Se1/0 IPCP: Install route to 11.11.11.1

Se1/0 CDPCP: I CONFREQ [ACKrcvd] id 2 len 4
Se1/0 CDPCP: O CONFACK [ACKrcvd] id 2 len 4
Se1/0 CDPCP: Event[Receive ConfReq+] State[ACKrcvd to Open]
Se1/0 CDPCP: State is Open
```

1.3.b [i] authentication [PAP, CHAP]

PPP Authentication:

Password authentication protocol (PAP)
-clear text username
-clear test password

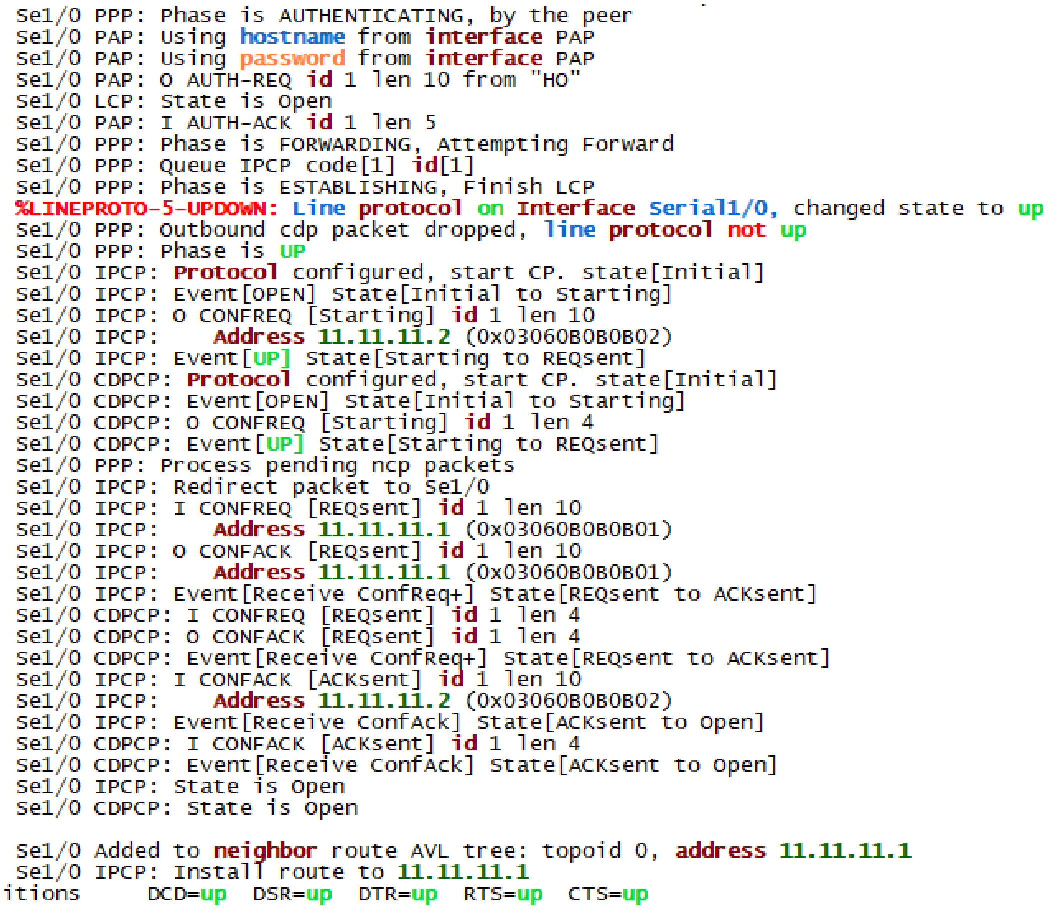
Challenge handshake authentication protocol (CHAP)
-clear text username
-MD5 hashed password
MSCHAP / MCHAPv2 / EAP/ etc



```
HO(config-if)#ppp authentication ?
  chap         Challenge Handshake Authentication Protocol (CHAP)
  eap          Extensible Authentication Protocol (EAP)
  ms-chap      Microsoft Challenge Handshake Authentication Protocol (MS-CHAP)
  ms-chap-v2   Microsoft CHAP Version 2 (MS-CHAP-V2)
  pap          Password Authentication Protocol (PAP)
```

Password authentication protocol (PAP)

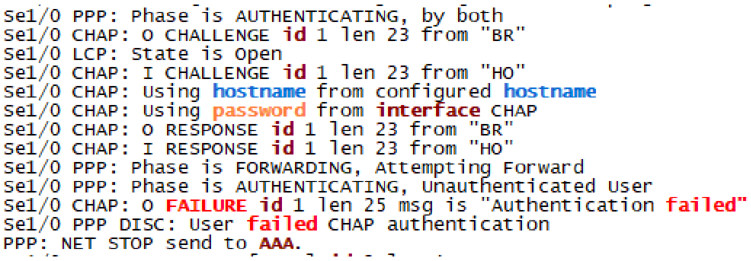
HO	BR
! **username HO password 0 BB** ! interface Serial1/0 ip address 11.11.11.1 255.255.255.0 encapsulation ppp ppp authentication pap serial restart-delay 0	interface Serial1/0 ip address 11.11.11.2 255.255.255.0 encapsulation ppp **ppp pap sent-username HO password 0 BB** serial restart-delay 0



```
 Se1/0 PPP: Phase is AUTHENTICATING, by the peer
 Se1/0 PAP: Using hostname from interface PAP
 Se1/0 PAP: Using password from interface PAP
 Se1/0 PAP: O AUTH-REQ id 1 len 10 from "HO"
 Se1/0 LCP: State is Open
 Se1/0 PAP: I AUTH-ACK id 1 len 5
 Se1/0 PPP: Phase is FORWARDING, Attempting Forward
 Se1/0 PPP: Queue IPCP code[1] id[1]
 Se1/0 PPP: Phase is ESTABLISHING, Finish LCP
 %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0, changed state to up
 Se1/0 PPP: Outbound cdp packet dropped, line protocol not up
 Se1/0 PPP: Phase is UP
 Se1/0 IPCP: Protocol configured, start CP. state[Initial]
 Se1/0 IPCP: Event[OPEN] State[Initial to Starting]
 Se1/0 IPCP: O CONFREQ [Starting] id 1 len 10
 Se1/0 IPCP:    Address 11.11.11.2 (0x03060B0B0B02)
 Se1/0 IPCP: Event[UP] State[Starting to REQsent]
 Se1/0 CDPCP: Protocol configured, start CP. state[Initial]
 Se1/0 CDPCP: Event[OPEN] State[Initial to Starting]
 Se1/0 CDPCP: O CONFREQ [Starting] id 1 len 4
 Se1/0 CDPCP: Event[UP] State[Starting to REQsent]
 Se1/0 PPP: Process pending ncp packets
 Se1/0 IPCP: Redirect packet to Se1/0
 Se1/0 IPCP: I CONFREQ [REQsent] id 1 len 10
 Se1/0 IPCP:    Address 11.11.11.1 (0x03060B0B0B01)
 Se1/0 IPCP: O CONFACK [REQsent] id 1 len 10
 Se1/0 IPCP:    Address 11.11.11.1 (0x03060B0B0B01)
 Se1/0 IPCP: Event[Receive ConfReq+] State[REQsent to ACKsent]
 Se1/0 CDPCP: I CONFREQ [REQsent] id 1 len 4
 Se1/0 CDPCP: O CONFACK [REQsent] id 1 len 4
 Se1/0 CDPCP: Event[Receive ConfReq+] State[REQsent to ACKsent]
 Se1/0 IPCP: I CONFACK [ACKsent] id 1 len 10
 Se1/0 IPCP:    Address 11.11.11.2 (0x03060B0B0B02)
 Se1/0 IPCP: Event[Receive ConfAck] State[ACKsent to Open]
 Se1/0 CDPCP: I CONFACK [ACKsent] id 1 len 4
 Se1/0 CDPCP: Event[Receive ConfAck] State[ACKsent to Open]
 Se1/0 IPCP: State is Open
 Se1/0 CDPCP: State is Open

 Se1/0 Added to neighbor route AVL tree: topoid 0, address 11.11.11.1
 Se1/0 IPCP: Install route to 11.11.11.1
itions      DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

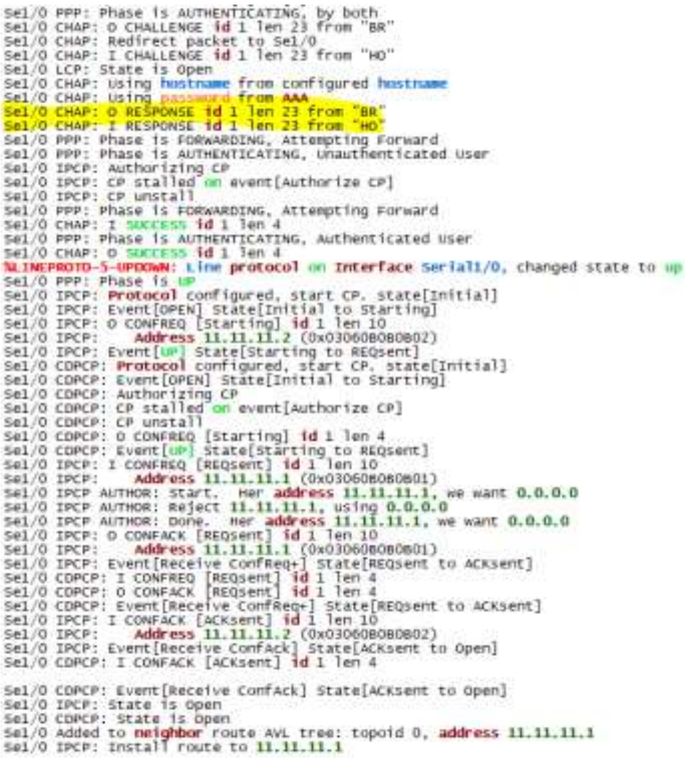
Challenge handshake authentication protocol (CHAP)

HO	BR
! **username BR password 0 BBHO** ! interface Serial1/0 ip address 11.11.11.1 255.255.255.0 encapsulation ppp ppp authentication chap serial restart-delay 0	! **username HO password 0 BBHO** **!** interface Serial1/0 ip address 11.11.11.2 255.255.255.0 encapsulation ppp ppp authentication chap serial restart-delay 0



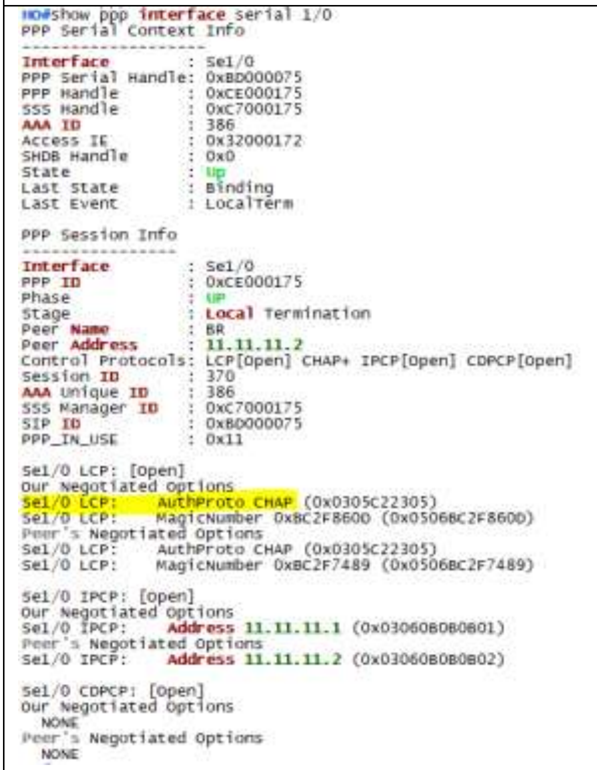
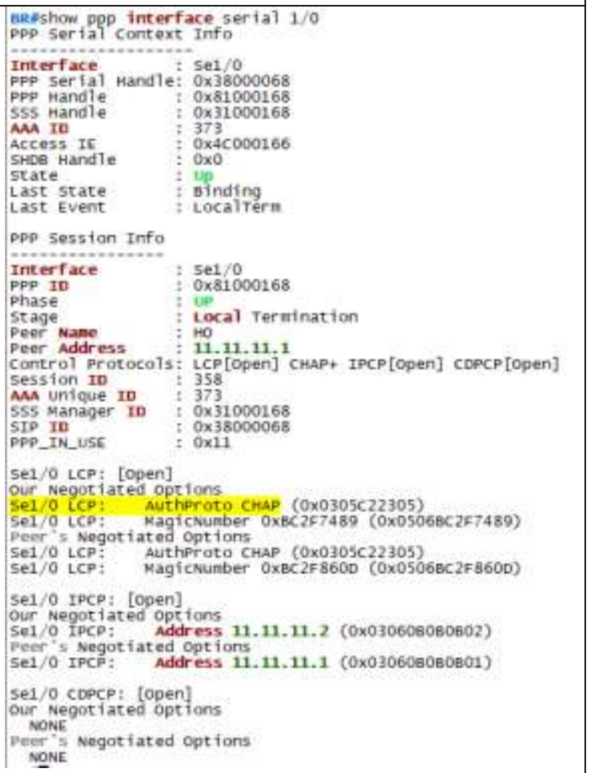
```
Se1/0 PPP: Phase is AUTHENTICATING, by both
Se1/0 CHAP: O CHALLENGE id 1 len 23 from "BR"
Se1/0 LCP: State is Open
Se1/0 CHAP: I CHALLENGE id 1 len 23 from "HO"
Se1/0 CHAP: Using hostname from configured hostname
Se1/0 CHAP: Using password from interface CHAP
Se1/0 CHAP: O RESPONSE id 1 len 23 from "BR"
Se1/0 CHAP: I RESPONSE id 1 len 23 from "HO"
Se1/0 PPP: Phase is FORWARDING, Attempting Forward
Se1/0 PPP: Phase is AUTHENTICATING, Unauthenticated User
Se1/0 CHAP: O FAILURE id 1 len 25 msg is "Authentication failed"
Se1/0 PPP DISC: User failed CHAP authentication
PPP: NET STOP send to AAA.
```

Once username and password correct



```
Se1/0 PPP: Phase is AUTHENTICATING, by both
Se1/0 CHAP: O CHALLENGE id 1 len 23 from "BR"
Se1/0 CHAP: Redirect packet to Se1/0
Se1/0 CHAP: I CHALLENGE id 1 len 23 from "HO"
Se1/0 LCP: State is Open
Se1/0 CHAP: Using hostname from configured hostname
Se1/0 CHAP: Using password from AAA
Se1/0 CHAP: O RESPONSE id 1 len 23 from "BR"
Se1/0 CHAP: I RESPONSE id 1 len 23 from "HO"
Se1/0 PPP: Phase is FORWARDING, Attempting Forward
Se1/0 PPP: Phase is AUTHENTICATING, Unauthenticated User
Se1/0 IPCP: Authorizing CP
Se1/0 IPCP: CP stalled on event[Authorize CP]
Se1/0 IPCP: CP unstall
Se1/0 PPP: Phase is FORWARDING, Attempting Forward
Se1/0 CHAP: I SUCCESS id 1 len 4
Se1/0 PPP: Phase is AUTHENTICATING, Authenticated User
Se1/0 CHAP: O SUCCESS id 1 len 4
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0, changed state to up
Se1/0 PPP: Phase is UP
Se1/0 IPCP: Protocol configured, start CP. state[Initial]
Se1/0 IPCP: Event[OPEN] State[Initial to Starting]
Se1/0 IPCP: O CONFREQ [Starting] id 1 len 10
Se1/0 IPCP:    Address 11.11.11.2 (0x03060B0B0B02)
Se1/0 IPCP: Event[UP] State[Starting to REQsent]
Se1/0 CDPCP: Protocol configured, start CP. state[Initial]
Se1/0 CDPCP: Event[OPEN] State[Initial to Starting]
Se1/0 CDPCP: Authorizing CP
Se1/0 CDPCP: CP stalled on event[Authorize CP]
Se1/0 CDPCP: CP unstall
Se1/0 CDPCP: O CONFREQ [Starting] id 1 len 4
Se1/0 CDPCP: Event[UP] State[Starting to REQsent]
Se1/0 IPCP: I CONFREQ [REQsent] id 1 len 10
Se1/0 IPCP:    Address 11.11.11.1 (0x03060B0B0B01)
Se1/0 IPCP AUTHOR: Start.  Her address 11.11.11.1, we want 0.0.0.0
Se1/0 IPCP AUTHOR: Reject 11.11.11.1, using 0.0.0.0
Se1/0 IPCP AUTHOR: Done.  Her address 11.11.11.1, we want 0.0.0.0
Se1/0 IPCP: O CONFACK [REQsent] id 1 len 10
Se1/0 IPCP:    Address 11.11.11.1 (0x03060B0B0B01)
Se1/0 IPCP: Event[Receive ConfReq+] State[REQsent to ACKsent]
Se1/0 CDPCP: I CONFREQ [REQsent] id 1 len 4
Se1/0 CDPCP: O CONFACK [REQsent] id 1 len 4
Se1/0 CDPCP: Event[Receive ConfReq+] State[REQsent to ACKsent]
Se1/0 IPCP: I CONFACK [ACKsent] id 1 len 10
Se1/0 IPCP:    Address 11.11.11.2 (0x03060B0B0B02)
Se1/0 IPCP: Event[Receive ConfAck] State[ACKsent to Open]
Se1/0 CDPCP: I CONFACK [ACKsent] id 1 len 4

Se1/0 CDPCP: Event[Receive ConfAck] State[ACKsent to Open]
Se1/0 IPCP: State is Open
Se1/0 CDPCP: State is Open
Se1/0 Added to neighbor route AVL tree: topoid 0, address 11.11.11.1
Se1/0 IPCP: Install route to 11.11.11.1
```

Checking status

HO	BR
	

```
HO#show ppp interface serial 1/0
PPP Serial Context Info
-------------------
Interface          : Se1/0
PPP Serial Handle: 0xBD000075
PPP Handle         : 0xCE000175
SSS Handle         : 0xC7000175
AAA ID             : 386
Access IE          : 0x32000172
SHDB Handle        : 0x0
State              : Up
Last State         : Binding
Last Event         : LocalTerm

PPP Session Info
----------------
Interface          : Se1/0
PPP ID             : 0xCE000175
Phase              : UP
Stage              : Local Termination
Peer Name          : BR
Peer Address       : 11.11.11.2
Control Protocols: LCP[Open] CHAP+ IPCP[Open] CDPCP[Open]
Session ID         : 370
AAA Unique ID      : 386
SSS Manager ID     : 0xC7000175
SIP ID             : 0xBD000075
PPP_IN_USE         : 0x11

Se1/0 LCP: [Open]
Our Negotiated Options
Se1/0 LCP:    AuthProto CHAP (0x0305C22305)
Se1/0 LCP:    MagicNumber 0xBC2F860D (0x0506BC2F860D)
Peer's Negotiated Options
Se1/0 LCP:    AuthProto CHAP (0x0305C22305)
Se1/0 LCP:    MagicNumber 0xBC2F7489 (0x0506BC2F7489)

Se1/0 IPCP: [Open]
Our Negotiated Options
Se1/0 IPCP:     Address 11.11.11.1 (0x03060B0B0B01)
Peer's Negotiated Options
Se1/0 IPCP:     Address 11.11.11.2 (0x03060B0B0B02)

Se1/0 CDPCP: [Open]
Our Negotiated Options
  NONE
Peer's Negotiated Options
  NONE
```

```
BR#show ppp interface serial 1/0
PPP Serial Context Info
-------------------
Interface          : Se1/0
PPP Serial Handle: 0x38000068
PPP Handle         : 0x81000168
SSS Handle         : 0x31000168
AAA ID             : 373
Access IE          : 0x4C000166
SHDB Handle        : 0x0
State              : Up
Last State         : Binding
Last Event         : LocalTerm

PPP Session Info
----------------
Interface          : Se1/0
PPP ID             : 0x81000168
Phase              : UP
Stage              : Local Termination
Peer Name          : HO
Peer Address       : 11.11.11.1
Control Protocols: LCP[Open] CHAP+ IPCP[Open] CDPCP[Open]
Session ID         : 358
AAA Unique ID      : 373
SSS Manager ID     : 0x31000168
SIP ID             : 0x38000068
PPP_IN_USE         : 0x11

Se1/0 LCP: [Open]
Our Negotiated Options
Se1/0 LCP:    AuthProto CHAP (0x0305C22305)
Se1/0 LCP:    MagicNumber 0xBC2F7489 (0x0506BC2F7489)
Peer's Negotiated Options
Se1/0 LCP:    AuthProto CHAP (0x0305C22305)
Se1/0 LCP:    MagicNumber 0xBC2F860D (0x0506BC2F860D)

Se1/0 IPCP: [Open]
Our Negotiated Options
Se1/0 IPCP:     Address 11.11.11.2 (0x03060B0B0B02)
Peer's Negotiated Options
Se1/0 IPCP:     Address 11.11.11.1 (0x03060B0B0B01)

Se1/0 CDPCP: [Open]
Our Negotiated Options
  NONE
Peer's Negotiated Options
  NONE
```

1.3.b [ii] PPPoE

PPP over Ethernet Server:
The first part is to configure the server. The device that will be aggregating multiple sessions.

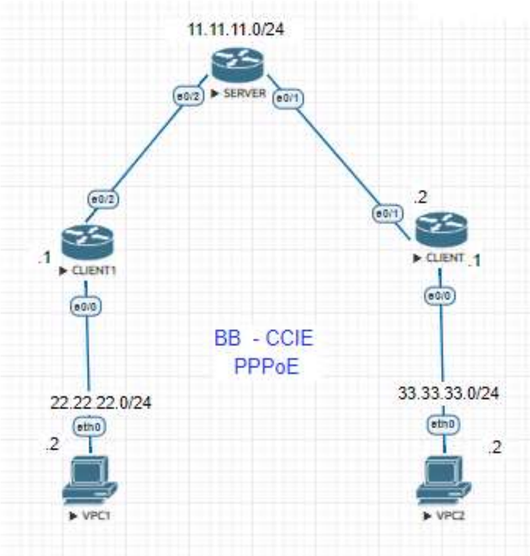
Basically from a DSL point of view the modems that are going to the access layer are the clients. Then the upstream links – normally ATTM PVCs – to the DSLAM are aggregated to the PPPoE server.

Define PPP interface:
-interface virtual-template (num)

Apply logical options:
-authentication, multilink IP address

Define BBA group:
-bba-group pope (name | global)
-virtual-template (num)

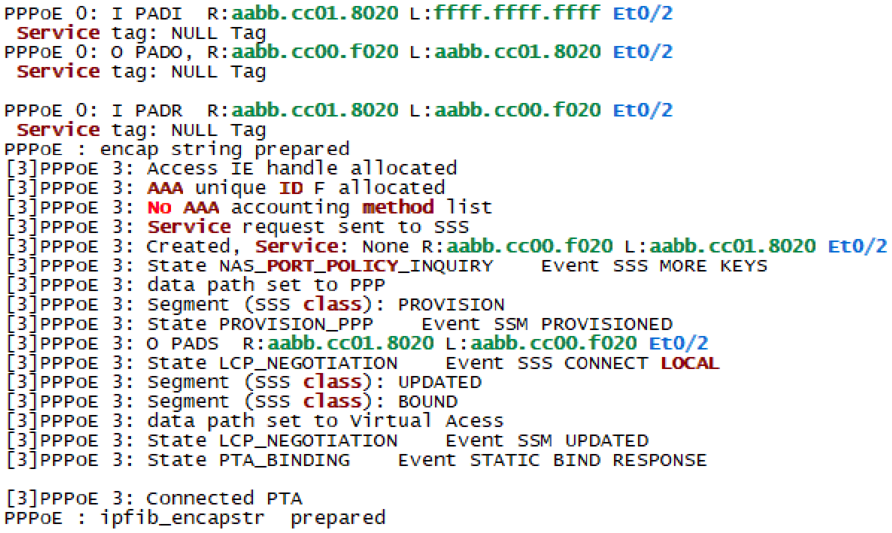
Bind to Link:
PPPoE enable group (name | global)



SERVER	CLIENT1	CLIENT2
! bba-group pppoe BBTEST virtual-template 12 ! ! interface Loopback12 ip address 11.11.11.1 255.255.255.0 ! interface Ethernet0/1 no ip address duplex auto pppoe enable group BBTEST ! interface Ethernet0/2 no ip address duplex auto pppoe enable group BBTEST ! ! interface Virtual-Template12 mtu 1492 ip unnumbered Loopback12 ip tcp adjust-mss 1452 peer default ip address pool BB-TEST ! ip local pool BB-TEST 11.11.11.10 11.11.11.11 !	! interface Ethernet0/2 no ip address shutdown duplex auto pppoe enable group global pppoe-client dial-pool-number 100 ! interface Dialer12 mtu 1492 ip address negotiated encapsulation ppp ip tcp adjust-mss 1452 dialer pool 100 !	! interface Ethernet0/1 no ip address shutdown duplex auto pppoe enable group global pppoe-client dial-pool-number 100 ! interface Dialer12 mtu 1492 ip address negotiated encapsulation ppp ip tcp adjust-mss 1452 dialer pool 100 !

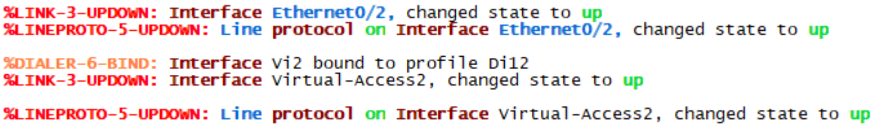
#

#Debug pppoe events - SERVER to see the connection



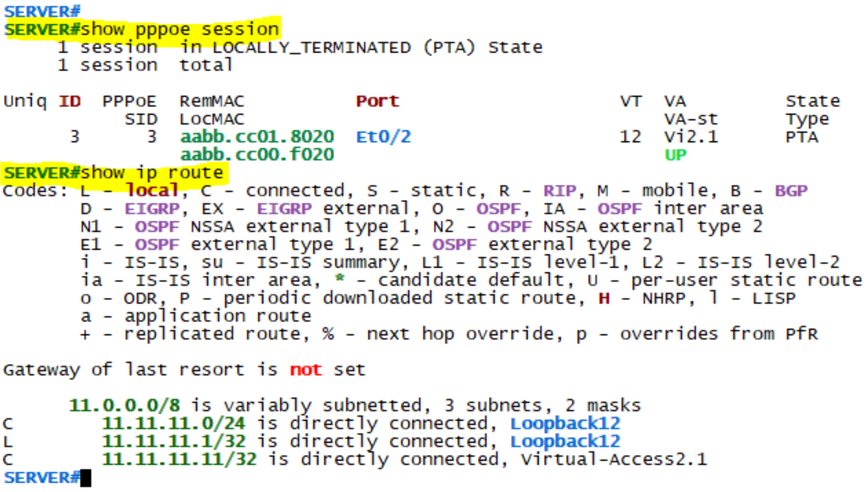
```
PPPoE 0: I PADI  R:aabb.cc01.8020 L:ffff.ffff.ffff Et0/2
 Service tag: NULL Tag
PPPoE 0: O PADO, R:aabb.cc00.f020 L:aabb.cc01.8020 Et0/2
 Service tag: NULL Tag

PPPoE 0: I PADR  R:aabb.cc01.8020 L:aabb.cc00.f020 Et0/2
 Service tag: NULL Tag
PPPoE : encap string prepared
[3]PPPoE 3: Access IE handle allocated
[3]PPPoE 3: AAA unique ID F allocated
[3]PPPoE 3: No AAA accounting method list
[3]PPPoE 3: Service request sent to SSS
[3]PPPoE 3: Created, Service: None R:aabb.cc00.f020 L:aabb.cc01.8020 Et0/2
[3]PPPoE 3: State NAS_PORT_POLICY_INQUIRY    Event SSS MORE KEYS
[3]PPPoE 3: data path set to PPP
[3]PPPoE 3: Segment (SSS class): PROVISION
[3]PPPoE 3: State PROVISION_PPP    Event SSM PROVISIONED
[3]PPPoE 3: O PADS  R:aabb.cc01.8020 L:aabb.cc00.f020 Et0/2
[3]PPPoE 3: State LCP_NEGOTIATION    Event SSS CONNECT LOCAL
[3]PPPoE 3: Segment (SSS class): UPDATED
[3]PPPoE 3: Segment (SSS class): BOUND
[3]PPPoE 3: data path set to Virtual Acess
[3]PPPoE 3: State LCP_NEGOTIATION    Event SSM UPDATED
[3]PPPoE 3: State PTA_BINDING    Event STATIC BIND RESPONSE

[3]PPPoE 3: Connected PTA
PPPoE : ipfib_encapstr  prepared
```

Client Side



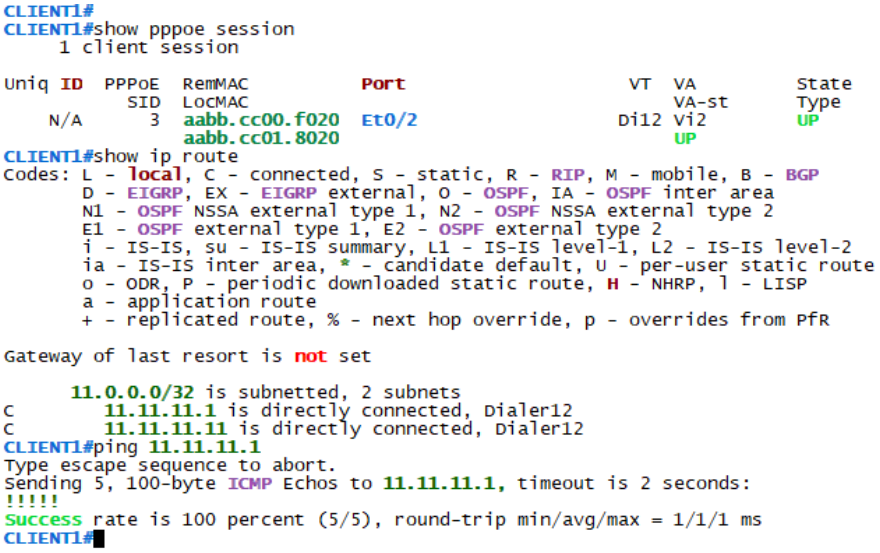
```
%LINK-3-UPDOWN: Interface Ethernet0/2, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/2, changed state to up

%DIALER-6-BIND: Interface Vi2 bound to profile Di12
%LINK-3-UPDOWN: Interface Virtual-Access2, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access2, changed state to up
```

Server Side check the PPPoE Sessions and ip route



```
SERVER#
SERVER#show pppoe session
     1 session  in LOCALLY_TERMINATED (PTA) State
     1 session  total

Uniq ID  PPPoE  RemMAC          Port                    VT  VA         State
           SID  LocMAC                                      VA-st      Type
      3      3  aabb.cc01.8020  Et0/2                   12  Vi2.1      PTA
                aabb.cc00.f020                              UP
SERVER#show ip route
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      11.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C        11.11.11.0/24 is directly connected, Loopback12
L        11.11.11.1/32 is directly connected, Loopback12
C        11.11.11.11/32 is directly connected, Virtual-Access2.1
SERVER#
```

Client side PPPoE session, ip route and ping test



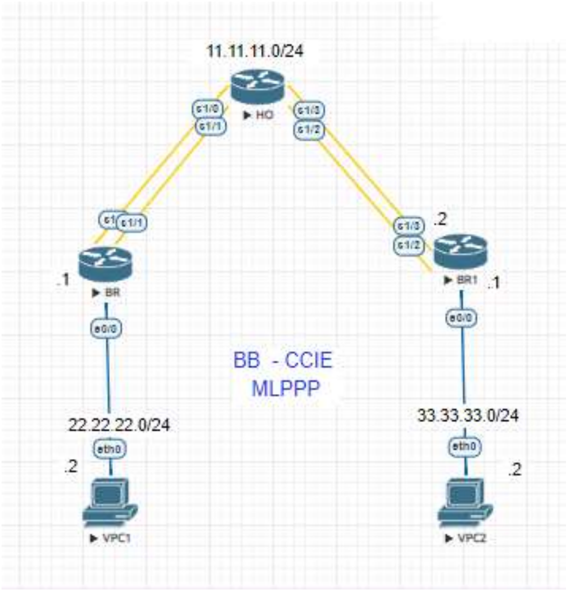
```
CLIENT1#
CLIENT1#show pppoe session
     1 client session

Uniq ID  PPPoE  RemMAC          Port                    VT  VA         State
           SID  LocMAC                                      VA-st      Type
    N/A      3  aabb.cc00.f020  Et0/2                 Di12  Vi2        UP
                aabb.cc01.8020                              UP
CLIENT1#show ip route
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      11.0.0.0/32 is subnetted, 2 subnets
C        11.11.11.1 is directly connected, Dialer12
C        11.11.11.11 is directly connected, Dialer12
CLIENT1#ping 11.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 11.11.11.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
CLIENT1#
```

1.3.b [iii] MLPPP

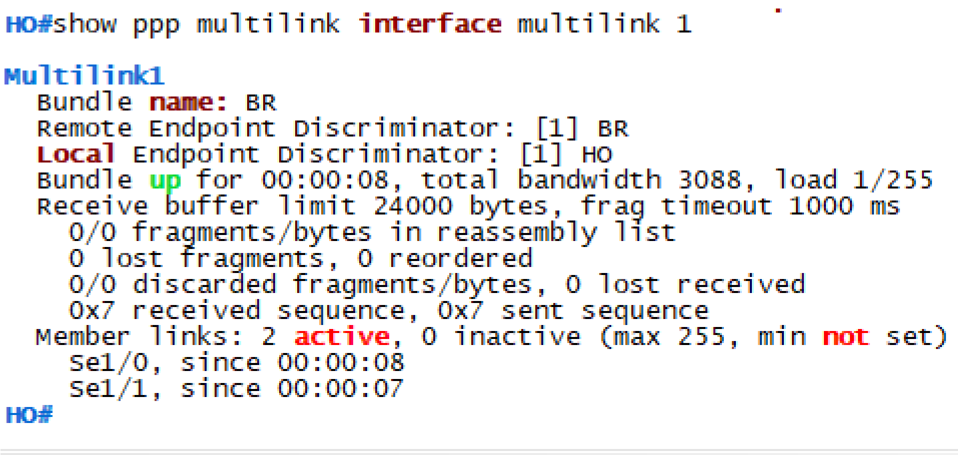
Multi-link PPP works the way that we think etherchannels should work when we first begin learning about link-aggregation: it chops packets up and uses each member interface to send a part of the packet.

Why do MLPPP bundles use this concept, where EtherChannels don’t?

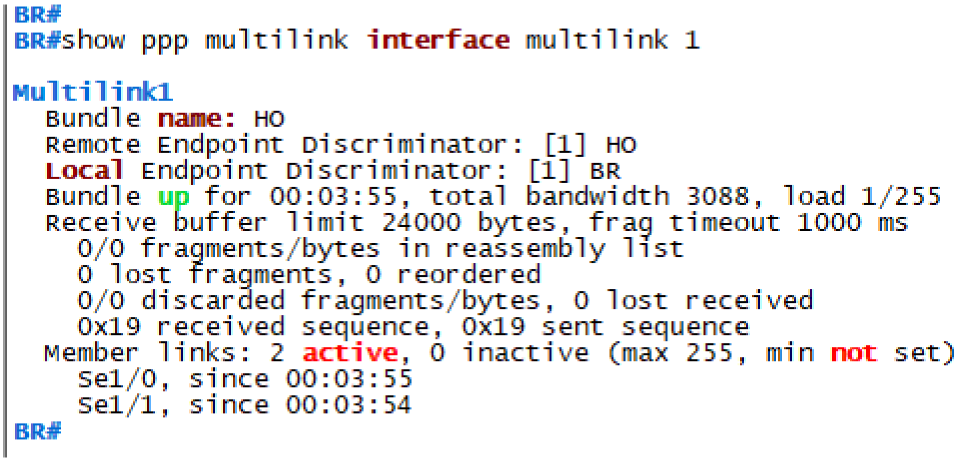
MLPPP connections are handled by the router’s CPU. Because Serial interface link speeds don’t usually come anywhere near that of Ethernet interfaces, the burden on a router’s CPU is much more manageable than this concept would be for an Etherchannel bundle.



HO	BR	BR1
config t ! hostname HO ! interface multilink1 ip address 11.11.11.1 255.255.255.252 ppp multilink ppp multilink group 1 ! interface serial 1/0 no ip address encapsulation ppp ppp multilink ppp multilink group 1 serial restart-delay 0 no shut ! interface serial 1/1 no ip address encapsulation ppp ppp multilink ppp multilink group 1 serial restart-delay 0 no shut ! end	config t ! hostname BR ! interface multilink1 ip address 11.11.11.2 255.255.255.252 ppp multilink ppp multilink group 1 ! interface serial 1/0 no ip address encapsulation ppp ppp multilink ppp multilink group 1 serial restart-delay 0 no shut ! interface serial 1/1 no ip address encapsulation ppp ppp multilink ppp multilink group 1 serial restart-delay 0 no shut ! End	config t ! hostname BR ! interface multilink1 ip address 11.11.11.4 255.255.255.252 ppp multilink ppp multilink group 1 ! interface serial 1/2 no ip address encapsulation ppp ppp multilink ppp multilink group 1 serial restart-delay 0 no shut ! interface serial 1/3 no ip address encapsulation ppp ppp multilink ppp multilink group 1 serial restart-delay 0 no shut ! End

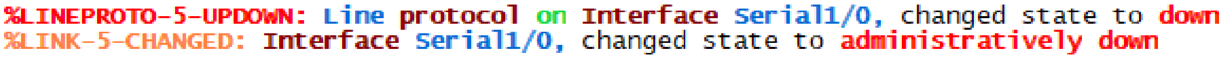


```
HO#show ppp multilink interface multilink 1

Multilink1
  Bundle name: BR
  Remote Endpoint Discriminator: [1] BR
  Local Endpoint Discriminator: [1] HO
  Bundle up for 00:00:08, total bandwidth 3088, load 1/255
  Receive buffer limit 24000 bytes, frag timeout 1000 ms
    0/0 fragments/bytes in reassembly list
    0 lost fragments, 0 reordered
    0/0 discarded fragments/bytes, 0 lost received
    0x7 received sequence, 0x7 sent sequence
  Member links: 2 active, 0 inactive (max 255, min not set)
    Se1/0, since 00:00:08
    Se1/1, since 00:00:07
HO#
```

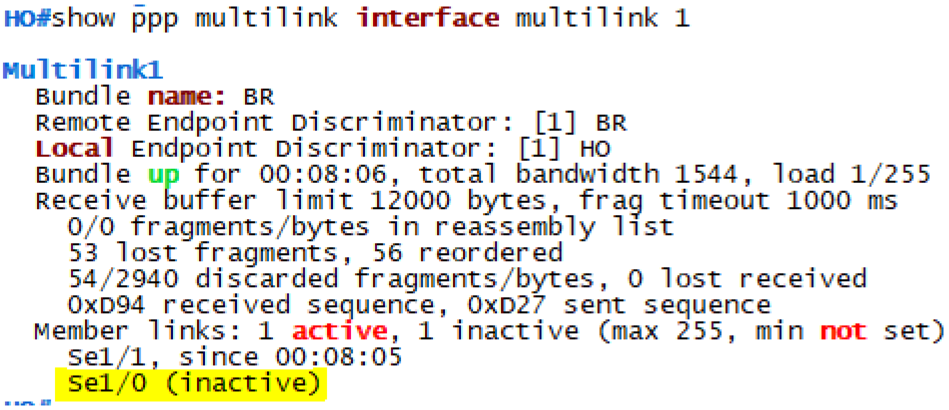


```
BR#
BR#show ppp multilink interface multilink 1

Multilink1
  Bundle name: HO
  Remote Endpoint Discriminator: [1] HO
  Local Endpoint Discriminator: [1] BR
  Bundle up for 00:03:55, total bandwidth 3088, load 1/255
  Receive buffer limit 24000 bytes, frag timeout 1000 ms
    0/0 fragments/bytes in reassembly list
    0 lost fragments, 0 reordered
    0/0 discarded fragments/bytes, 0 lost received
    0x19 received sequence, 0x19 sent sequence
  Member links: 2 active, 0 inactive (max 255, min not set)
    Se1/0, since 00:03:55
    Se1/1, since 00:03:54
BR#
```

Lets test brining one link down from HO

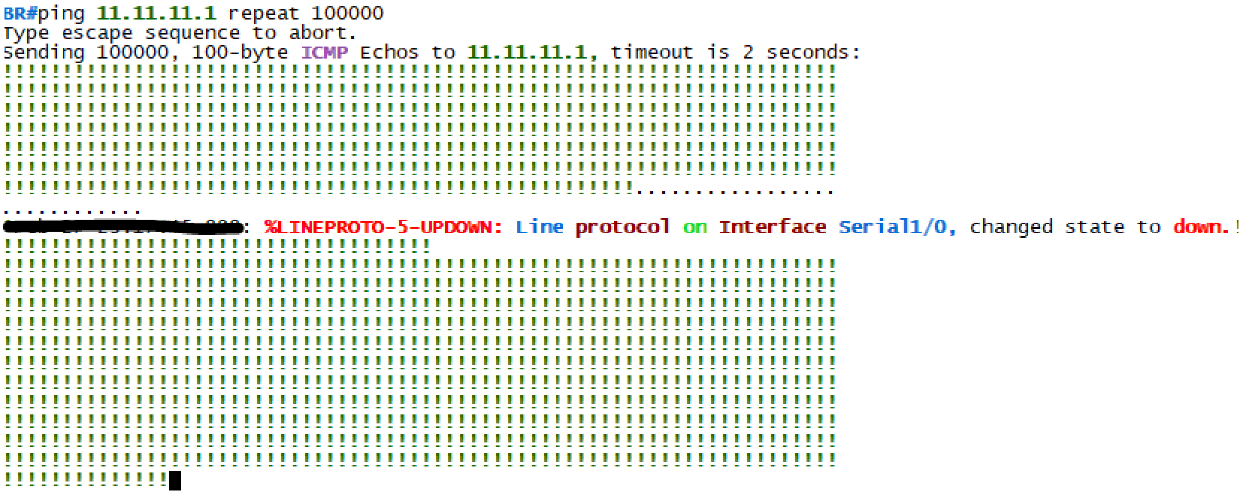


```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0, changed state to down
%LINK-5-CHANGED: Interface Serial1/0, changed state to administratively down
```



```
HO#show ppp multilink interface multilink 1

Multilink1
  Bundle name: BR
  Remote Endpoint Discriminator: [1] BR
  Local Endpoint Discriminator: [1] HO
  Bundle up for 00:08:06, total bandwidth 1544, load 1/255
  Receive buffer limit 12000 bytes, frag timeout 1000 ms
    0/0 fragments/bytes in reassembly list
    53 lost fragments, 56 reordered
    54/2940 discarded fragments/bytes, 0 lost received
    0xD94 received sequence, 0xD27 sent sequence
  Member links: 1 active, 1 inactive (max 255, min not set)
    Se1/1, since 00:08:05
    Se1/0 (inactive)
```

Client side ping repeat count high



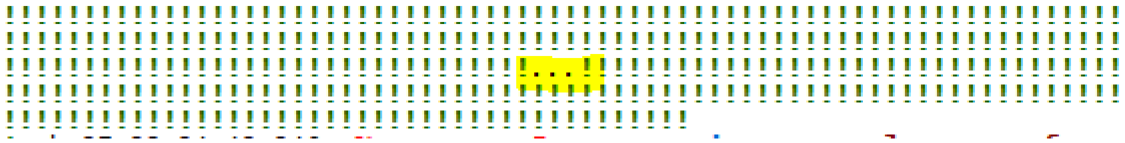
```
BR#ping 11.11.11.1 repeat 100000
Type escape sequence to abort.
Sending 100000, 100-byte ICMP Echos to 11.11.11.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.................
............
[redacted]: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0, changed state to down.!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!
```

So It took 100Seconds to recovery from the link

Let’s change the keep alive to 5seconds both the side

interface Multilink1
keepalive 1 5

here we lost only 3 pings



```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!...!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

1.4 Troubleshooting layer 2 technologies

1.4.a Use IOS troubleshooting tools

some of the best tools you can use are the following:

- ping
- telnet
- trace
- ssh
- debug ip packet
- debug frame

•Ping – A command utility used to test IP connectivity between hosts. It sends out requests (ICMP echo requests) to a specific host and expects a response. The receiving host responds with an ICMP echo response. If the response is received, IP connectivity exists. A successful ping displays the round trip time (RTT) which is a measure of network performance. Can be used from a router/switch as well as most servers.
•Traceroute – Uses TTL timeouts with ICMP error messages to find the path a packet takes through an internetwork. Can be used from a router/switch as well as most servers. Note: Windows hosts utilize the command tracert.
•Telnet – Provides capability to remotely access another device (i.e. computer, server, etc.). Telnet can also be used to test services running on a remote system by telnetting to service port in question and detemining if a response is received. For example, telnetting to port 80 to test if an HTTP server is up.
•Secure Shell (SSH) – Similar to telnet with the exception that it is secure.

1.4.a [i] debug, conditional debug

Debug is a troubleshooting command that is available from the privileged exec mode (of Cisco IOS). This command can be used to display information about various router operations and the related traffic generated or received by the router, as well as any error messages. This tool is very useful and informative, but you must be aware of the following facts regarding its use: Debug is treated as a very high priority task. It can consume a significant amount of resources, and the router is forced to process-switch the packets being debugged. Debug must not be used as a monitoring tool—it is meant to be used for a short period of time and as a troubleshooting tool. By using it you discover significant facts about the working and faulty software and/or hardware components. The following is a list of recommendations on proper usage of the debug command.

Conditional debug (Using an Access List with Debug)

With the debug ip packet detail command, you have the option to enter the name or number of an access list. Doing that causes the debug command to get focused only on those packets satisfying (permitted by) the access list's statements

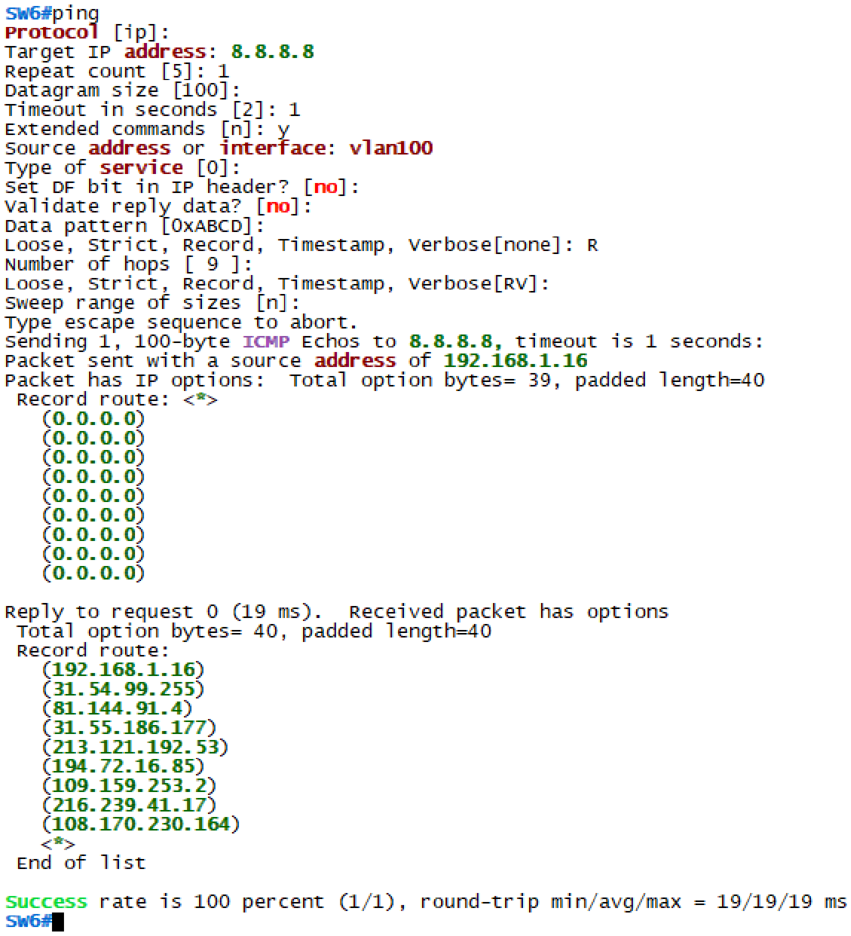
Conditional debugging gives you the ability to specify certain conditions that must be met to trigger a debugging message, such as an interface, IP address, MAC address, or application. As you might imagine, this can help filter your debugging greatly! To turn them on, simply use the privileged exec mode **debug condition** *condition* command. For example, to debug only on interface Gig 0/0 you would use the **debug condition interface Gig 0/0** command.

1.4.a [ii] ping, traceroute with extended options

The extended ping command is invoked from the privileged exec mode by typing ping and pressing Enter. The following arguments can be modified:

- Protocol [ip] – specify the protocol, such as appletalk, clns, ip, novell, apollo, vines, decnet, or xns. The default is ip.
- Target IP address – specify the IP address or the hostname of the host to ping.
- Repeat count – specify the number of ping packets that will be sent to the destination address. 5 by default.
- Datagram size – specify the size of the ping packet (in bytes). The default is 100 bytes.
- Timeout in seconds – specify the timeout interval. The default is 2 seconds. The echo reply needs to be received before the timeout expires in order for ping to be successful.
- Extended commands – specify whether or not a series of additional commands will appear. The default is no. If you type yes additional arguments will be shown.
- Source address or interface – specify the interface or the IP address of the router to use as the source address for the ping packets.
- Type of service – specifies the Type of Service (ToS). This is the Internet service’s quality selection. The default is 0.
- Set DF bit in IP header? – specify whether or not the Don’t Fragment (DF) bit will be set on the ping packet. If yes is entered, the Don’t Fragment option does not allow the packet to be fragmented. The default is no.
- Validate reply data? – specify whether or not to validate the reply data. The default is no.
- Data pattern – specify the data pattern. Data patterns are used to troubleshoot framing errors and clocking problems on serial lines. The default is [0xABCD].
- Loose, Strict, Record, Timestamp, Verbose – specify the IP header options.
- Sweep range of sizes – specify the sizes of the ping echo packets that are sent. This parameter is used to determine the minimum sizes of the MTUs configured on the nodes along the path to the destination address. The default is no.

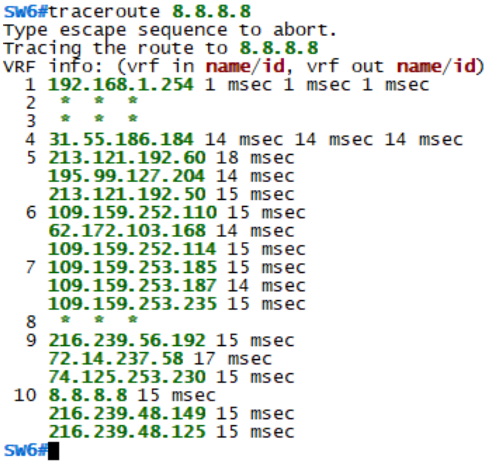
Extended **PING**:



```
SW6#ping
Protocol [ip]:
Target IP address: 8.8.8.8
Repeat count [5]: 1
Datagram size [100]:
Timeout in seconds [2]: 1
Extended commands [n]: y
Source address or interface: vlan100
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: R
Number of hops [ 9 ]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 8.8.8.8, timeout is 1 seconds:
Packet sent with a source address of 192.168.1.16
Packet has IP options:  Total option bytes= 39, padded length=40
 Record route: <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)

Reply to request 0 (19 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
   (192.168.1.16)
   (31.54.99.255)
   (81.144.91.4)
   (31.55.186.177)
   (213.121.192.53)
   (194.72.16.85)
   (109.159.253.2)
   (216.239.41.17)
   (108.170.230.164)
   <*>
 End of list

Success rate is 100 percent (1/1), round-trip min/avg/max = 19/19/19 ms
SW6#
```

Traceroute:

- number of miliseconds – the round-trip time in milliseconds.
- * – the probe has timed out.
- A – administratively prohibited (for example, with an access-list).
- Q – source quench (the destination is too busy).
- I – user interrupted test.
- U – port is unreachable.
- N – the network is unreachable.
- P – the protocol is unreachable.
- T – timeout.
- ? – unknown packet type.

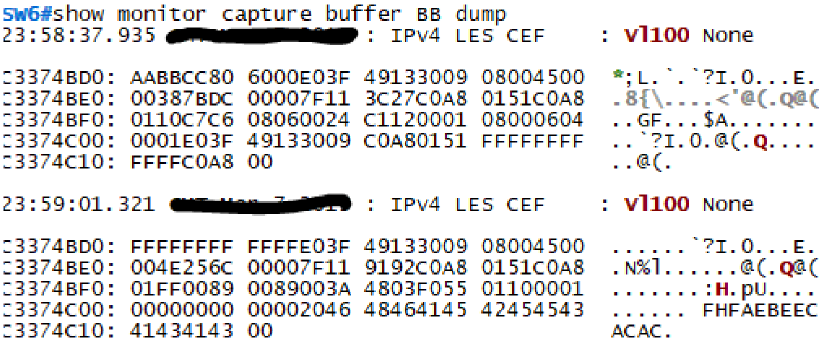


```
SW6#traceroute 8.8.8.8
Type escape sequence to abort.
Tracing the route to 8.8.8.8
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.1.254 1 msec 1 msec 1 msec
  2  *  *  *
  3  *  *  *
  4 31.55.186.184 14 msec 14 msec 14 msec
  5 213.121.192.60 18 msec
    195.99.127.204 14 msec
    213.121.192.50 15 msec
  6 109.159.252.110 15 msec
    62.172.103.168 14 msec
    109.159.252.114 15 msec
  7 109.159.253.185 15 msec
    109.159.253.187 14 msec
    109.159.253.235 15 msec
  8  *  *  *
  9 216.239.56.192 15 msec
    72.14.237.58 17 msec
    74.125.253.230 15 msec
 10 8.8.8.8 15 msec
    216.239.48.149 15 msec
    216.239.48.125 15 msec
SW6#
```

1.4.a [iii] Embedded packet capture

When enabled, the router captures the packets sent and received. The packets are stored within a buffer in DRAM and are thus not persistent through a reload. Once the data is captured, it can be examined in a summary or detailed view on the router. In addition, the data can be exported as a packet capture (PCAP) file to allow for further examination. **The tool is configured in exec mode and is considered a temporary assistance tool. As a result, the tool configuration is not stored within the router configuration and will not remain in place after a system reload.**

SW6#monitor capture point ip cef BB vlan 100 both



```
SW6#show monitor capture buffer BB dump
23:58:37.935 [redacted] : IPv4 LES CEF     : Vl100 None

C3374BD0: AABBCC80 6000E03F 49133009 08004500  *;L.`.`?I.0...E.
C3374BE0: 00387BDC 00007F11 3C27C0A8 0151C0A8  .8{\....<'@(.Q@(
C3374BF0: 0110C7C6 08060024 C1120001 08000604  ..GF...$A.......
C3374C00: 0001E03F 49133009 C0A80151 FFFFFFFF  ..`?I.0.@(.Q....
C3374C10: FFFFC0A8 00                          ..@(.

23:59:01.321 [redacted] : IPv4 LES CEF     : Vl100 None

C3374BD0: FFFFFFFF FFFFE03F 49133009 08004500  ......`?I.0...E.
C3374BE0: 004E256C 00007F11 9192C0A8 0151C0A8  .N%l......@(.Q@(
C3374BF0: 01FF0089 0089003A 4803F055 01100001  .......:H.pU....
C3374C00: 00000000 00002046 48464145 42454543  ...... FHFAEBEEC
C3374C10: 41434143 00                          ACAC.
```

1.4.b Apply troubleshooting methodologies

1.4.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]

Define the problem: Utilize an SNMP network monitoring tool. There are numerous products available from various vendors.

Gather Information: There are several things that can be utilize here. Some are simple CLI “show” and “debug” commands.

Analyze: You should have a baseline for your network to compare traffic when a problem surfaces with your baseline. This could assist with determining the problem. One method of baselining your network is by using Cisco Netflow Accounting.

Test the hypothesis: Here you should have very good Configuration Management (CM) which would allow you to easily test changes and rollback to the previous configuration of the changes do not work.

1.4.b [ii] Design and implement valid solutions according to constraints

1.4.b [iii] Verify and monitor resolution

1.4.c Interpret packet capture

1.4.c [i] Using wireshark trace analyzer

1.4.c [ii] Using IOS embedded packet capture

************** Intention Left Blank *********************************

2.0 Layer 3 Technologies

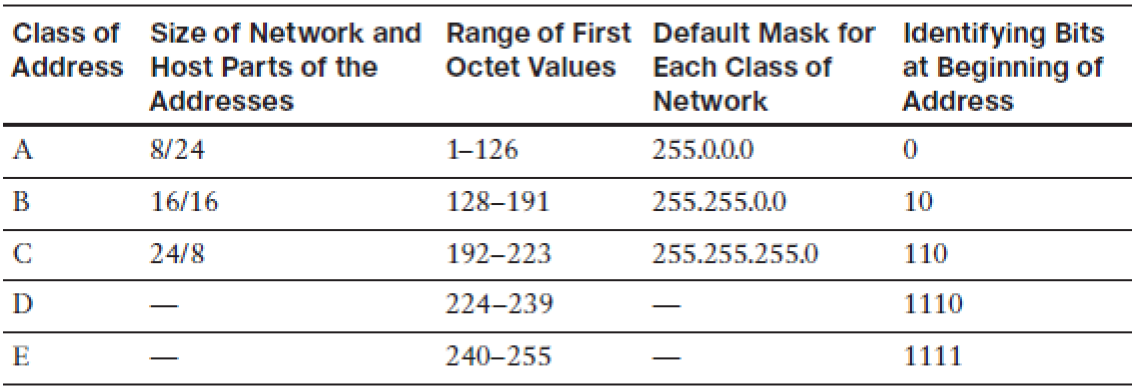
40%

2.1 Addressing technologies

2.1.a Identify, implement and troubleshoot IPv4 addressing and sub-netting
2.1.a [i] Address types, VLSM

IP Addressing and Subnetting
You need a postal address to receive letters; similarly, Computers must use an IP address to be able to send and receive data using the TCP/IP protocols. Just as the postal service dictates the format and meaning of a postal address to aid the efficient delivery of mail, the TCP/IP protocol suite imposes some rules about IP address assignment so that routers can efficiently forward packets between IP hosts. This chapter begins with coverage of the format and meaning of IP addresses, with required consideration for how they are grouped to aid the routing process.

Classfull :



Class of Address	Size of Network and Host Parts of the Addresses	Range of First Octet Values	Default Mask for Each Class of Network	Identifying Bits at Beginning of Address
A	8/24	1–126	255.0.0.0	0
B	16/16	128–191	255.255.0.0	10
C	24/8	192–223	255.255.255.0	110
D	—	224–239	—	1110
E	—	240–255	—	1111

VLSM
A Variable Length Subnet Mask (VLSM) is a means of allocating IP addressing resources to subnets according to their individual need rather than some general network-wide rule (i.e. Class A, B or C). Of the IP routing protocols supported by Cisco, OSPF, IS-IS, BGP, and EIGRP support “classless” or VLSM routes.

2.1.a [ii] ARP

ARP provides a dynamic mapping between IPv4 addresses and the hardware addresses (used by various technologies) for devices on the same network segment.

Proxy ARP: Proxy ARP allows a system to answer ARP requests for a diffent host. Proxy ARP is not commonly used and should be avoided where possible.

Gratuitous ARP : Gratuitous ARP occurs when a host sends an ARP request looking for its own address.

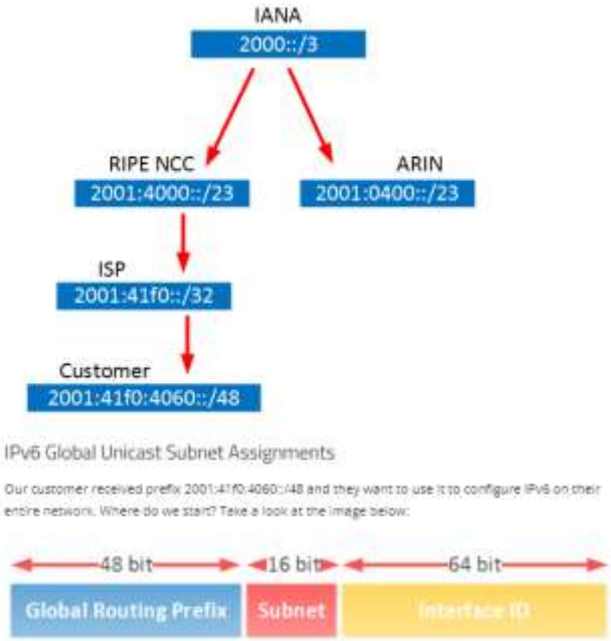
Benefits are as follows:
a. It lets a host determine if another host is using the same IPv4 address i.e. it will get an unexpected reply.
b. If a host has changed its hardware/IPv4 address, a gratuitous ARP request will inform other hosts of the change and they will update their cache.

2.1.b Identify, implement and troubleshoot IPv6 addressing and sub-netting

2.1.b [i] Unicast, multicast

Unicast

Unicast IPv6 addresses are similar to unicast IPv4 addresses. These are meant to configure on one interface so that you can send and receive IPv6 packets

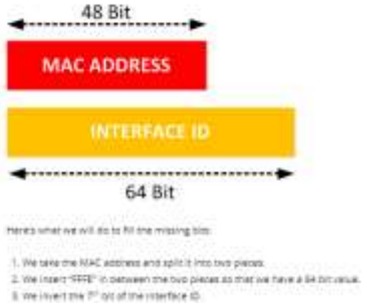


IPv6 Global Unicast Subnet Assignments

Our customer received prefix 2001:41f0:4060::/48 and they want to use it to configure IPv6 on their entire network. Where do we start? Take a look at the image below:

2.1.b [ii] EUI-64

EUI-64 (Extended Unique Identifier) is a method we can use to automatically configure IPv6 host addresses. An IPv6 device will use the MAC address of its interface to generate a unique 64-bit interface ID. However, a MAC address is 48 bit and the interface ID is 64 bit. What are we going to do with the missing bits?



Here's what we will do to fill the missing bits:

1. We take the MAC address and split it into two pieces.
2. We insert "FFFE" in between the two pieces so that we have a 64 bit value.
3. We invert the 7th bit of the interface ID.

2.1.b [iii] ND, RS/RA

Neighbor Discovery protocol is in fact an umbrella term for many interrelated subprotocols and mechanisms whose main responsibilities include the following:

Resolution of IPv6 addresses into MAC addresses of neighboring hosts

Duplicate address detection
Router discovery
Stateless address auto-configuration
Host redirection

Router Solicitation:

Upon an interface of a node being enabled, Router Solicitation messages can be used to request all routers on the same local link to send Router Advertisements immediately, rather than waiting until the next periodically scheduled advertisement.

Router Advertisement:

Routers send Route Advertisement messages out of each IPv6-enabled interface, informing attached hosts of their presence. Router Advertisement messages may also contain vital information allowing the hosts to automatically configure their IPv6 stack: the IPv6 prefix of the particular network, the first-hop MTU, the router's MAC address (allowing hosts to learn about the router's MAC address right away), and even the list of DNS domains and DNS servers. Using the information in a received Router Advertisement, a host can automatically configure its IPv6 stack and obtain IPv6 connectivity without the need for any dedicated DHCP service on a network. Router Advertisements are sent periodically, though infrequently, and they are also sent immediately as a response to a Router Solicitation message originated by an end host.

2.1.b [iv] Autoconfig/SLAAC temporary addresses [RFC4941]

IPv6 Stateless Address Autoconfiguration (SLAAC)
Utilizes 2 ICMPv6 messages to communicate the prefix to clients:
Router solicitation
Router advertisement

SLAAC provides the ability to address a host based on a network prefix that is advertised from a local network router via Router Advertisements (RA). RA messages are sent by default by most IPV6 routers; these messages are sent out periodically by the router and include information including:

- One or more IPv6 prefixes (Link-local scope)
- Prefix lifetime information
- Flag information
- Default device information (Default router to use and its lifetime)

Temporary Addresses:
What issue was quickly identified with using SLAAC and EUI64?
An end user's device could be tracked accross multiple internet service providers because they would always generate the same EUI64 address.
RFC4941, or temporary addresses or privay addresses is a technique to NOT use the MAC address of your computer for the IPv6 address in SLAAC.
Temporary addresses use 64 bits from an MD5 hash to generate a unique address.
These addresses expire and are changed often (average of once a day) on a system so it's not uncommon to see multiple IPv6 addresses on a system.
RFC4941 relies on IPV6's DAD mechanism for avoiding duplicate addresses.
This feature is usually enabled by default on windows, linux, and mac.

2.1.b [v] Global prefix configuration feature

The IPv6 generic prefix feature simplifies network renumbering and allows for automated prefix definition. An IPv6 generic (or general) prefix (for example, /48) holds a short prefix, based on which a number of longer, more-specific prefixes (for example, /64) can be defined. When the general prefix is changed, all of the more-specific prefixes based on it will change, too.

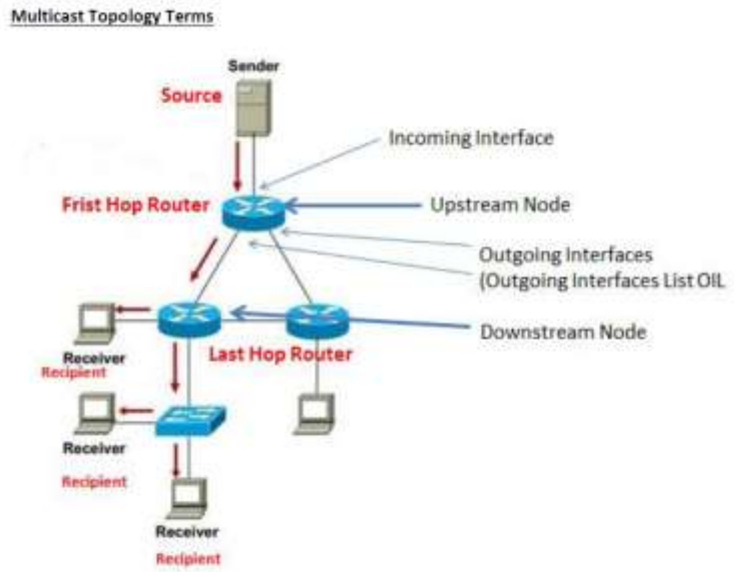
Finding Feature Information
Information About IPv6 Generic Prefix
How to Configure IPv6 Generic Prefix
Additional References
Feature Information for IPv6 Generic Prefix

2.2 Layer 3 Multicast

There are three types of traffic that we can choose from for our networks:
Unicast
Broadcast
Multicast

If you want to send a message from one source to one destination, we use **unicast**.
If you want to send a message from one source to everyone, we use **broadcast**.
if we want to send a message from one source to a group of receivers - we use **multicast**.

IGMP hosts can tell routers they want to receive multicast traffic.
IGMP snooping so the switch knows where to forward multicast traffic.
Multicast routing: we need a protocol like PIM that can route multicast traffic.
Multicast has many advantages; the main advantage is the scalability compared to unicast traffic.



2.2.a Troubleshoot reverse path forwarding

An RPF check examines the source address of an incoming packet and checks it against the router's unicast routing table to see what interface should be used to get back to the source network.
If the incoming multicast packet is using that interface, the RPF check passes, and the packet is forwarded.
If the multicast packet comes in a different interface, the RPF check fails, and the packet is discarded.

Strict RPF—Using the rx keyword, the router checks to see if the matching route uses an outgoing interface that is the same interface on which the packet was received.
If not, the packet is discarded.

Loose RPF—Using the any keyword, the router checks for any route that can be used to reach the source IP address.

2.2.a [i] RPF failure

What is the RPF check, or Reverse Path Forwarding check, is for multicast. PIM is known as Protocol Independent Multicast routing because it does not exchange its own information about the network topology. Instead it relies on the accuracy of an underlying unicast routing protocol like OSPF or EIGRP to maintain a loop free topology. When a multicast packet is received by a router running PIM the device first looks at what the source IP is of the packet. Next the router does a unicast lookup on the source, as in the “show ip route w.x.y.z”, where “w.x.y.z” is the source. Next the outgoing interface in the unicast routing table is compared with the interface in which the multicast packet was received on. If the incoming interface of the packet and the outgoing interface for the route are the same, it is assumed that the packet is not looping, and the packet is candidate for forwarding. If the incoming interface and the outgoing interface are *not* the same, a loop-free path can not be guaranteed, and the RPF check fails. All packets for which the RPF check fails are dropped.

2.2.a[ii] RPF failure with tunnel interface

2.2.b Implement and troubleshoot IPv4 protocol independent multicast

How multicast work
1-source app send UDP multicast traffic with "group" destination address
2-interested receivers "join" group address by signaling routers
3-routers communicate to build loop free "tree" from sender to receiver
4-parts of the network without receivers will not receive traffic for that group

Protocols used in Multicast
Communications between Host & Router: IGMPv1, 2(default) and 3
Communications between Router to Router: PIM in IPv4, MLD in IPv6
Communications between Switch & Router: IGMP Snooping or CGMP

Multicast Main Components:
Control plane (routing) : IGMP , PIM
Data plane (forwarding) : RPF , Mroute table

2.2.b [i] PIM dense mode, sparse mode, sparse-dense mode

PIM Modes:
Dense push mode use source tree
Sparse pull mode use source tree or shared tree

PIM Dense Mode PIM-DM:
Flood multicast traffic for all groups out all multicast-enabled interfaces. Routers which determine they have no clients interested in receiving the traffic then send prune messages up toward the source, requesting that the flow of multicast traffic downstream be stemmed.

PIM Sparse Mode PIM-SM:
Multicast traffic from a source isn't forwarded to group members. When a member somewhere in the network decides it wants to receive traffic for a group, it sends a join request to its nearest router. The join request is propagated up the multicast tree toward the source router. Upon receiving the join request, the source router begins forwarding multicast traffic for the group out the appropriate interface(s).

PIM-DM

RFC 3973 , push model

All traffic flooded through LAN not centralized Router such as RP taking care of receivers requests also routers have no receivers prune (unjoin) the link , this mode had no scale because of flooding and (S,G) creation

1-Routers discover PIM neighbors using 224.0.0.13

2-routers once receive traffic insert (S,G) into routing table :

Incoming interface is attached to server , OIL is all other interfaces

3-flood all multicast traffic

4-prune unwanted traffic , Prune used to tell upstream neighbor to stop sending tarffic for (S.G)

Prune occurs if :

- Multicast feed fails RPF check
- no downstream neighbors or receivers
- downstream neighbors already sent prune

Note when traffic flow stop , (S,G) remains in table .

PIM Dense Mode use some features to maintain Multicast routing Table :

Graft

What happen if i pruned (S.G) but then receive IGMP join message from receiver ?

Graft messages used in dense mode to unprune

Assert

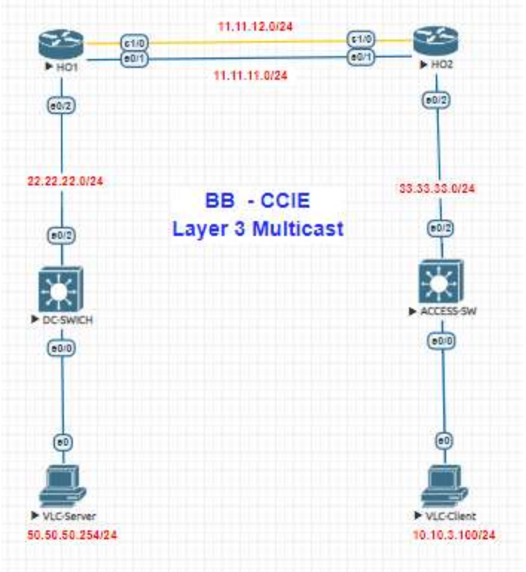
To prune duplicate multicast feed transmissions

-winner is loweset metric to source or if equal teh highest ip address

State refersh

Once (S.G) is pruned , traffic re-flooded about every 3 minutes

So ,state refresh is keep alive for prune state

Dense Mode	Basic Configs	Final Config
 11.11.12.0/24 e1/0 e0/1 e1/0 e0/1 HO1 11.11.11.0/24 HO2 e0/2 e0/2 22.22.22.0/24 33.33.33.0/24 BB - CCIE Layer 3 Multicast e0/2 e0/2 DC-SWICH ACCESS-SW e0/0 e0/0 e0 e0 VLC-Server VLC-Client 50.50.50.254/24 10.10.3.100/24	L3-Multicast.txt	pim-dens-mode-full-config.txt

Cisco Good reference Start with:

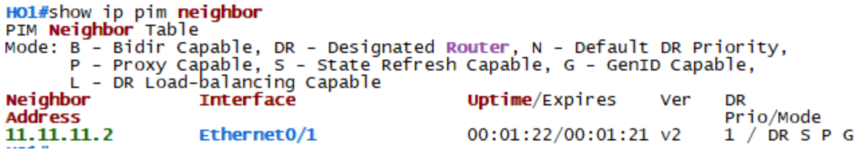
https://www.cisco.com/c/en/us/support/docs/ip/ip-multicast/9356-48.html?referring_site=bodynav

Now we enable PIM DensMode and Test it. (In the Real world I have not come across any setup who is using dense mode)

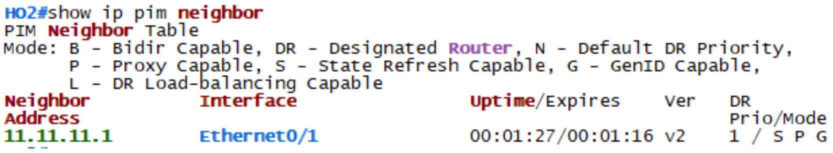
Enable pim dense mode on all the interface

Ip multicast-routing - to enable pim global config
!
Ip pim dense-mode – enable dense mode interface specific
Ip igmp join-group x.x.x.x – to join the source group.

HO1 and HO2 enabled Multicast routing and pim dense-mode :

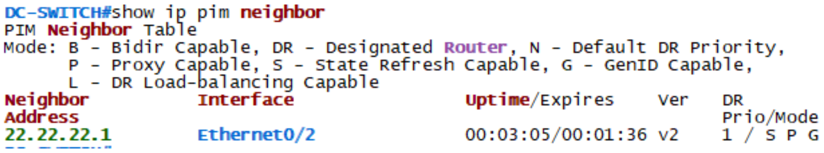


```
HO1#show ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable,
      L - DR Load-balancing Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                            Prio/Mode
11.11.11.2        Ethernet0/1              00:01:22/00:01:21 v2    1 / DR S P G
```

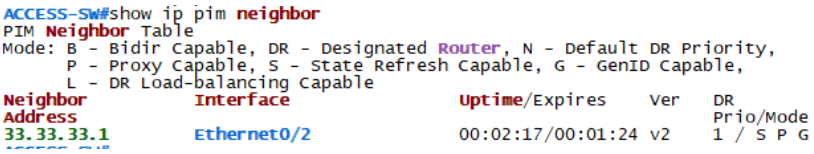


```
HO2#show ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable,
      L - DR Load-balancing Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                            Prio/Mode
11.11.11.1        Ethernet0/1              00:01:27/00:01:16 v2    1 / S P G
```

Now Enable DC-Switch (where the Source of VLC Player multicasting the Video)
Enable Access Switch, where the Client receiving the video

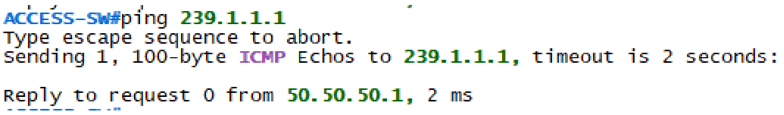


```
DC-SWITCH#show ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable,
      L - DR Load-balancing Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                            Prio/Mode
22.22.22.1        Ethernet0/2              00:03:05/00:01:36 v2    1 / S P G
```



```
ACCESS-SW#show ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable,
      L - DR Load-balancing Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                            Prio/Mode
33.33.33.1        Ethernet0/2              00:02:17/00:01:24 v2    1 / S P G
```

Lets ping 239.1.1.1 from Access Switch.



```
ACCESS-SW#ping 239.1.1.1
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 239.1.1.1, timeout is 2 seconds:

Reply to request 0 from 50.50.50.1, 2 ms
```

Lets enable video on VLC Server and check the same Video Receive on VLC Player.

Here’s how to do it:

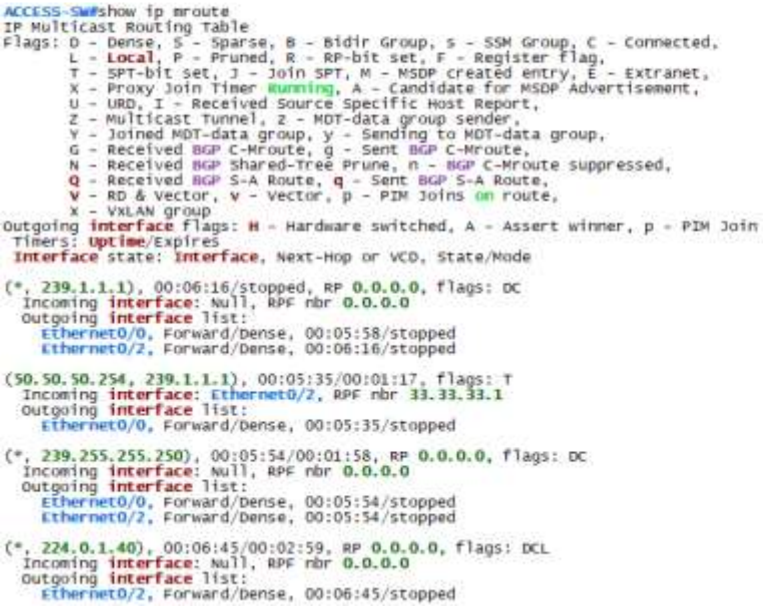
1. Open VLC Media Player, In the Media menu, choose “Open Network Stream”
2. In the Open Media dialog - go to File tab, click “add” and choose the file you want to stream and click “Open”
3. At the bottom, click the “Play" pull down menu and select "Stream” button
4. This opens the “Stream Output” dialog showing the source file you have chosen. Click Next to set destination.
5. In “Destinations”, choose “RTP /MPEG Transport Stream” and click the “Add” button

6. In the “Address” box, enter the required multicast address (eg 239.1.1.1) and set the port (or leave default at 5004) - Click Next.
7. In transcoding options, Make suer check the box "Activate Transcoding". Profile Select " Vodeo for Android SD Low" (choose the requirement you like for testing i have choosen this)
Click "Next"
8. Option Setup "click the check box " Stream all elementary Streams" in the "Generated Steam output String" after mux=ts,ttl=20 (by default it will be 1 - so client will be more than 1 hop away so this should be set based on the requirement)
then Click "Next and click stream".

To view the stream From client Side, open another instance of VLC media player (try it on the same PC before trying it over the network)

1. Choose Media/Open Network Stream
2. In address, enter rtp://232.1.1.1:5004 – choose the correct address and port you entered when setting up the stream.
3. Click “Play”

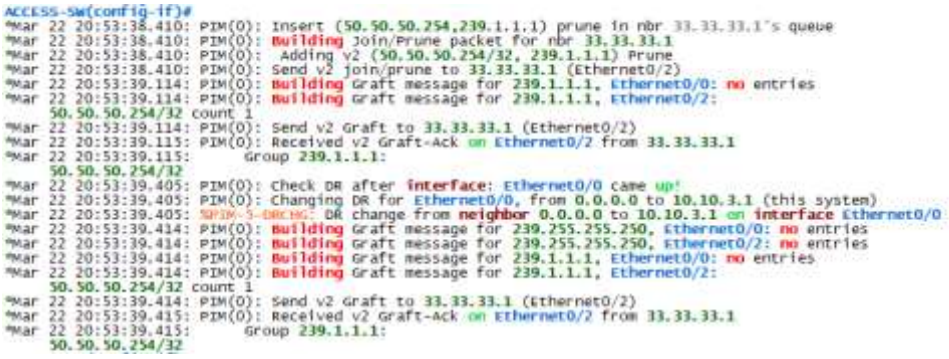
Check ip mroute



```
ACCESS-SW#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:06:16/stopped, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Dense, 00:05:58/stopped
    Ethernet0/2, Forward/Dense, 00:06:16/stopped

(50.50.50.254, 239.1.1.1), 00:05:35/00:01:17, flags: T
  Incoming interface: Ethernet0/2, RPF nbr 33.33.33.1
  Outgoing interface list:
    Ethernet0/0, Forward/Dense, 00:05:35/stopped

(*, 239.255.255.250), 00:05:54/00:01:58, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Dense, 00:05:54/stopped
    Ethernet0/2, Forward/Dense, 00:05:54/stopped

(*, 224.0.1.40), 00:06:45/00:02:59, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/2, Forward/Dense, 00:06:45/stopped
```

Debug messages

```
%LINK-5-CHANGED: Interface Ethernet0/0, changed state to administratively down
PIM(0): Prune Ethernet0/0/239.255.255.250 from (*, 239.255.255.250)
PIM(0): Prune Ethernet0/0/239.1.1.1 from (*, 239.1.1.1)
PIM(0): Prune Ethernet0/0/239.1.1.1 from (50.50.50.254/32, 239.1.1.1)
PIM(0): Insert (50.50.50.254,239.1.1.1) prune in nbr 33.33.33.1's queue
PIM(0): Building Join/Prune packet for nbr 33.33.33.1
PIM(0):  Adding v2 (50.50.50.254/32, 239.1.1.1) Prune
PIM(0): Send v2 join/prune to 33.33.33.1 (Ethernet0/2)
```



```
ACCESS-SW(config-if)#
*Mar 22 20:53:38.410: PIM(0): Insert (50.50.50.254,239.1.1.1) prune in nbr 33.33.33.1's queue
*Mar 22 20:53:38.410: PIM(0): Building Join/Prune packet for nbr 33.33.33.1
*Mar 22 20:53:38.410: PIM(0):  Adding v2 (50.50.50.254/32, 239.1.1.1) Prune
*Mar 22 20:53:38.410: PIM(0): Send v2 join/prune to 33.33.33.1 (Ethernet0/2)
*Mar 22 20:53:39.114: PIM(0): Building Graft message for 239.1.1.1, Ethernet0/0: no entries
*Mar 22 20:53:39.114: PIM(0): Building Graft message for 239.1.1.1, Ethernet0/2:
     50.50.50.254/32 count 1
*Mar 22 20:53:39.114: PIM(0): Send v2 Graft to 33.33.33.1 (Ethernet0/2)
*Mar 22 20:53:39.115: PIM(0): Received v2 Graft-Ack on Ethernet0/2 from 33.33.33.1
*Mar 22 20:53:39.115:       Group 239.1.1.1:
     50.50.50.254/32
*Mar 22 20:53:39.405: PIM(0): Check DR after interface: Ethernet0/0 came up!
*Mar 22 20:53:39.405: PIM(0): Changing DR for Ethernet0/0, from 0.0.0.0 to 10.10.3.1 (this system)
*Mar 22 20:53:39.405: %PIM-5-DRCHG: DR change from neighbor 0.0.0.0 to 10.10.3.1 on interface Ethernet0/0
*Mar 22 20:53:39.414: PIM(0): Building Graft message for 239.255.255.250, Ethernet0/0: no entries
*Mar 22 20:53:39.414: PIM(0): Building Graft message for 239.255.255.250, Ethernet0/2: no entries
*Mar 22 20:53:39.414: PIM(0): Building Graft message for 239.1.1.1, Ethernet0/0: no entries
*Mar 22 20:53:39.414: PIM(0): Building Graft message for 239.1.1.1, Ethernet0/2:
     50.50.50.254/32 count 1
*Mar 22 20:53:39.414: PIM(0): Send v2 Graft to 33.33.33.1 (Ethernet0/2)
*Mar 22 20:53:39.415: PIM(0): Received v2 Graft-Ack on Ethernet0/2 from 33.33.33.1
*Mar 22 20:53:39.415:       Group 239.1.1.1:
     50.50.50.254/32
```

Useful commands:

Command	Comments
show ip pim interfaces	**Display PIM interfaces**
show ip pim neighbors	**Display PIM neighbors**
show ip int x/x | I Multicast	**Display the int x/x groups joined for Multicast**
show ip mroute	**Display Multicast Routing Table**
Show ip mroute count	**Display FIB count**
Debug ip pim **debug ip mpacket**	**Debug Pim join debugging messages**

PIM Sparse Mode

PIM Sparse Mode is based on the “pull model” or “explicit join” which use a combination of both a shared tree and a source-based tree. The RP is making the connection between the shared-tree (tree build down the Multicast receivers) and the source-based tree (tree built up to the source).
PIM Sparse Mode Uses both shared trees & source Based trees while dense mode use only source tree.

How Sparse mode shared tree work?
1-discover PIM neighbor & elect DR
2-discover RP
3-tell RP about sources & receivers
4-build shared tree from sender to receiver through RP
5-join shortest path tree
6-leave shared tree

Remember, two trees in shared tree:

- From receiver to RP
- From RP to sender (source)

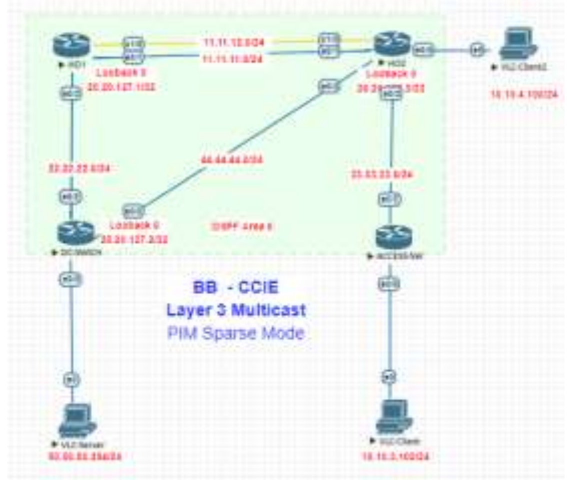
What is RP ?
RP is used as reference point for the root of the shared tree
RP learns about sources through unicast PIM register messages tells about (S,G)
RP learns about receivers through PIM join messages tells to add an interface to the OIL
RP is used to merge the two trees together

Register messages (way to know sources):
When first hop routers connected to source hears traffic , a unicast register message sent to RP
(If many first hop router exists, the DR registers)
If RP accept this message it ack with register Stop and insert (S,G) into table
At this point only DR and RP know (S,G)

Join Message (way to know receivers):
When last hop router receive IGMP join , PIM join is generated up the reverse path tree towards the RP
All routers in the reverse path install (*.G) AND FORWARD TEH JOIN HOP-BY-HOP to the RP
At this point RP and all downstream devices towards the receivers know (*.G)

Merging the trees
Once RP know about sender & receiver for same group , he will send PIM join message up to source
All routers in the path from RP to source will install (*.G) with OIL point to RP
Like dense mode, sparse mode use state refresh to ensure that feeds do not timeout

Now let’s build a config with basic Sparse Mode and test it.

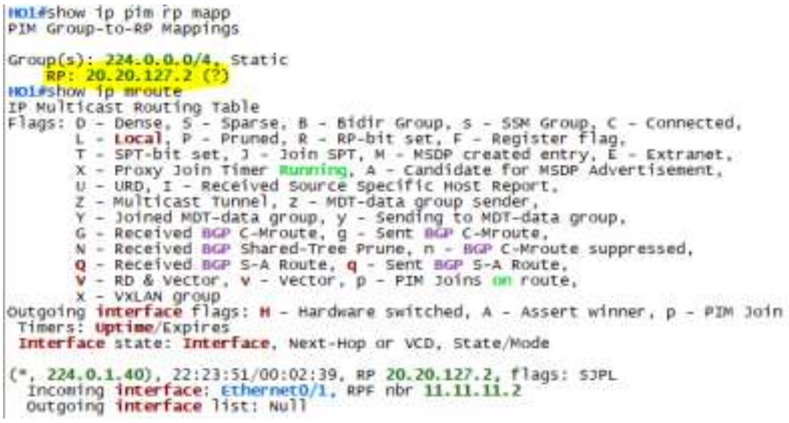
<table>
<tr>
<td>
</td>
<td>Basic config with OSPF
L3-Multicast-sparse-mode.txt</td>
<td>PIM Sparse Mode Config / Static RP
sparse-mode-static-rp.txt</td>
</tr>
</table>

2.2.b [ii] Static RP, auto-RP, BSR

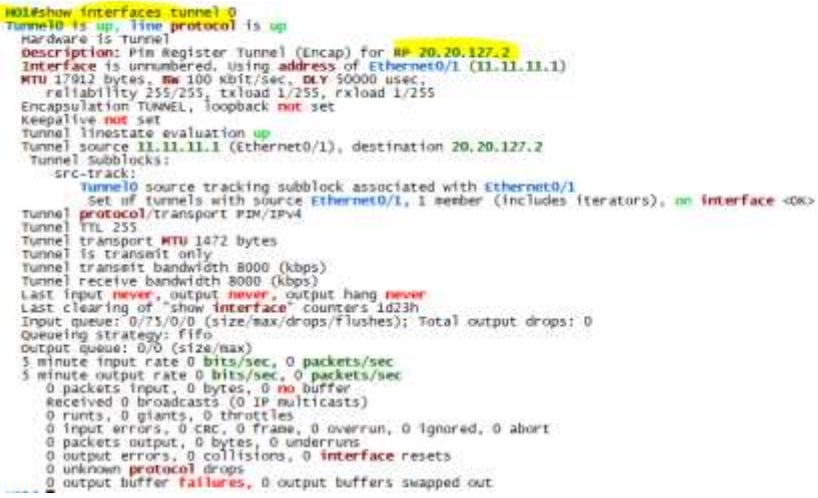
Static-RP

Here I have elected HO2 as Static RP Loopback 0 address 20.20.127.1

You can verify Static RP



```
HO1#show ip pim rp mapp
PIM Group-to-RP Mappings

Group(s): 224.0.0.0/4, Static
    RP: 20.20.127.2 (?)
HO1#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VXLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 224.0.1.40), 22:23:51/00:02:39, RP 20.20.127.2, flags: SJPL
  Incoming interface: Ethernet0/1, RPF nbr 11.11.11.2
  Outgoing interface list: Null
```



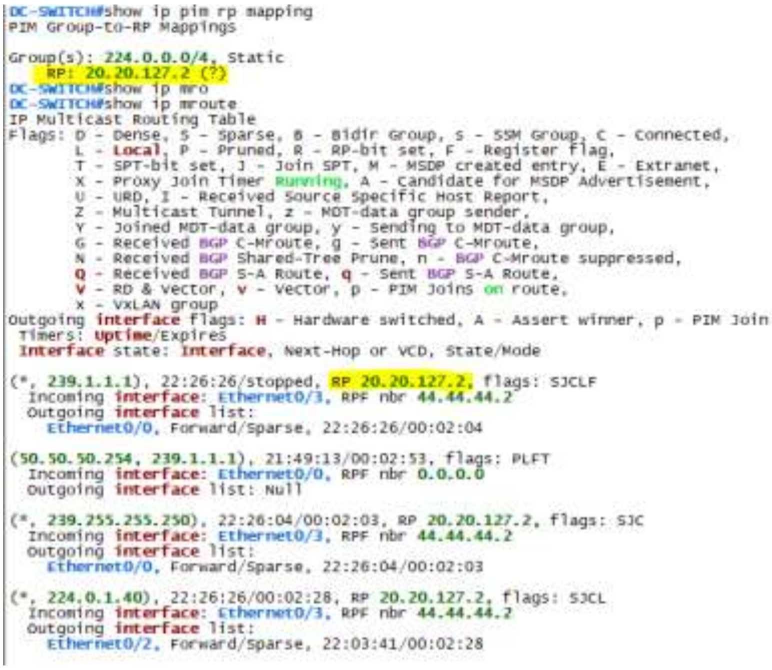
```
H01#show interfaces tunnel 0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Description: Pim Register Tunnel (Encap) for RP 20.20.127.2
  Interface is unnumbered. Using address of Ethernet0/1 (11.11.11.1)
  MTU 17912 bytes, BW 100 Kbit/sec, DLY 50000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel linestate evaluation up
  Tunnel source 11.11.11.1 (Ethernet0/1), destination 20.20.127.2
   Tunnel Subblocks:
      src-track:
         Tunnel0 source tracking subblock associated with Ethernet0/1
          Set of tunnels with source Ethernet0/1, 1 member (includes iterators), on interface <OK>
  Tunnel protocol/transport PIM/IPv4
  Tunnel TTL 255
  Tunnel transport MTU 1472 bytes
  Tunnel is transmit only
  Tunnel transmit bandwidth 8000 (kbps)
  Tunnel receive bandwidth 8000 (kbps)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters 1d23h
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/0 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts (0 IP multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
```

You can not find the running config for the tunnel 0

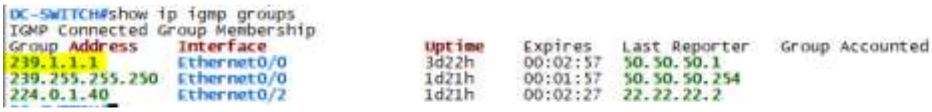
```
H01#show run interface tunnel 0
Building configuration...

Current configuration : 5 bytes
end

H01# sh ip pim tunn
Tunnel0
  Type       : PIM Encap
  RP         : 20.20.127.2
  Source     : 11.11.11.1
  State      : UP
  Last event : RP address reachable (00:44:10)
```

You have to issue **show ip pim tunn** to see the tunnel interface configuration.

Same verify in DC-Switch

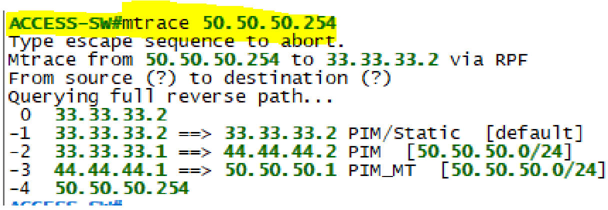


```
DC-SWITCH#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s): 224.0.0.0/4, Static
    RP: 20.20.127.2 (?)
DC-SWITCH#show ip mro
DC-SWITCH#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 22:26:26/stopped, RP 20.20.127.2, flags: SJCLF
  Incoming interface: Ethernet0/3, RPF nbr 44.44.44.2
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 22:26:26/00:02:04

(50.50.50.254, 239.1.1.1), 21:49:13/00:02:53, flags: PLFT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list: Null

(*, 239.255.255.250), 22:26:04/00:02:03, RP 20.20.127.2, flags: SJC
  Incoming interface: Ethernet0/3, RPF nbr 44.44.44.2
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 22:26:04/00:02:03

(*, 224.0.1.40), 22:26:26/00:02:28, RP 20.20.127.2, flags: SJCL
  Incoming interface: Ethernet0/3, RPF nbr 44.44.44.2
  Outgoing interface list:
    Ethernet0/2, Forward/Sparse, 22:03:41/00:02:28
```



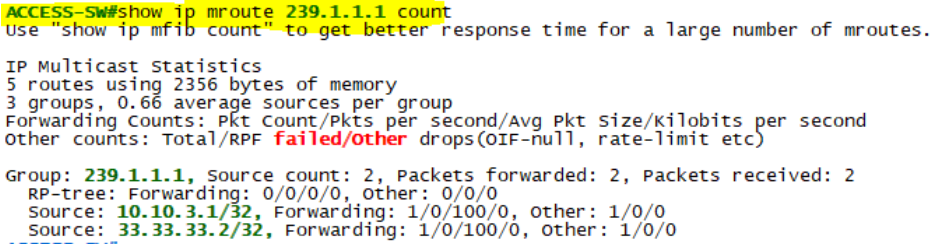
```
DC-SWITCH#show ip igmp groups
IGMP Connected Group Membership
Group Address    Interface                Uptime    Expires   Last Reporter   Group Accounted
239.1.1.1        Ethernet0/0              3d22h     00:02:57  50.50.50.1
239.255.255.250  Ethernet0/0              1d21h     00:01:57  50.50.50.254
224.0.1.40       Ethernet0/2              1d21h     00:02:27  22.22.22.2
```

Let’s verify mtrace where the Multicast Server running (here 50.50.50.254)



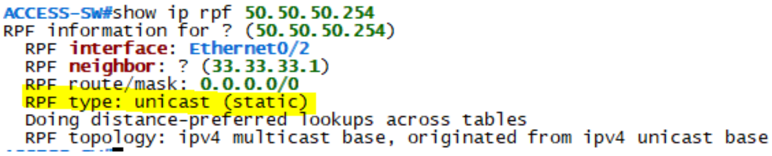
```
ACCESS-SW#mtrace 50.50.50.254
Type escape sequence to abort.
Mtrace from 50.50.50.254 to 33.33.33.2 via RPF
From source (?) to destination (?)
Querying full reverse path...
 0  33.33.33.2
-1  33.33.33.2 ==> 33.33.33.2 PIM/Static  [default]
-2  33.33.33.1 ==> 44.44.44.2 PIM  [50.50.50.0/24]
-3  44.44.44.1 ==> 50.50.50.1 PIM_MT  [50.50.50.0/24]
-4  50.50.50.254
```

Show ip mroute 239.1.1.1 count

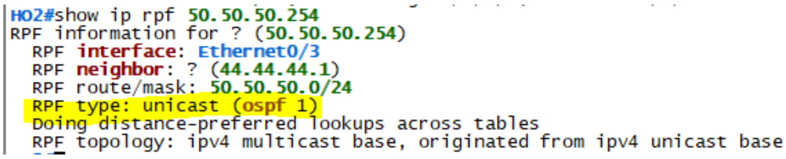


```
ACCESS-SW#show ip mroute 239.1.1.1 count
Use "show ip mfib count" to get better response time for a large number of mroutes.

IP Multicast Statistics
5 routes using 2356 bytes of memory
3 groups, 0.66 average sources per group
Forwarding Counts: Pkt Count/Pkts per second/Avg Pkt Size/Kilobits per second
Other counts: Total/RPF failed/Other drops(OIF-null, rate-limit etc)

Group: 239.1.1.1, Source count: 2, Packets forwarded: 2, Packets received: 2
  RP-tree: Forwarding: 0/0/0/0, Other: 0/0/0
  Source: 10.10.3.1/32, Forwarding: 1/0/100/0, Other: 1/0/0
  Source: 33.33.33.2/32, Forwarding: 1/0/100/0, Other: 1/0/0
```

Lets check RPF check to VLC Server , From Access switch (which has static route) and H02 running OSPF

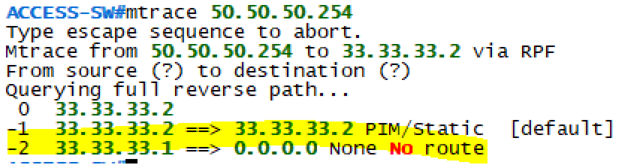


```
ACCESS-SW#show ip rpf 50.50.50.254
RPF information for ? (50.50.50.254)
  RPF interface: Ethernet0/2
  RPF neighbor: ? (33.33.33.1)
  RPF route/mask: 0.0.0.0/0
  RPF type: unicast (static)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

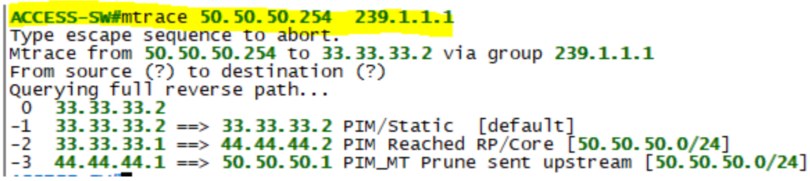


```
H02#show ip rpf 50.50.50.254
RPF information for ? (50.50.50.254)
  RPF interface: Ethernet0/3
  RPF neighbor: ? (44.44.44.1)
  RPF route/mask: 50.50.50.0/24
  RPF type: unicast (ospf 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

Lets remove config pim sparse-mode from H02 towards DC-Switch and do the Trace again.



```
ACCESS-SW#mtrace 50.50.50.254
Type escape sequence to abort.
Mtrace from 50.50.50.254 to 33.33.33.2 via RPF
From source (?) to destination (?)
Querying full reverse path...
 0  33.33.33.2
-1  33.33.33.2 ==> 33.33.33.2 PIM/Static  [default]
-2  33.33.33.1 ==> 0.0.0.0 None No route
```



```
ACCESS-SW#mtrace 50.50.50.254  239.1.1.1
Type escape sequence to abort.
Mtrace from 50.50.50.254 to 33.33.33.2 via group 239.1.1.1
From source (?) to destination (?)
Querying full reverse path...
 0  33.33.33.2
-1  33.33.33.2 ==> 33.33.33.2 PIM/Static  [default]
-2  33.33.33.1 ==> 44.44.44.2 PIM Reached RP/Core [50.50.50.0/24]
-3  44.44.44.1 ==> 50.50.50.1 PIM_MT Prune sent upstream [50.50.50.0/24]
```

If in the mtrace output there were any failures this would indicate that PIM is not enabled on that particular interface.

You can do same testing as above we did in the section Dense mode, running VLC Player as Server and client, make sure you have Video play on Clients if all config ok.

Couple of important Flags :

F flag: Source is directly connected and the register process must be used to notify the RP to this source.
P flag: Outgoing interface is null as no one has joined the SPT tree yet for this source.
T flag: traffic is being received from the source.

SPT Switchover

Cisco implementation of PIM SM dictates that the last-hop router on the shared tree can join the SPT upon receiving the first multicast packet down the shared tree in order to have optimal path to the source and removes the RP as a possible “bottleneck” of all multicast traffic flows. The advantage of SPT switchover is that traffic goes directly from the source to receiver without going through the RP first thereby reducing network latency and possible congestion at the RP.

In short when SPT switchover occurs it will always choose the shared tree directly to the source. You could think it as of this way: the first few packets go through the RP and once the client knows who the source is, it will switchover to the source, using the shared tree.

#ip pim spt-thresold infinity to prevent SPT switchover.

Auto-RP

Auto-RP is a Cisco proprietary protocol.
Transport: UDP port 496 IP address 224.0.1.39 and 224.0.1.40
Address-family: IPv4 (IPv6 not supported)

Auto-RP requires that you configure the RPs to announce their availability as RPs and mapping agents. The RPs use 224.0.1.39 (UDP Port 496) to send their announcements. The RP mapping agent listens to the announced packets from the RPs, then sends RP-to-group mappings in a discovery message that is sent to 224.0.1.40. These discovery messages are used by the remaining routers for their RP-to-group map. You can use one RP that also serves as the mapping agent, or you can configure multiple RPs and multiple mapping agents for redundancy purposes.

Auto-RP defines tow new concepts:

Candidate RP (cRP)
Mapping Agents (MA)

There are two packet types used by Auto-RP:

– RP-announce packets – send by C-RP to all MAs using group address 224.0.1.39. C-RP announce their intention to be RP for a particular group or group range.
– RP-discovery packets – send by MA to all PIM routers using group address 224.0.1.40. These packets used to announce information about elected RPs for particular groups.

This implies that:

– PIM-enabled routers should join the Cisco-RP-discovery group (224.0.1.40)
– MAs should join the Cisco-RP-announce group (224.0.1.39)

How the multicast groups 224.0.1.39 and 224.0.1.40 can be propagated across the network?
Four options here:
– Static RP mapping for groups 224.0.1.39 and 224.0.1.40 – which is not so "auto"
– Dense mode – not scalable
– Sparse-Dense mode
– – if there is RP mapping for particular Group address, router treats this Group as in Sparse mode
– – if there is no RP mapping for particular Group address, router treats this Group as in Dense mode. This behaviour also called Dense Mode fallback – may result in a network dense-mode flood
– Auto-RP Listener feature – traffic for Groups 224.0.1.39 and 224.0.1.40 is flooded across the interfaces configured for PIM-SM

RP-announce
Configuration of router to act like C-RP consist of:
– RP address
– TTL value for the packet
– (optional) List of groups, using access-list:
– – permit statements – define groups that RP willing to process
– – deny statements – define groups that should be processed in Dense mode*
– – if no access-list specified – default group is announced – 224.0.0.0/4
– (optional) Announce Interval of announcements. Default value is 60 seconds. Each RP-Announce contains Holdtime value, which tells how long this announcement is valid. Holdtime is 3 x Announce Interval (180 seconds by default).
*for such groups to be processed in Dense mode router must be configured with Sparse-Dense Mode.

RP-discovery
Configuration of router to act like MA consist of:
– MA address
– TTL value for the packet
– (optional) Discovery Interval of announcements. Default value is 60 seconds. Each RP-Discovery contains Holdtime value, which tells how long this Group-to-RP mapping is valid. Holdtime is 3 x Discovery Interval (180 seconds by default). If information about Group-to-RP mapping expired, then static RP address is used, if configured. If no statically configured RP – router switches group(s) to Dense Mode – traffic will be forwarded only if router is configured with Sparse-Dense Mode.

Candidates RP are the routers who are willing to become RP.

A router is configured as a candidate RP using the following command**:**

ip pim send-rp-annonce *interface* **scope** *ttl* **(group-list acl)**

MAs use the following process when building a discovery message:

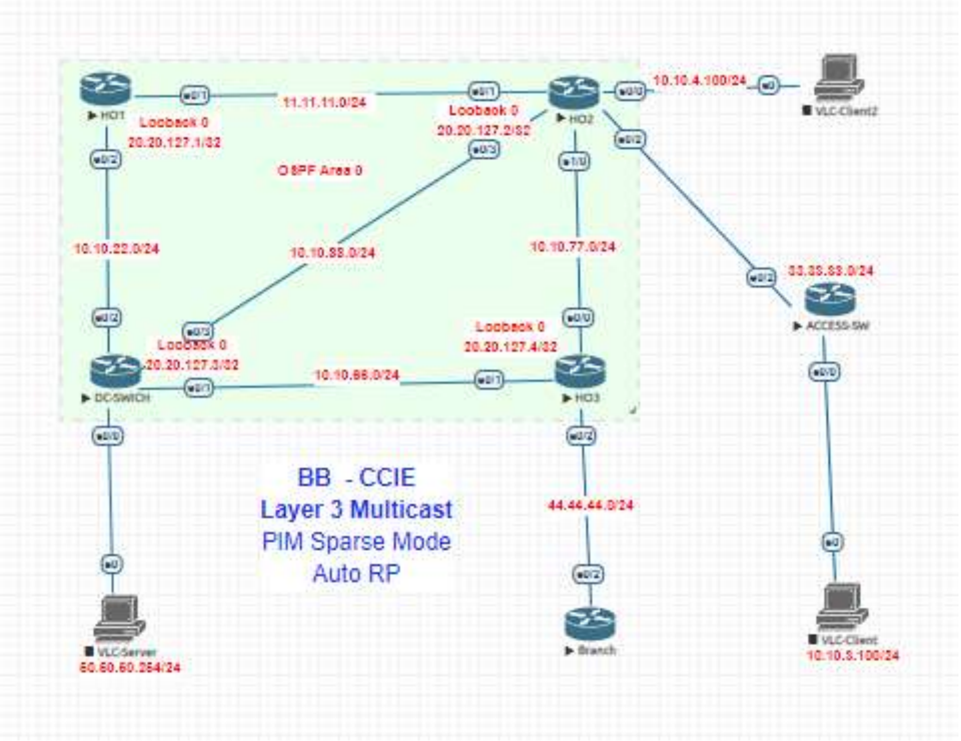
If there are 2 announcements with the same group range but different RPs, the MA will select the RP with the highest RP IP address.

If there 2 announcements where one group is a subset of another, but the RP are different. Both will be sent.

MAs are configured with the following command:

ip pim sedn-rp-discovery <interface> scope <TTL> interval <seconds>.

Here is the Topology using for testing Auto RP



We going to config Candidate RP on DC-Switch and HO2 for 239.1.1.1 and 239.2.2.2

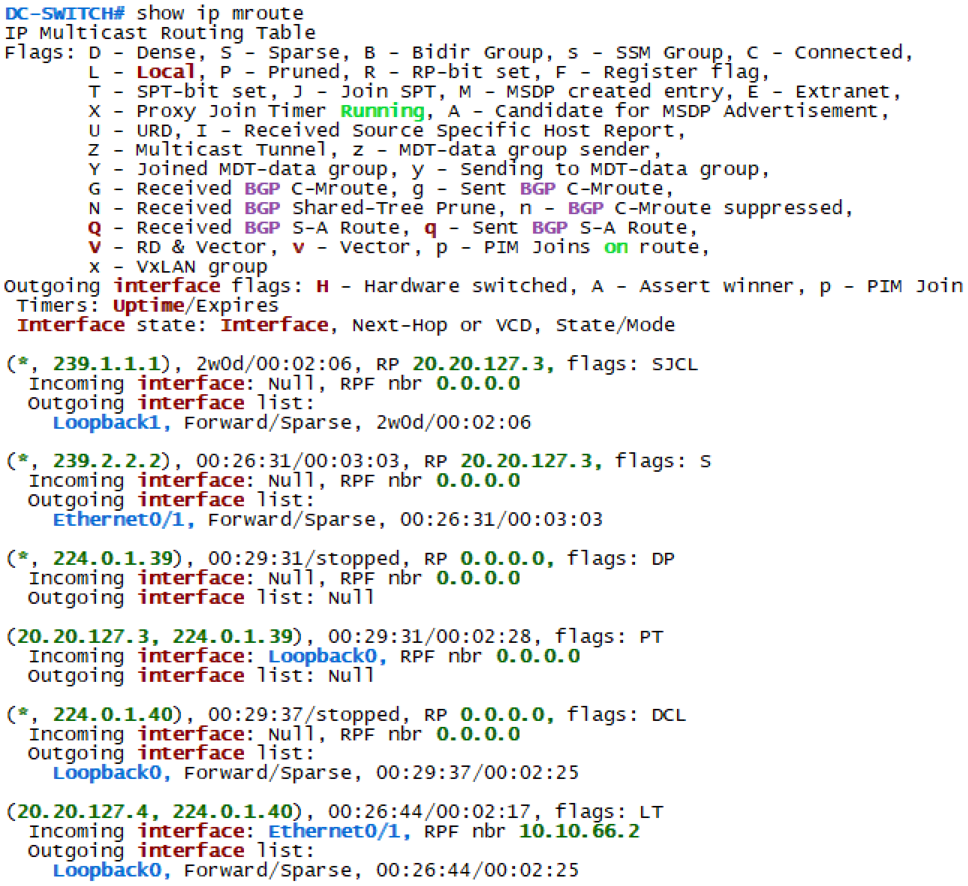
RP Candidate:

```
access-list 11 permit 239.1.1.1
access-list 11 permit 239.2.2.2
ip pim send-rp-announce lo0 scope 10 group-list 11
```

Mapping Agent will be H02

```
ip pim send-rp-discovery lo0 scope 10
```

Lets do some check of mroute



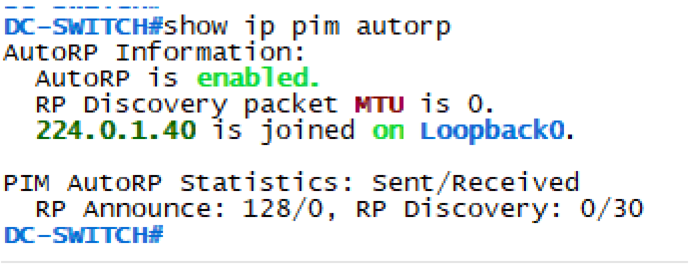
```
DC-SWITCH# show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 2w0d/00:02:06, RP 20.20.127.3, flags: SJCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Loopback1, Forward/Sparse, 2w0d/00:02:06

(*, 239.2.2.2), 00:26:31/00:03:03, RP 20.20.127.3, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/1, Forward/Sparse, 00:26:31/00:03:03

(*, 224.0.1.39), 00:29:31/stopped, RP 0.0.0.0, flags: DP
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list: Null

(20.20.127.3, 224.0.1.39), 00:29:31/00:02:28, flags: PT
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list: Null

(*, 224.0.1.40), 00:29:37/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:29:37/00:02:25

(20.20.127.4, 224.0.1.40), 00:26:44/00:02:17, flags: LT
  Incoming interface: Ethernet0/1, RPF nbr 10.10.66.2
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:26:44/00:02:25
```

We can see two dense-mode routes: (*, 224.0.1.39) and (*, 224.0.1.40), with the flag “D” for Dense. These two routes are for auto-RP. Auto-RP need to use dense-mode for the groups 224.0.1.39 and 224.0.1.40, to be able to send the information to the entire PIM domain.

1. The candidate RP (cRP) send RP-announce messages to the group 224.0.1.39, to propose itself as an RP (UDP packets to port 496, every 60 seconds by default). These messages contain a list of multicast groups the cRP would like to be the RP for.
2. The Mapping agent (MA) listen to 224.0.1.39 to collect all candidates-RP (cRP) information and elect which cRP will be RP for every group (based on highest IP address).
3. The mapping agent send RP discovery messages to 224.0.1.40.
4. The RP info embedded in the RP discovery message sent to 224.0.1.40 contains the best elected RP-to-group mapping information.
5. All cRP listen and receive these messages to know who is the RP.

This is the reason why we need to use **ip pim sparse-dense-mode** on the entire multicast-domain, or the **ip pim autorp listener** command if we use ip pim sparse-mode. The ip pim autorp listener command causes the IP multicast traffic for the two Auto-RP groups, 224.0.1.39 and 224.0.1.40, to be PIM dense-mode (DM) flooded across the interfaces configured for PIM sparse-mode (SM).

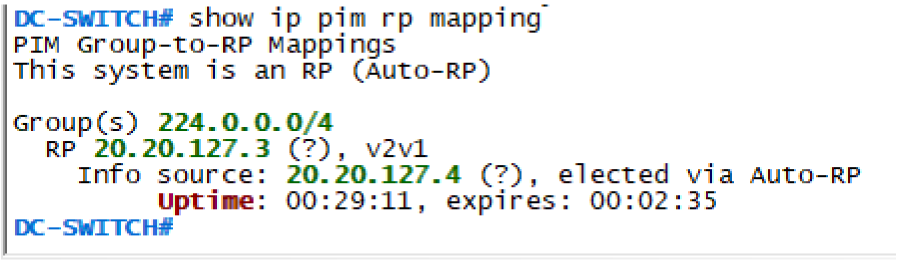
Check Auto RP



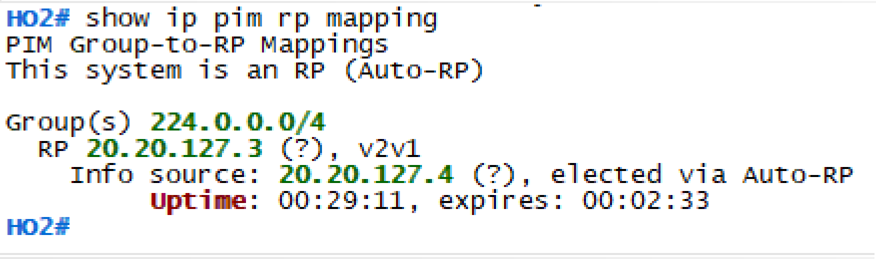
```
DC-SWITCH#show ip pim autorp
AutoRP Information:
  AutoRP is enabled.
  RP Discovery packet MTU is 0.
  224.0.1.40 is joined on Loopback0.

PIM AutoRP Statistics: Sent/Received
  RP Announce: 128/0, RP Discovery: 0/30
DC-SWITCH#
```

Show rp mappings

DC-Switch and HO2



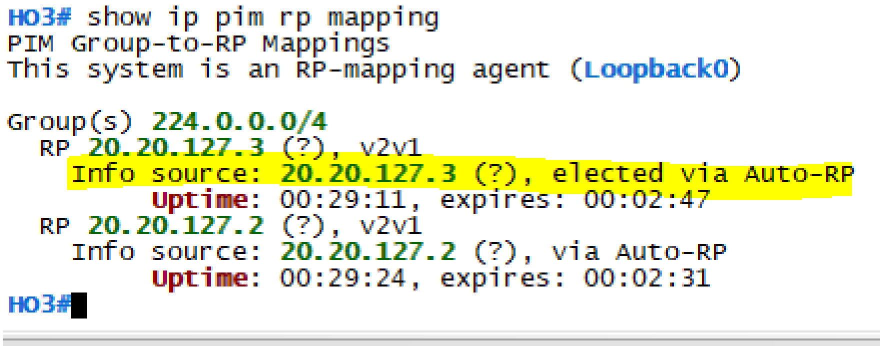
```
DC-SWITCH# show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP (Auto-RP)

Group(s) 224.0.0.0/4
  RP 20.20.127.3 (?), v2v1
    Info source: 20.20.127.4 (?), elected via Auto-RP
         Uptime: 00:29:11, expires: 00:02:35
DC-SWITCH#
```



```
HO2# show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP (Auto-RP)

Group(s) 224.0.0.0/4
  RP 20.20.127.3 (?), v2v1
    Info source: 20.20.127.4 (?), elected via Auto-RP
         Uptime: 00:29:11, expires: 00:02:33
HO2#
```

Noticed here RP address for DC-Switch as RP, because of Loopback Address is Higher (20.20.127.3 compare to 20.20.127.2)
You can see Info Source Mapping agend 20.20.127.4

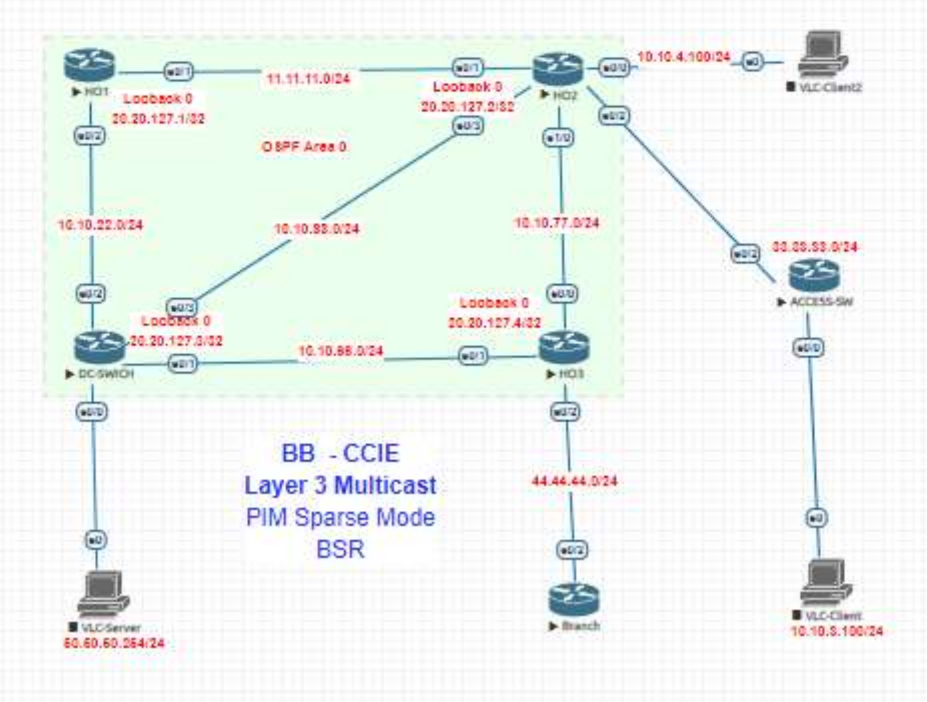
Mapping agend H03 output.



```
HO3# show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP-mapping agent (Loopback0)

Group(s) 224.0.0.0/4
  RP 20.20.127.3 (?), v2v1
    Info source: 20.20.127.3 (?), elected via Auto-RP
         Uptime: 00:29:11, expires: 00:02:47
  RP 20.20.127.2 (?), v2v1
    Info source: 20.20.127.2 (?), via Auto-RP
         Uptime: 00:29:24, expires: 00:02:31
HO3#
```

Bootstrap router – BSR (Open standard)

- BSR uses the group 224.0.0.13, but it does NOT need to be in dense mode, unlike auto-rp
- 224.0.0.13 is the link-local ALL PIM ROUTERS address as well
- The Bootstrap router is the Mapping Agent in auto-rp, configured using: ip pim bsr-candidate (int) (hash) (pri)
- The RP uses the same name as in auto-rp, configured using: ip pim rp-candidate (int) (ttl) (pri) (acl)
- The hash field will allow you to load balance groups over your RPs – A hash value of 31 ensures all even groups are with on RP and odds are with another, if you have 2 RPs
- You can see which group is mapped to which RP with show ip pim rp-hash (group)
- BSR has priority fields so you can choose which router does what without having to rely on the high IP taking over
- If priority is the same, then highest IP will be chosen like auto-rp
- ip pim bsr-border will allow you to run PIM on an edge interface, but not to share bsr messages



Lets configure DC-Switch and H02 as BSR RP-Candidate
DC-Switch as Prioty 10 and H02 as priority 0

DC-Switch

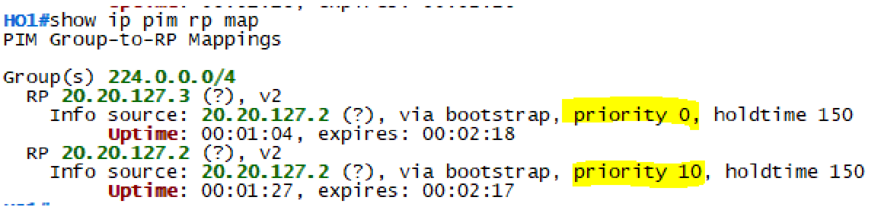
ip pim bsr-candidate loopback 0

ip pim rp-candidate loopback 0 group-list 1 priority 0

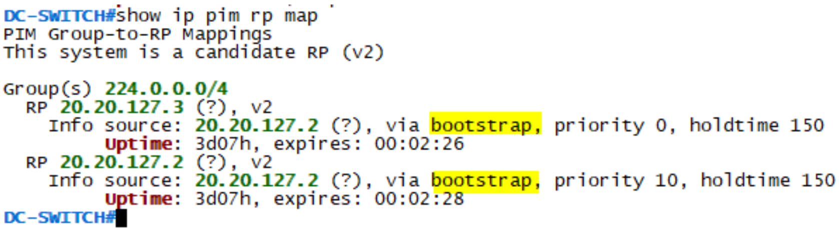
RP-Candidate

HO2
ip pim rp-candidate loopback 0 group-list 1 priority 10

Check BSR RP mappings



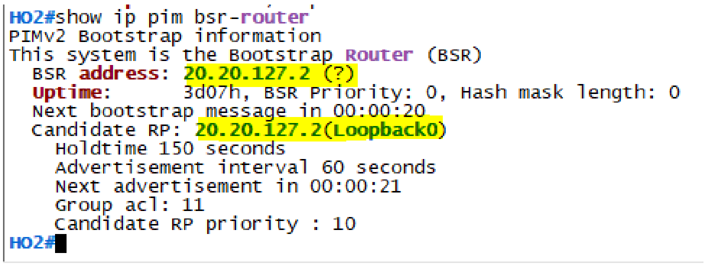
```
H01#show ip pim rp map
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/4
  RP 20.20.127.3 (?), v2
    Info source: 20.20.127.2 (?), via bootstrap, priority 0, holdtime 150
         Uptime: 00:01:04, expires: 00:02:18
  RP 20.20.127.2 (?), v2
    Info source: 20.20.127.2 (?), via bootstrap, priority 10, holdtime 150
         Uptime: 00:01:27, expires: 00:02:17
```



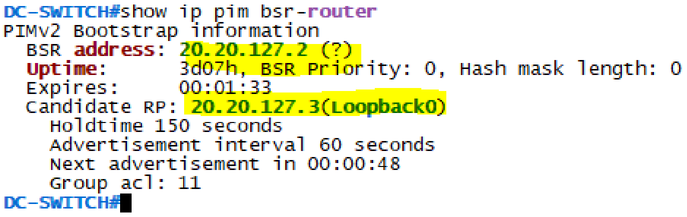
```
DC-SWITCH#show ip pim rp map
PIM Group-to-RP Mappings
This system is a candidate RP (v2)

Group(s) 224.0.0.0/4
  RP 20.20.127.3 (?), v2
    Info source: 20.20.127.2 (?), via bootstrap, priority 0, holdtime 150
         Uptime: 3d07h, expires: 00:02:26
  RP 20.20.127.2 (?), v2
    Info source: 20.20.127.2 (?), via bootstrap, priority 10, holdtime 150
         Uptime: 3d07h, expires: 00:02:28
DC-SWITCH#
```

Check the BSR-router

#show ip pim bsr-router



```
H02#show ip pim bsr-router
PIMv2 Bootstrap information
This system is the Bootstrap Router (BSR)
  BSR address: 20.20.127.2 (?)
  Uptime:      3d07h, BSR Priority: 0, Hash mask length: 0
  Next bootstrap message in 00:00:20
  Candidate RP: 20.20.127.2(Loopback0)
    Holdtime 150 seconds
    Advertisement interval 60 seconds
    Next advertisement in 00:00:21
    Group acl: 11
    Candidate RP priority : 10
H02#
```



```
DC-SWITCH#show ip pim bsr-router
PIMv2 Bootstrap information
  BSR address: 20.20.127.2 (?)
  Uptime:      3d07h, BSR Priority: 0, Hash mask length: 0
  Expires:     00:01:33
  Candidate RP: 20.20.127.3(Loopback0)
    Holdtime 150 seconds
    Advertisement interval 60 seconds
    Next advertisement in 00:00:48
    Group acl: 11
DC-SWITCH#
```

2.2.b [iii] Bidirectional PIM

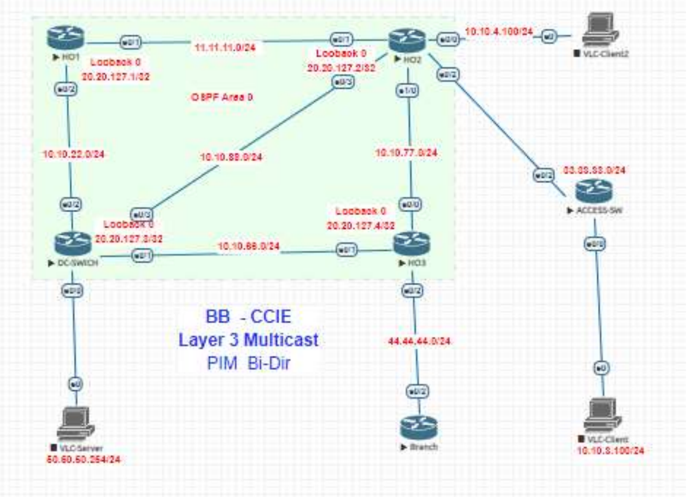
Bidir-PIM use a concept of an elected Designated Forwarder (DF) that establishes a loop-free Shared Tree routed at the RP.

– On each point-to-point link and every network segment one DF is elected for every RP of Bidirectional group, the DF will be responsible for forwarding multicast traffic received on that network.
– The election of DF is based on the best unicast routing metric of the path to the RP, and consider only one path (no Assert messages) the order is as follow:
– Best Administrative Distance.
– Best Metric.
– And then highest IP address.

This way multiple routers can be elected as DF on a segment, one for each RP and one router can be elected DF on more than one interface.
Both Bidir-PIM and the traditional PIM-SM can perfectly coexist in the same PIM domain using the same RPs.

One of the use case was for Bi-directional PIM is: Trading – Low Latency.

One of the major applications for Multicast is financial services, stock markets etc, and in today's climate of HFT (High-Frequency-Trading), latency is a big no-no.



Configure H02 as RP

```
ip pim bidir-enable
ip pim rp-address 20.20.127.2
ip pim send-rp-announce Loopback0 scope 255 bidir
ip pim send-rp-discovery Loopback0 scope 255
```

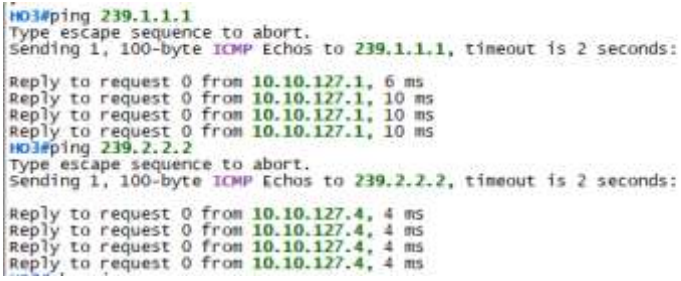
Rest all routers same (H01, HO3, DC-Switch) along with sparse-mode config

```
ip pim bidir-enable
ip pim rp-address 20.20.127.2
```

Lets do checkings and tests.

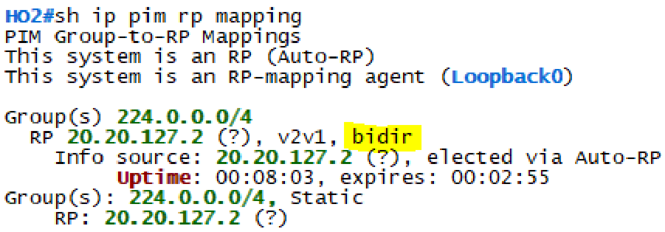
Command	Description
# mrinfo	- Shows info about PIM neighbor connectivity
# sh ip pim interface	- Shows the interfaces with PIM configured - Shows the mode, query interval and DR per segment
# sh ip pim rp mapping	- Shows the PIM group-to-RP mappings
# sh ip pim neighbors	- Shows PIM neighbours
# sh ip pim int df	- Shows info about the elected DF for each RP of an interface

From H03 able to ping 238.1.1.1 and 238.2.2.2

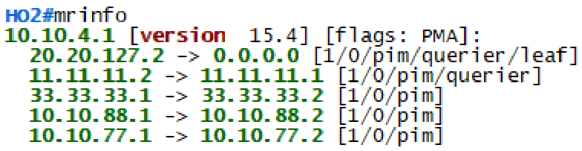


```
H03#ping 239.1.1.1
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 239.1.1.1, timeout is 2 seconds:

Reply to request 0 from 10.10.127.1, 6 ms
Reply to request 0 from 10.10.127.1, 10 ms
Reply to request 0 from 10.10.127.1, 10 ms
Reply to request 0 from 10.10.127.1, 10 ms
H03#ping 239.2.2.2
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 239.2.2.2, timeout is 2 seconds:

Reply to request 0 from 10.10.127.4, 4 ms
Reply to request 0 from 10.10.127.4, 4 ms
Reply to request 0 from 10.10.127.4, 4 ms
Reply to request 0 from 10.10.127.4, 4 ms
```

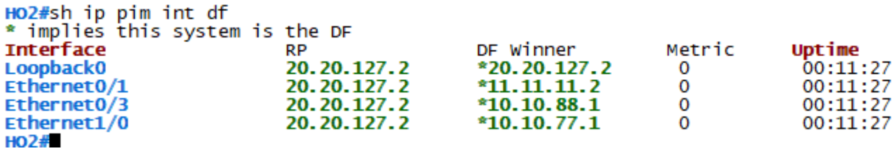
sh ip pim rp mapping



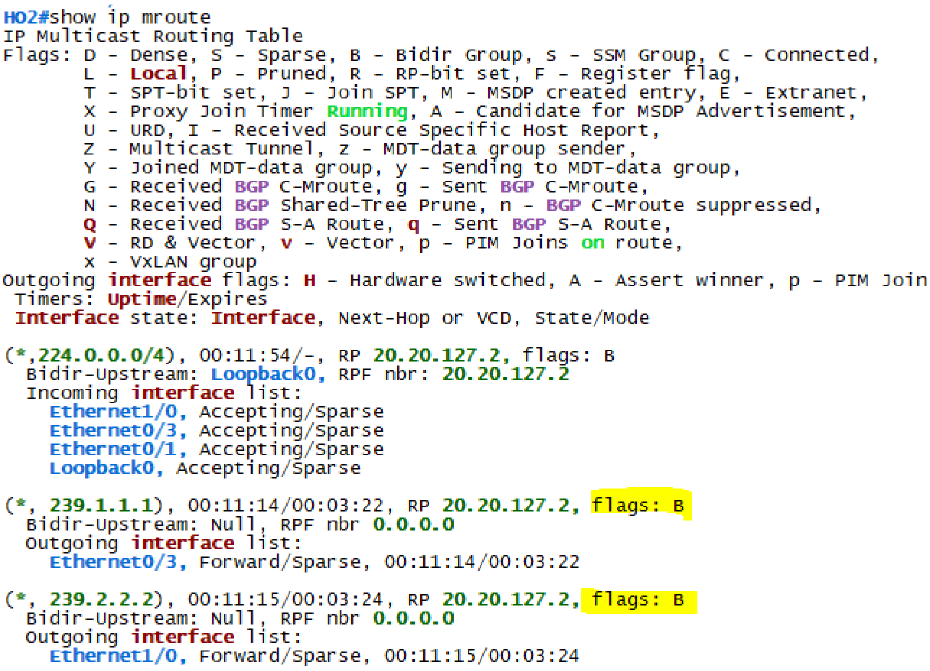
```
HO2#sh ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP (Auto-RP)
This system is an RP-mapping agent (Loopback0)

Group(s) 224.0.0.0/4
  RP 20.20.127.2 (?), v2v1, bidir
    Info source: 20.20.127.2 (?), elected via Auto-RP
         Uptime: 00:08:03, expires: 00:02:55
Group(s): 224.0.0.0/4, Static
    RP: 20.20.127.2 (?)
```



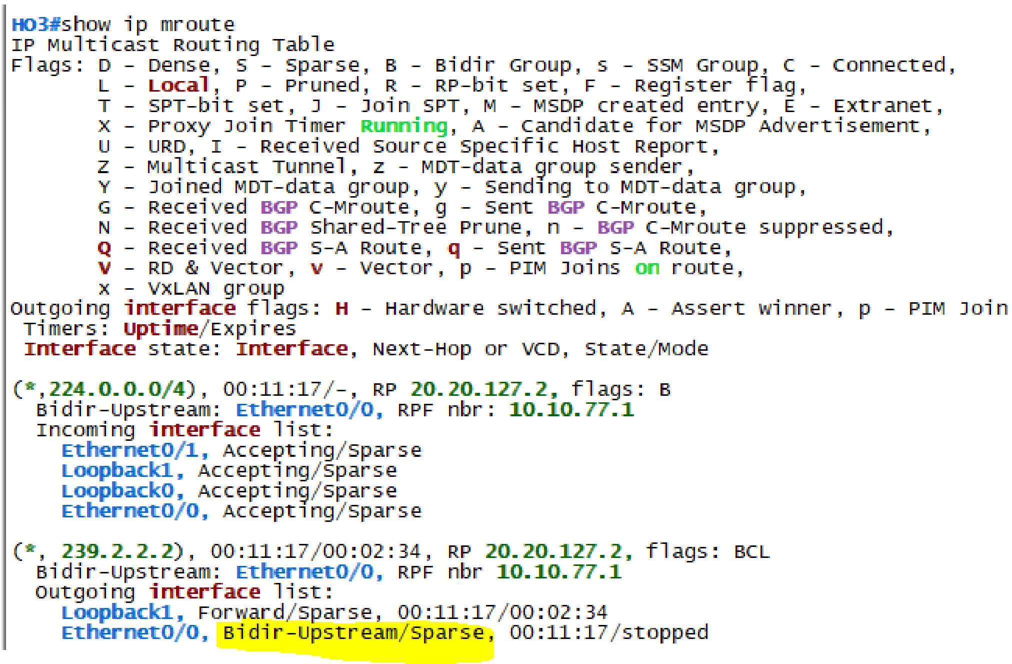
```
HO2#mrinfo
10.10.4.1 [version  15.4] [flags: PMA]:
  20.20.127.2 -> 0.0.0.0 [1/0/pim/querier/leaf]
  11.11.11.2 -> 11.11.11.1 [1/0/pim/querier]
  33.33.33.1 -> 33.33.33.2 [1/0/pim]
  10.10.88.1 -> 10.10.88.2 [1/0/pim]
  10.10.77.1 -> 10.10.77.2 [1/0/pim]
```



```
HO2#sh ip pim int df
* implies this system is the DF
Interface                RP               DF Winner        Metric     Uptime
Loopback0                20.20.127.2      *20.20.127.2      0          00:11:27
Ethernet0/1              20.20.127.2      *11.11.11.2       0          00:11:27
Ethernet0/3              20.20.127.2      *10.10.88.1       0          00:11:27
Ethernet1/0              20.20.127.2      *10.10.77.1       0          00:11:27
HO2#█
```



```
HO2#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*,224.0.0.0/4), 00:11:54/-, RP 20.20.127.2, flags: B
  Bidir-Upstream: Loopback0, RPF nbr: 20.20.127.2
  Incoming interface list:
    Ethernet1/0, Accepting/Sparse
    Ethernet0/3, Accepting/Sparse
    Ethernet0/1, Accepting/Sparse
    Loopback0, Accepting/Sparse

(*, 239.1.1.1), 00:11:14/00:03:22, RP 20.20.127.2, flags: B
  Bidir-Upstream: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/3, Forward/Sparse, 00:11:14/00:03:22

(*, 239.2.2.2), 00:11:15/00:03:24, RP 20.20.127.2, flags: B
  Bidir-Upstream: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet1/0, Forward/Sparse, 00:11:15/00:03:24
```



```
HO3#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*,224.0.0.0/4), 00:11:17/-, RP 20.20.127.2, flags: B
  Bidir-Upstream: Ethernet0/0, RPF nbr: 10.10.77.1
  Incoming interface list:
    Ethernet0/1, Accepting/Sparse
    Loopback1, Accepting/Sparse
    Loopback0, Accepting/Sparse
    Ethernet0/0, Accepting/Sparse

(*, 239.2.2.2), 00:11:17/00:02:34, RP 20.20.127.2, flags: BCL
  Bidir-Upstream: Ethernet0/0, RPF nbr 10.10.77.1
  Outgoing interface list:
    Loopback1, Forward/Sparse, 00:11:17/00:02:34
    Ethernet0/0, Bidir-Upstream/Sparse, 00:11:17/stopped
```

2.2.b [iv] Source-specific multicast

The multicast that you are probably familiar with (PIM sparse and dense mode) using IGMPv2 are also known as ASM (Any Source Multicast). This means that the receivers really don't care what source they receive multicast traffic from, all sources are accepted.

Using sparse mode our receivers require the RP (Rendezvous Point) to discover new sources in the network.

SSM (Source Specific Multicast) requires IGMPv3 and lets us join multicast groups from specified source addresses.

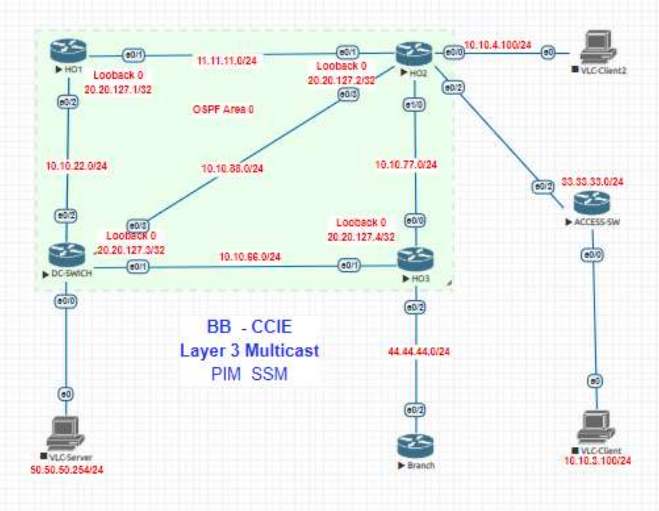
SSM uses no RP, instead it uses IGMP version 3 to signal what channel (source) it wants to join for a group. IGMPv3 can use INCLUDE messages that specify that only these sources are allowed or they can use EXCLUDE to allow all sources except for these ones.

SSM has the IP range 232.0.0.0/8 allocated and it is the default range in IOS but we can also use SSM for other IP ranges. If we do we need to specify that with an ACL.

SSM can be enabled on all routers that should work in SSM mode but it is only really needed on the routers that have receivers connected since that is the place where the behaviour is really changed. Instead of sending a (*,G) join to the RP the Last Hop Router (LHR) sends a (S,G) join directly to the source.

- SSM is an extension of IP multicast where datagram traffic is forwarded to receivers from only those multicast sources to which the receivers have explicitly joined.
- For multicast groups configured for SSM, only source-specific multicast distribution trees (no shared trees) are created.
- IANA has reserved the address range 232.0.0.0/8 for SSM applications and protocols.

Command	Description
#ip pim ssm [default | range-acl]	Enables SSM by defining the SSM range of IP multicast addresses
#interface fa0/0 #ip igmp version 3	Enables IGMPv3 on this interface. IGMPv3 required for SSM (def = v2)
#sh ip igmp groups detail	Shows the (S,G) channel subscription through IGMPv3
#sh ip mroute	Shows whether a multicast group supports SSM service or whether a sourcespecific host report was received



basic config all device

----

ip multicast-routing

Enable pim sparse-mode on all interface participated

----

ip pim sparse-mode

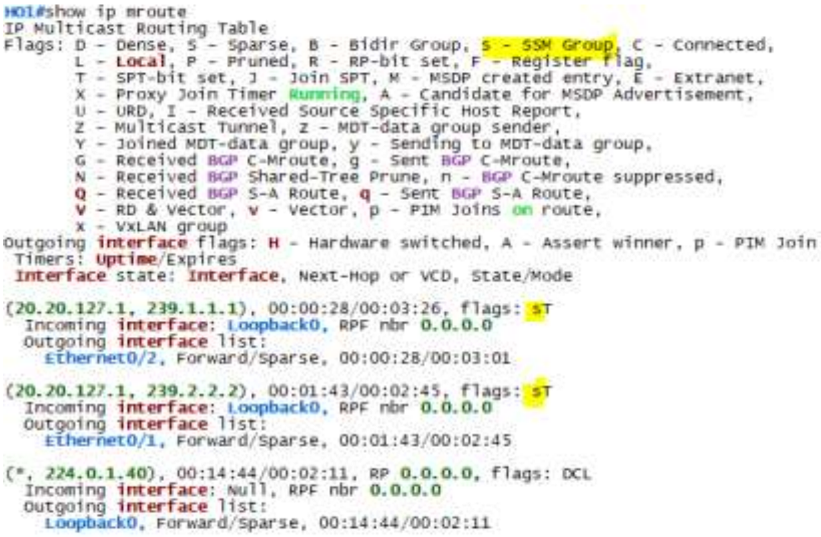
ACL to allow different range other than 232.0.0.0/8 on the source

----

```
config t
!
access-list 12 permit 239.1.1.1
access-list 12 permit 239.2.2.2
ip pim ssm range 12
!
end
```

IGMP join from client (make sure igmp v3 enablesd on the interface)

```
interface Loopback1
 ip address 10.10.127.1 255.255.255.0
 ip pim sparse-mode
 ip igmp join-group 239.1.1.1 source 20.20.127.3    < -- all the source ip to join
 ip igmp join-group 239.1.1.1 source 20.20.127.1
 ip igmp join-group 239.1.1.1 source 20.20.127.2
 ip igmp join-group 239.1.1.1 source 20.20.127.4
 ip igmp version 3
```



```
HO1#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(20.20.127.1, 239.1.1.1), 00:00:28/00:03:26, flags: sT
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/2, Forward/Sparse, 00:00:28/00:03:01

(20.20.127.1, 239.2.2.2), 00:01:43/00:02:45, flags: sT
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/1, Forward/Sparse, 00:01:43/00:02:45

(*, 224.0.1.40), 00:14:44/00:02:11, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:14:44/00:02:11
```

2.2.b [v] Group to RP mapping

Auto-RP automates the distribution of group-to-rendezvous point (RP) mappings in a PIM network. To make Auto-RP work, a device must be designated as an RP mapping agent, which receives the RP announcement messages from the RPs and arbitrates conflicts. The RP mapping agent then sends the consistent group-to-RP mappings to all other devices by way of dense mode flooding. Thus, all routers automatically discover which RP to use for the groups they support. The Internet Assigned Numbers Authority (IANA) has assigned two group addresses, 224.0.1.39 and 224.0.1.40, for Auto-RP.

The mapping agent receives announcements of intention to become the RP from Candidate-RPs. The mapping agent then announces the winner of the RP election. This announcement is made independently of the decisions by the other mapping agents.

2.2.b [vi] Multicast boundary

Controlling the border for a multicast domain can be done with two basic mechanisms. Fist using the TTL option with AutoRP but some topologies make this difficult to implement effectively. Another option is the ip multicast boundary acl filter-autorpcommand. This feature can filter both the data and control planes based on the applied acl. Standard access lists specify the group to filter and extended access lists define source and group.

- IP PIM sparse mode configuration
- Controlling the edge of a multicast domain
- Using ACL's to define filtering criteria

2.2.c Implement and troubleshoot multicast source discovery protocol

Multicast Source Discovery Protocol (MSDP) is a protocol designed to address the interconnection among multiple PIM-SM multicast domains. The goal of this protocol is to help a multicast domain to **discover information about multicast sources**located in other PIM-SM multicast domains.

MSDP is a fundamental piece in an **inter-domain multicast solution**. The applicability of MSDP is limited to the Any Source Multicast (ASM) model with PIM-SM as the intra-domain multicast routing protocol. Another important applicability point to consider is that MSDP is designed to work with IPv4, and there is not an IPv6 version.

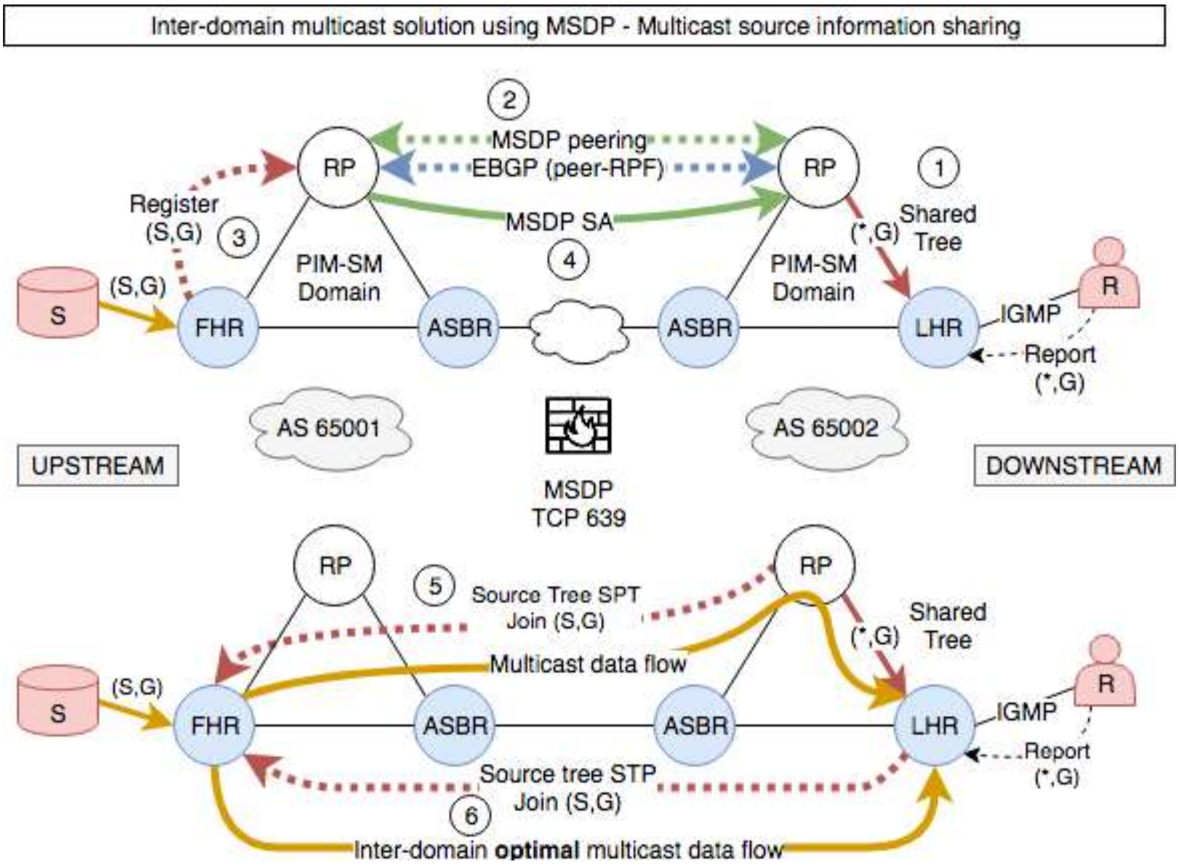
Architecture

Let’s start first with a brief introduction to the problem solved by MSDP, followed by a simple example of an Interdomain multicast case using MSDP and some important MSDP design considerations.

In the ASM model with PIM-SM, the receiver Designated Router (DR) sends joins messages to the RP in response to the receiver interest in receiving some group G. This results in the construction of a shared tree from the RP to the receivers. When a source starts transmitting, the source DR registers to the RP, letting know its presence and starting the process of data flow down the shared tree to the receivers. The point here is that the **RP is the central point of the PIM-SM multicast domain that obtains all the information about the interested receivers and all the active sources**.

For several reasons (security, efficiency, etc.), the network administration may decide to split up the multicast domain in two different PIM-SM domains. After that, the RP in different multicast domains will be isolated and they won’t have access to multicast source information from any other domains. **MSDP solves the problem of source discovery between isolated multicast domains**.

The figure shows two PIM-SM domains in two different AS. MSDP may be used in this scenario to provide an inter-domain multicast solution. With MSDP, remote RP can discover active multicast sources in other domains, enabling multicast traffic delivery to the interested local receivers.



A brief explanation of the pictured steps:

Receiver joins to (*,G).
MSDP peering in place, with MBGP in parallel to ensure peer-RPF check PASS.
Source appears and registers to its local RP.
MSDP shares information about sources with peers via MSDP SA messages.
Remote RP joins the source (S,G) and starts delivering multicast traffic via shared tree to interested receivers.
Remote FHR (LHR in the data path) may decide to join the Source if the SPT switchover is enabled, resulting in inter-domain optimal multicast data flow.

The MSDP peering is established between routers to share multicast source information (TCP 639). The peering is usually configured between RP devices, but any other router can be a MSDP peer too. Once the source is discovered PIM-SM mechanisms allows multicast users in one domain to receive multicast data from sources located in remote domains.

MSDP brings some other important functionalities with it, like **Anycast RP**. Anycast RP provides redundancy and load sharing functionality to high availability scenarios. Basically, it allows several RP in the same domain to share the same IP address. Setting up MSDP peers between RP, the multicast sources register... and the multicast receivers join... to the closest RP, ending up in a load balanced with RP redundancy scenario.

Reverse Path Forwarding Rules

MSDP peers must follow the peer-RPF forwarding rues to avoid MSDP SA message looping conditions.
In some cases, the peer-RPF check may be avoided. This should happen in the following cases:

The MSDP peer is the Originating RP of the MSDP SA message.
The MSDP peer is a mesh-group peer (in this case the receiver device doesn't forward the MSDP SA message to any other mesh group member).
The MSDP peer is configured as the default MSDP peer.
The MSDP peer is the one and only MSDP peer configured.

When MSDP is used in combination with MBGP to ensure peer-RPF check pass, the following rules apply:

Sending MSDP peer	Condition to pass the peer-RPF check
iBGP peer	MSDP peer address == BGP peer address
eBGP peer	First AS in the path to RP == MSDP peer AS
Not a BGP peer	First AS in the path to RP == Sending MSDP peer AS

In the RPF calculation is important to consider the **preference** of the different routing tables:

Static Mroute
MBGP
Unicast

The multicast routing information base (MRIB) is always checked before the unicast RIB (URIB). If there is no path on any table to the MSDP peer, then the peer-RPF will fail and the MSDP SA message will be discarded.

Mesh groups

MSDP mesh groups help to **reduce the number of MSDP SA messages** that circulates among MSDP peers. It's basically a MSDP optimization mechanism to reduce the control traffic transported by the network.

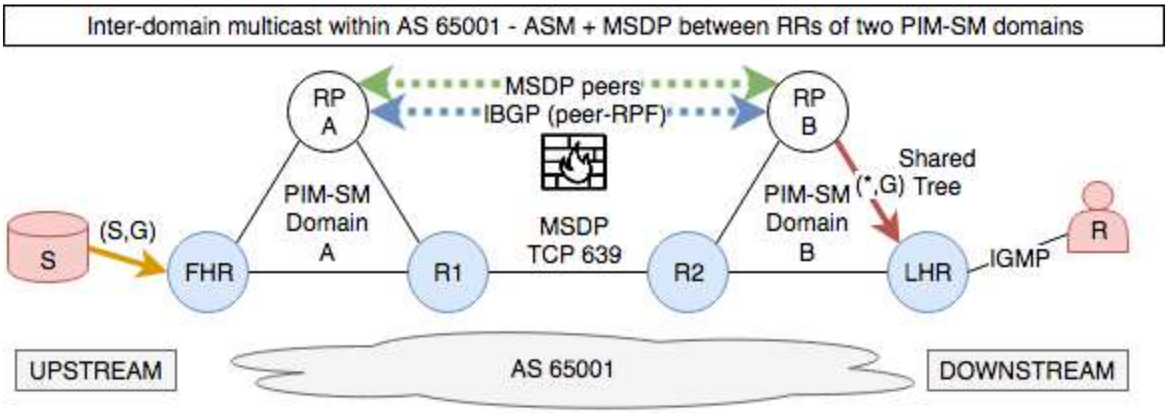
MSDP peers are configured in full-mesh configuration and its members can be in the same or in different multicast domains as well on the same or different AS.

The **peer-RPF check for mesh group members** is a bit different than in a nonmember. After the reception of the MSDP SA message, the source of the peer is checked:

If the MSDP peer is an outsider, then perform a peer-RPF check. If the check PASS, then forward the message to all mesh group members; else drop.
If the MSDP peer is a mesh group member, then accept the MSDP SA message without peer-RPF check (and do not forward the message to any other member of the mesh group).

Interdomain multicast within an AS

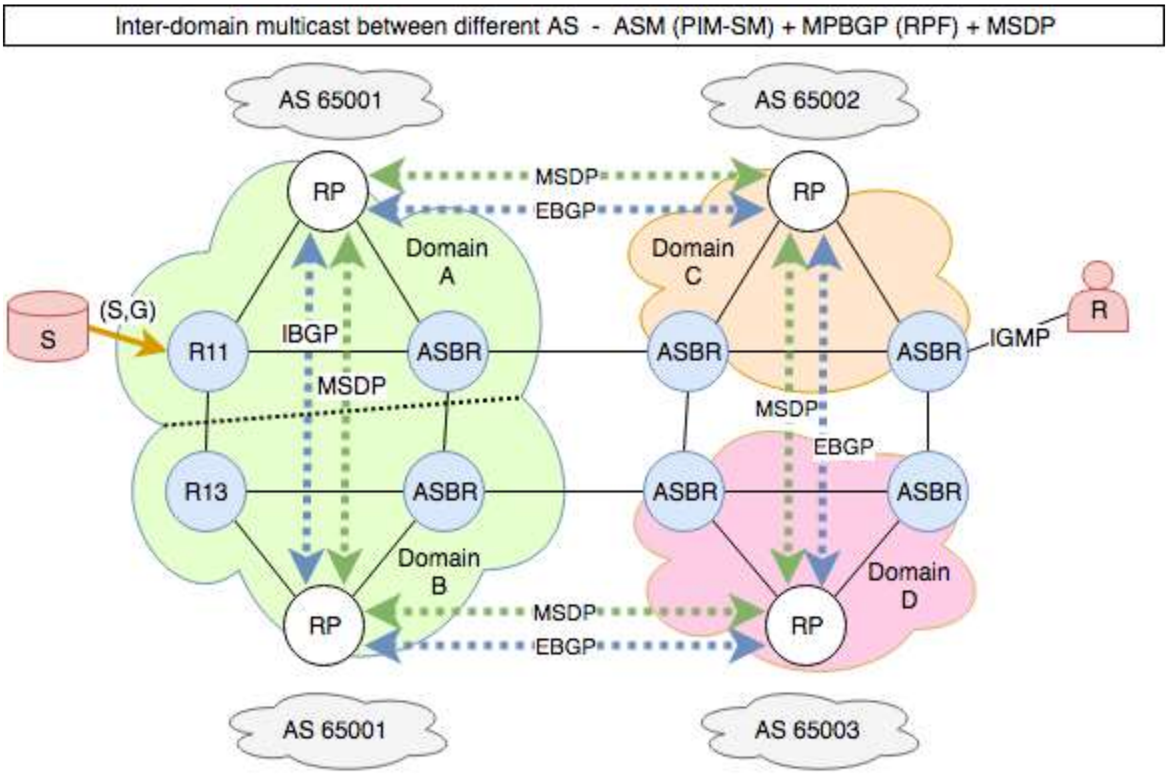
This is one of the typical MSDP use cases. Two PIM-SM multicast domains administratively isolated within the same AS.



In this case, as each RP has only one MSDP peer, the IBGP session to perform the peer-RPF check is optional. The MSDP SA messages are automatically accepted without any check (see RPF rules section).

Interdomain multicast among different AS

In this case the source information is distributed among different multicast domains, some of them in the same AS and some in remote AS. In this case MSDP in combination with MBGP in parallel could be used.



The MBGP parallel the MSDP peer relationships to ensure the RPF check success.

Summary

MSDP enabled remote source discovery and opened the possibility to modularize and isolate different multicast domains that may or not be spread among autonomous systems. Let's review some of the **design advantages** derived from using MSDP:

Modularity & isolation: a network may be designed with different PIM-SM domains, each one independent from each other from the functionality point of view. As each domain uses its own RP, local sources and interested receivers can work independently. Modularity and isolation between modules can help in case of failure contention and clear choke points between modules reduce the Mean Time to Repair (MTTR).
Security: from the security point of view, the network administration can decide if the multicast source information of one domain is shared or not with other domains within their own AS or with remote AS.
Efficiency: interested receivers within a multicast domain can signal their interest in receiving information about a multicast group to their local RP. This way, the state of the network is kept lower and distributed among multicast domains.

Last but not least, MSDP applications like Anycast RP provide for redundancy and load-balancing functionality to RP deployments. Anycast RP is available for IPv4 and IPv6, but MSDP it's not, so in IPv6 scenarios an extension of PIM should be used to provide that kind of functionality.

2.2.c.[i] Intra-domain MSDP [anycast RP]

IP multicast is deployed as an integral component in mission-critical networked applications throughout the world. These applications must be robust, hardened, and scalable to deliver the reliability that users demand.
Using Anycast RP is an implementation strategy that provides load sharing and redundancy in Protocol Independent Multicast sparse mode (PIM-SM) networks. Anycast RP allows two or more rendezvous points (RPs) to share the load for source registration and the ability to act as hot backup routers for each other. Multicast Source Discovery Protocol (MSDP) is the key protocol that makes Anycast RP possible.
The scope of this document is to explain the basic concept of MSDP and the theory behind Anycast RP. It also provides an example of how to deploy Anycast RP.

2.2.c.[ii] SA filter

MSDP-SA messages contain (source, group (S,G)) information for rendezvous points (RPs) (called MSDP peers) in Protocol Independent Multicast sparse-mode (PIM-SM) domains. This mechanism allows RPs to learn about multicast sources in remote PIM-SM domains so that they can join those sources if there are local receivers in their own domain. You can also use MSDP between multiple RPs in a single PIM-SM domain to establish MSDP mesh-groups.

With a default configuration, MSDP exchanges SA messages without filtering them for specific source or group addresses.

Typically, there are a number of (S,G) states in a PIM-SM domain that should stay within the PIM-SM domain, but, due to default filtering, they get passed in SA messages to MSDP peers. Examples of this include domain local applications that use global IP multicast addresses, and sources that use local IP addresses (such as 10.x.y.z). In the native IP multicast Internet, this default leads to excessive (S,G) information being shared. To improve the scalability of MSDP in the native IP multicast Internet, and to avoid global visibility of domain local (S,G) information, we recommend using the following configuration to reduce unnecessary creation, forwarding, and caching of some of these well-known domain local sources.

Reading References:

https://www.cisco.com/c/en/us/support/docs/ip/multicast/200101-Multicast-Routing-MSDP-and-PIM-walk-th.html

https://www.cisco.com/c/en/us/about/security-center/multicast-toolkit.html

==================== LEFT BLANK==

2.3 Fundamental routing concepts

2.3.a Implement and troubleshoot static routing
2.3.b Implement and troubleshoot default routing

- Static routing refers to routes to destinations being listed manually, or statically, as the name implies, in the router. Network reachability in this case is not dependent on the existence and state of the network itself. Whether a destination is active or not, the static routes remain in the routing table, and traffic is still sent toward the specified destination.
- Default routing refers to a "last resort" outlet. Traffic to destinations that is unknown to the router is sent to that default outlet. Default routing is the easiest form of routing for a domain connected to a single exit point.
- Dynamic routing refers to routes being learned via an interior or exterior routing protocol. Network reachability is dependent on the existence and state of the network. If a destination is down, the route disappears from the routing table, and traffic is not sent toward that destination.

Default routes are used to direct packets addressed to networks not explicitly listed in the routing table. Default routes are invaluable in topologies where learning all the more specific networks is not desirable, as in case of stub networks, or not feasible due to limited system resources such as memory and processing power.

This document explains how to configure a default route, or gateway of last resort.

```
ip default-gateway
ip default-network

ip route 0.0.0.0 0.0.0.0 x.x.x.x
```

2.3.c Compare routing protocol types
2.3.c [i] distance vector
2.3.c [ii] link state
2.3.c [iii] path vector

Distance vector	**Link state**	**Path vector**
Distance – hop or metric Vector – direction ex. **Ripv2** • Bellman-Ford • periodic transmission of entire routing table • mathematical comparison using measurement of distance • limited by hop count (15Max)	ex. **OSPF and IS-IS** • Dijkstra • sends local connection information to all nodes on the internetwork • forms adjacencies with and sends link state information to same protocol speaking connected neighbors • about the state of it's own links	ex. **BGP** • subset of DV • constructs lists of passed AS paths to establish metrics and remain loop free • attributes are employed to affect sending and reception of traffic

	• constructs a view of the entire topology with this information	
Advanced DV ex. **EIGRP** • DUAL (Diffusing Update Algrithm) • exhibits characteristics of both link state and DV		

2.3.d Implement, optimize and troubleshoot administrative distance

Best path selection – first criterion a router uses to determine which routing protocol to use if 2 protocols provide information to the same destination

Lower admin distance is considered more reliable

Modify AD of a protocol with 'distance' cmd in the routing process – Local to that router

Routing decision criteria	Administrative Distances (default)	Routing Table
1) Valid next hop upon update receipt, the router verifies a valid next hop 2) Metric the router then attempts to install the best routing metric (OSPF=cost, ie) into the table 3) Administrative Distance If multiple routing protocols are running, the route with the lowest AD is used 4) Prefix Longest match preferred; trumps AD	Connected interface 0 Static route 1 EIGRP summary route 5 External BGP 20 Internal EIGRP 90 IGRP 100 OSPF 110 IS-IS 115 RIP 120 Exterior Gateway Protocol 140 On-demand routing 160 External EIGRP 170 Internal BGP 200 Unknown 255 A floating static route is a route artificially increased from it's default, ie. for EIGRP a floating static could be added with an AD of 93, which would be used in the event of a dynamic EIGRP route (AD 90) not found.	Routing's classic model, hop-by-hop, keeps a list of destinations with their reachable next hops in the routing table. In recent years the hop-by-hop paradigm has given way to more advanced methods such as MPLS, which enables label lookup to dictate the next hop determination. Traffic engineering considerations have become popular as well, such as QOS, that are not limited to routing table only criteria.

2.3.e Implement and troubleshoot passive interface

Passive interfaces participate in routing protocols however they do not send hello messages.
This prevents adjacencies from being formed on that interface.
The interface will also not receive routing updates.
The benefit is that the interface will still be advertised into the routing protocol.

You can configure passive interfaces in two ways.

The first was is to specify a specific interface as passive:

#passive-interface <interface>

The second option is to configure all interfaces as passive by default, and then specify an interface as not passive: (in config mode)

#passive-interface default
#no passive-interface <interface>

2.3.f Implement and troubleshoot VRF lite

VRF without MPLS (VRF Lite)

Support different routing tables on the same device

- Allows for overlapping IP addressing as long as they are in different vrf's
- Uses input interfaces to distinguish routes form over vpns

TCAM resources are shared between all vrf's

Note : Does not support ISIS

Configuration

Following only works with IPv4
```
ip vrf [name]
rd [route-distinguisher] --> required
route-target {export | import | both} --> optional in vrf-lite
import map route-map
----
```
Following works with IPv4 and IPv6
```
vrf definition RED
 rd 1:1
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
Interface > ip vrf forwarding [vrf name] --> removes current IP when applied

Router(config-if)#vrf forwarding RED
```

EIGRP

```
router eigrp [as]
address-family ipv4 vrf VRF
```

```
Router(config)#router eigrp NAMED
Router(config-router)#address-family ipv4 vrf RED as 1
```

OSPF

```
router ospf [process id] vrf [vrf name]
```

BGP

```
router bgp [as]
address-family ipv4 vrf [vrf name]
```

2.3.g Implement, optimize and troubleshoot filtering with any routing protocol

Options:
Control routing updates with passive interface

Control processing and advertising in routing updates

- Distribute List – Filter prefixes inbound/outbound
- Offset List – Change incoming/outgoing metrics
- Administrative Distance
- Summarization
- Floating Routes
- OSPF – LSA Filters

Match IP/Prefixes

- Standard Access List
- Extended Access List
- Prefix List
- Tag
- Route Map
  - Match next-hop
  - Match route-source
  - Match Metric
  - Match route-type
  - Match Tag
  - Match Interface
  - Match IP address

OSPF
OSPF can filter on more than just tags or IP addresses. You can also filter LSA types from being forwarded into other areas based on the area type.
Stub Area: Stops type 4 and 5 LSAs

Totally Stubby: Stops type 3, 4 and 5 LSA's, only gets 1 default type 3 LSA
Not So Stubby Area: Translates type 7 into 5. Stops type 4 and 5 LSAs
Totally No So Stubby Area: Translates type 7 into 5. Stops type 3, 4 and 5 LSA's, only gets 1 default type 3 LSA

BGP
BGP is also not limited to filtering on the normal set of tags and ip addresses. Routes can be filtered based on the AS number and communities. More details of this will be in the BGP section

RIP	EIGRP	OSPF	ISIS
Offset List access-list 1 permit host [ip] access-list 2 permit host [ip] ! router rip offset-list 1 in 3 [int] offset-list 2 out 16 [int] **Distribute List with Prefix-list** ip prefix-list NOT_FROM_R4 seq 5 deny 155.1.0.4/32 ip prefix-list NOT_FROM_R4 seq 10 permit 0.0.0.0/0 le 32 ! ip prefix-list PERMIT_ALL seq 5 permit 0.0.0.0/0 le 32 ! ip prefix-list RIP_FILTER_TO_R8 seq 5 deny 150.1.6.6/32 ip prefix-list RIP_FILTER_TO_R8 seq 15 permit 0.0.0.0/0 le 32 ! router rip distribute-list prefix RIP_FILTER_TO_R8 out GigabitEthernet1.58 distribute-list prefix PERMIT_ALL gateway NOT_FROM_R4 in **Distribute List with Standard ACL**	**Distribute List with Prefix List** ip prefix-list NOT_FROM_R4 seq 5 deny 155.1.146.4/32 ip prefix-list NOT_FROM_R4 seq 10 permit 0.0.0.0/0 le 32 ! ip prefix-list PERMIT_ALL seq 5 permit 0.0.0.0/0 le 32 ! router eigrp 100 distribute-list prefix PERMIT_ALL gateway NOT_FROM_R4 in **Distribute List with Standard ACL** access-list 1 permit 0.0.0.0 255.255.254.255 ! router eigrp 100 distribute-list 1 in GigabitEthernet1.79 **Distribute List with Exteneded ACL** access-list 100 deny ip host 155.1.0.4 host 150.1.9.9 access-list 100 permit ip any any ! router eigrp 100 distribute-list 100 in Tunnel0 **Offset List** router eigrp 100	**Distribute List with Standard ACL** router ospf 1 distribute-list 1 in ! access-list 1 deny 150.1.1.1 0.0.0.0 access-list 1 permit any **Administrative Distance** access-list 10 permit 155.1.146.0 ! router ospf 1 distance 255 150.1.5.5 0.0.0.0 10 **Distribute List with Route Map** route-map DENY_VLAN146_FROM_R4 deny 10 match ip address 3 match ip next-hop 4 ! route-map DENY_VLAN146_FROM_R4 permit 20 ! router ospf 1 distribute-list route-map DENY_VLAN146_FROM_R4 in	**Route filtering by route destination with ACL** access-list access-list-number {permit | deny} ip any destination-address destination-wildcard ! interface type number ip router isis [route-tag] ! router isis route-tag distribute-list access-list-number in [interface-type interface-number]

access-list 1 permit 0.0.1.0 255.255.254.255 ! router rip distribute-list 1 in **Distribute List with Extended ACL** access-list 100 deny ip host 155.1.0.1 host 150.1.1.1 access-list 100 permit ip any any ! router rip distribute-list 100 in tunnel0 **Administrative Distance** router rip distance 255 0.0.0.0 255.255.255.255 [acl]	offset-list 1 in 2000 GigabitEthernet1.37 **Administrative Distance** access-list 4 permit host 150.1.4.4 ! router eigrp 100 distance 255 0.0.0.0 255.255.255.255 4 **Distribute List with Route Map** route-map FILTER_ON_TAGS deny 10 match tag 4 ! route-map FILTER_ON_TAGS permit 20 ! router eigrp 100 distribute-list route-map FILTER_ON_TAGS in		

2.3.h Implement, optimize and troubleshoot redistribution between any routing protocol

RIP	**EIGRP**	**OSPF**	**ISIS**	**BGP**
RIP uses a hop count as it’s metric for routes, 16 being unreachable. You’ll want the metric to be defined in a way that will not allow for a routing loop to occur. A metric must be defined when redistibuting other protocols into RIP	EIGRP uses a composite metric can can be computed based on: bandwidth, load, delay, reliability and MTU. A metric must be defined when redisitrubtuing other routing protocols into EIGRP Metric order: Bandwidth (Kbps), delay (tens of microseconds), reliability (255=100%), load (0-255,	OSPF does not require a seed metric (cost), but one can be defined . The default value is 20, except for BGP which is 1. The subnet keyword needs to be defined on the redistributed protocol or only major nets will be redistributed. There are 4 different route types for OSPF when redistributing, E1>E2, N1>N2	Metrics for ISIS are between 1 – 63. There is no default metric and one should be configured. The default metric is 0 if one is not included.	BGP does not use a seed metric when redistrbuting other protocols into it. OSPF has specicial requirements when redistributing into BGP. Only intra-area and inter-area are included by default. You’ll need to use the external keyword if you want external OSPF routes to be included

	255=100%loaded), MTU (1500) Becareful in named mode EIGRP, metric 1 1 1 1 1 no longer works because you’re using wide metrics	Difference is how the cost is calculated ▪ E1 – Normal area, type 5 LSA, external cost plus the internal cost to reach the route ▪ E2 – Normal area, type 5 LSA, cost of route is always the external (20) ▪ N1 – NSSA area, type 7 LSA, external cost plus the internal cost to reach the route ▪ N2 – NSSA area, type 7 LSA, cost of route is always the external (20)		
router rip network 10.2.0.0 redistribute static redistribute eigrp[metric] redistribute ospf[metric] redistribute isis default-metric [metric]	router eigrp [AS] network 10.1.0.0 redistribute static redistribute ospf [metric] redistribute rip redistribute isis default-metric 10000 100 255 1 1500	router ospf [process ID] network 10.10.0.0 0.0.255.255 area 0 redistribute static metric 200 [subnets] redistribute rip metric 200 [subnets] redistribute eigrp 1 metric 100 [subnets] redistribute isis metric 10 [subnets]	router isis network 49.1234.1111.1111.1111.00 redistribute static redistribute rip metric 20 redistribute eigrp 1 metric 20 redistribute ospf 1 metric 20	router bgp [as] redistribute static redistribute eigrp [as] redistribute isis ? WORD ISO routing area tag clns Redistribution of OSI dynamic routes ip Redistribution of IP dynamic routes level-1 IS-IS level-1 routes only level-1-2 IS-IS level-1 and level-2 routes level-2 IS-IS level-2 routes only metric Metric for redistributed routes route-map Route map reference

				<cr> redistribute ospf [pid] match redistribute ospf 1 match ? external Redistribute OSPF external routes internal Redistribute OSPF internal routes nssa-external Redistribute OSPF NSSA external routes

2.3.i Implement, optimize and troubleshoot manual and auto summarization with any routing protocol

Routes learned from other routing protocols can be summarized.
The metric used to advertise the summary is the smallest metric of all the more specific routes.
This command helps reduce the size of the routing table.

RIP
By default RIP will automatically summarize subnet routes into network level routes

```
router rip
version 2
[no] auto-summary
network x.x.x.x
```

Manual summarization can be done at an interface level.
The summary entry will be entered into the RIP database

```
R1(config-if)#ip summary-address rip x.x.x.x 255.255.255.0
```

EIGRP
Auto summary is enabled in EIGRP by default.

Summary routers are given an administrative distance of 5

Manual summarization can be performed at any point in the network

```
R1(config-if)#ip summary-address eigrp 1 0.0.0.0/0
```

OSPF
OSPF does not have an auto summaization feature and can only perform summarization at specific points in the network.

OSPF requires at an area has a the same database on each router in that area which means summarization can only be done at area borders.

There are 2 types of summarization

Area range – ABR, summarize internal routes as they get passed to another area as a type 3 interarea LSA
Summary address – ASBR, summarize external routes before injecting them into the OSPF domain at type 5 external LSA's

BGP
BGP has multiple different ways to summarize routes

network w/ static route
Aggregate-address

2.3.j Implement, optimize and troubleshoot policy-based routing

Policy based routing is being able to manipulate the path of traffic from what is being directed from the RIB.

Typically, this is implemented using route maps and match a source of traffic and changing the destination path.

All packets received on an interface with PBR enabled are passed through enhanced packet filters known as route maps. The route maps used by PBR dictate the policy, determining to where the packets are forwarded.

- Route maps are composed of statements. The route map statements can be marked as permit or deny, and they are interpreted in the following ways
- If the packets do not match any route map statements, then all the set clauses are applied.
- If a statement is marked as deny, the packets meeting the match criteria are sent back through the normal forwarding channels and destination-based routing is performed.
- If the statement is marked as permit and the packets do not match any route map statements, the packets are sent back through the normal forwarding channels and destination-based routing is performed.

For traffic originated outside of the router

```
interface GigabitEthernet0/1
 ip policy route-map PBR

!
route-map PBR permit 10
 match ip address prefix-list LOOPBACK
 set ip next-hop 2.2.2.2
```

For traffic originated from the router

```
ip local policy route-map LOCAL_PBR
```

2.3.k Identify and troubleshoot sub-optimal routing

Sub-optimal routing can occur when using PBR as it may be asymetric due to not capturing traffic in both directions.
Asymetric routing is when traffic goes out one interface and is recieved back on a different interface.
This may not cause a problem in normal cases, but if a firewall which requires state information, this will cause traffic to be dropped.

You can identify this by checking the routing table, cef table and using traceroute to follow the traffic path.

2.3.l Implement and troubleshoot bidirectional forwarding detection

BFD is a detection protocol designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols.

BFD is not tied to any routing protocol. A routing protocol can utilize BFD to help detect neighbor failures faster. Enabled at an interface level. Must be configured on both ends of the link

CEF and IP routing is required on the router

Used to detect faults between 2 nodes connected by a link

- Low overhead detection on physical media that doesn't support failure detection
- 3 way handshake to establish session
- Supports authentication
- Must be explicitly configured

Modes

- Asynchronous
- ▪ Periodically send Hellos between each other
- ▪ If number of packets are not received, session is considered down

- Demand
- ▪ No hellos are exchanged after session is established
- ▪ Assumed endpoints have another way to verify connectivity

Echo mode is enabled by default, works with asynchronous BFD

Config

```
interface [interface]
bfd internal [ms] mix-rx [ms] multiplier [interval]
bfd interval 50 min_rx 50 multiplier 5

router bgp [as]
neighbor [ip] fall-over bfd

router eigrp [as]
bfd all-interfaces

router ospf [pid]
```

bfd all-interfaces

2.3.m Implement and troubleshoot loop prevention mechanisms

2.3.m [i] Route tagging, filtering

Loops can occur when you are redistributing bidirectionally at 2 different points in the network
Route Tagging, Filtering
Route tagging is the process of adding an indentifier (tag) to a set of prefixes. This tag can then be called in a route map.

Route tagging and filtering

- Route redistribution can lead to loops
- Metrics are normally destroyed when moving between protocols
- In some scenarios, a redistributed route may look better than a native route, leading to a loop
- Route tagging allows you to mark a route so that it can be identified even after the metric has been changed
- You can prevent learning your own route back again through another protocol
- Basic Steps:
    - When redistributing out, apply a tag to all routes
    - When redistributing in, filter any routes containing your tag from step #1
    - Perform the process on all border routers

Note: use show ip route to view the tag

2.3.m [ii] Split horizon

Split horizon is used by distance vector routing protocols. This prevents a router from sending a route back out an interface that the route was learned from. The only time you would disable this is in a hub/spoke setup where the hub would need to send routes learned back to other spokes.

- A loop prevention mechanism built in to Cisco routers to assist EIGRP with loop prevention
- Prevents a router from advertising a route out the same interface it was learned on
- Enabled by default
- May need to be overridden in certain cases

2.3.m [iii] Route poisoning

Used by distance vector routing protocols. Once you learn of a route through an interface, advertise it as unreachable back through that same interface

- When a route fails, a router will republish the route with the maximum metric (unreachable) set
- This prevents the router from re-learning its own route back again
- Routers learning a poisoned route will remove it from their routing table without waiting for the hold-down timers to expire due to the increased metric
- Only occurs when it is a “graceful” failure
    - e.g. Not during power loss
    - The router doesn’t have time to transmit the poisoned routes

2.3.n Implement and troubleshoot routing protocol authentication

2.3.n [i] MD5

2.3.n [ii] key-chain

2.3.n [iii] EIGRP HMAC SHA2-256bit

2.3.n [iv] OSPFv2 SHA1-196bit

2.3.n [v] OSPFv3 IPsec authentication

Routing protocols can be configured to authenticate their neighbors to add some security to Peer Device routing with.

RIP and EIGRP utilize key chains for authentication and can be configured per interface.

The lowest number valid key will be used for authentication and is ordered in a top-down for which key will be used.

RIP Config

RIP can authenticate with text or md5

Do not add the mode if you want to use text authentication

```
key chain [name]
 key [#]
 key-string [string]
 accept-lifetime [start] {infinite | end-time | duration seconds}
 send-lifetime [start] {infinite | end-time | duration seconds}

interface [interface]
 ip rip authentication key-chain [keychain name]
 ip rip authentication mode md5
```

EIGRP Config

```
key chain [name]
 key [#]
 key-string [string]
 accept-lifetime [start] {infinite | end-time | duration seconds}
 send-lifetime [start] {infinite | end-time | duration seconds}

interface [interface]
 ip authentication mode eigrp [as] md5
 ip authentication key-chain eigrp [as] [key-chain]
```

Named Mode

```
router eigrp [name]
 af-interface default
 authentication key-chain [keychain name]
 authentication mode [hmac-sha-256 password | md5]
```

OSPFv2

There are 3 authentication types for OSPF, null, text and md5

- Null Authentication—This is also called Type 0 and it means no authentication information is included in the packet header. It is the default
- Plain Text Authentication—This is also called Type 1 and it uses simple clear-text passwords
- MD5 Authentication—This is also called Type 2 and it uses MD5 cryptographic passwords.
- Plain Text

```
interface [interface]
 ip ospf authentication-key [password]

router ospf [pid]
 area [area] authentication
```

MD5

```
interface [interface]
 ip ospf message-digest-key [#] [password]

router ospf [pid]
 area [area] authentication message-digest

show ip ospf interface [interface]
```

OSPFv3

OSPFv3 uses IPSec to enable authentication

```
interface [interface]
 ospfv3   authentication {ipsec spi} {md5 | sha1}{ key-encryption-type key} | null
ipv6 ospf authentication {null | ipsec spi spi authentication-algorithm [key-encryption-type] [key]}

ipv6 router ospf [pid]
 area [area] authentication ipsec spi [spi authentication-algorithm] [key-encryption-type] [key]
```

2.4 RIP v2

2.4.a Implement and troubleshoot RIPv2

RIP v2 – Version one is now considered legacy and is a Class-full protocol.
Not a very feature rich protocol. RIP is the most basic of all dynamic routing protocols.

Distance Vector IGP
- Uses split-horizon, poison reverse, and count to infinity (loop prevention)
-UDP port 520 for transport
– Metric is hop count
– Metric 16 is infinite metric
– Timers

- Update timer: 30 seconds
- Invalid timer: 180 seconds
- Hold-down timer: 180 seconds
- Flush timer: 240 seconds

– Supports authentication

- Plain text
- MD5

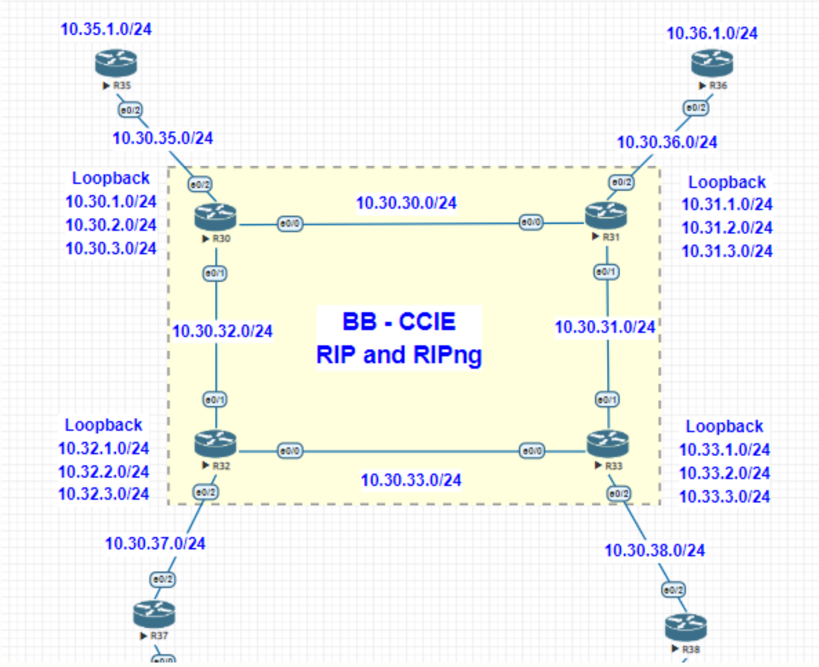
– AD is 120

Two version:

RIPv1
- Classful
- Updates as broadcast

RIPv2
-Classless
-updates as multicast to 224.0.0.9

Here is the Topology we use for Test RIP and RIPng



RIPv2

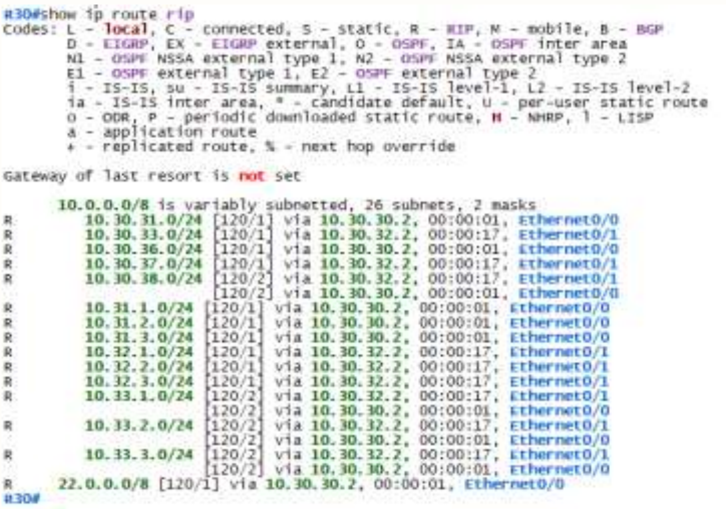
Basic Configuration:

R30 / R31 / R32 / R33

router rip
  version 2
  network 10.0.0.0.
  network x.x.x.x - respected connected network.

Check the output

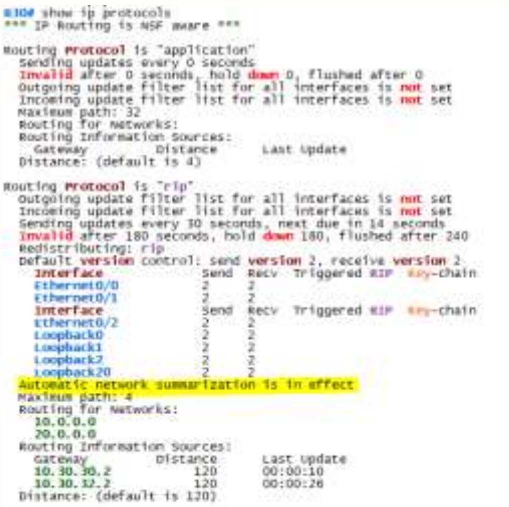
#show ip route rip



```
R30#show ip route rip
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 26 subnets, 2 masks
R        10.30.31.0/24 [120/1] via 10.30.30.2, 00:00:01, Ethernet0/0
R        10.30.33.0/24 [120/1] via 10.30.32.2, 00:00:17, Ethernet0/1
R        10.30.36.0/24 [120/1] via 10.30.30.2, 00:00:01, Ethernet0/0
R        10.30.37.0/24 [120/1] via 10.30.32.2, 00:00:17, Ethernet0/1
R        10.30.38.0/24 [120/2] via 10.30.32.2, 00:00:17, Ethernet0/1
                       [120/2] via 10.30.30.2, 00:00:01, Ethernet0/0
R        10.31.1.0/24 [120/1] via 10.30.30.2, 00:00:01, Ethernet0/0
R        10.31.2.0/24 [120/1] via 10.30.30.2, 00:00:01, Ethernet0/0
R        10.31.3.0/24 [120/1] via 10.30.30.2, 00:00:01, Ethernet0/0
R        10.32.1.0/24 [120/1] via 10.30.32.2, 00:00:17, Ethernet0/1
R        10.32.2.0/24 [120/1] via 10.30.32.2, 00:00:17, Ethernet0/1
R        10.32.3.0/24 [120/1] via 10.30.32.2, 00:00:17, Ethernet0/1
R        10.33.1.0/24 [120/2] via 10.30.32.2, 00:00:17, Ethernet0/1
                      [120/2] via 10.30.30.2, 00:00:01, Ethernet0/0
R        10.33.2.0/24 [120/2] via 10.30.32.2, 00:00:17, Ethernet0/1
                      [120/2] via 10.30.30.2, 00:00:01, Ethernet0/0
R        10.33.3.0/24 [120/2] via 10.30.32.2, 00:00:17, Ethernet0/1
                      [120/2] via 10.30.30.2, 00:00:01, Ethernet0/0
R     22.0.0.0/8 [120/1] via 10.30.30.2, 00:00:01, Ethernet0/0
R30#
```

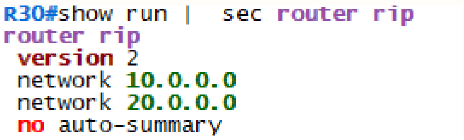
It learned all the networks from other rip enabled devices.

By default, RIP does the Auto Summary. (check **with show ip protocols**)



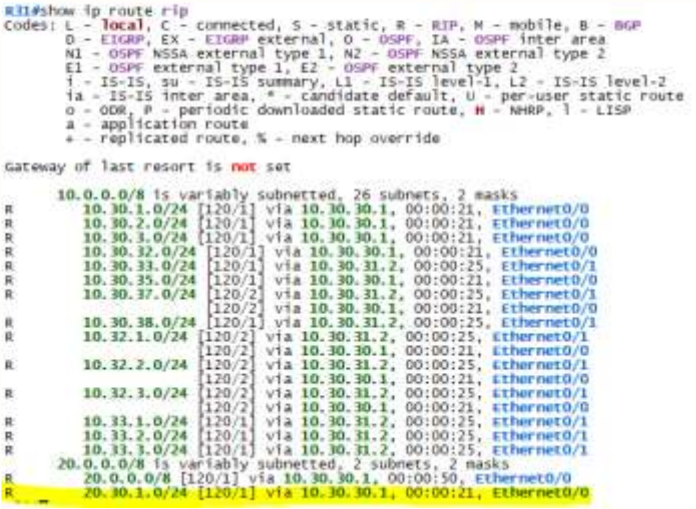
```
R30# show ip protocols
*** IP Routing is NSF aware ***

Routing Protocol is "application"
  Sending updates every 0 seconds
  Invalid after 0 seconds, hold down 0, flushed after 0
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Maximum path: 32
  Routing for Networks:
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: (default is 4)

Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 14 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Ethernet0/0           2     2
    Ethernet0/1           2     2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Ethernet0/2           2     2
    Loopback0             2     2
    Loopback1             2     2
    Loopback2             2     2
    Loopback20            2     2
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
    20.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.30.30.2           120      00:00:10
    10.30.32.2           120      00:00:26
  Distance: (default is 120)
```

Disable Auto Sumamry and Lets looks the RIP route.

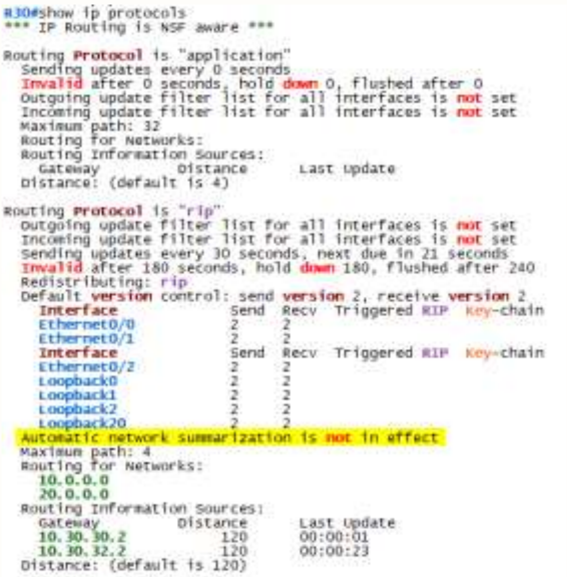


```
R30#show run |  sec router rip
router rip
 version 2
 network 10.0.0.0
 network 20.0.0.0
 no auto-summary
```

Let's look at the Route (now we can see the exact network announced other rip enabled routers)

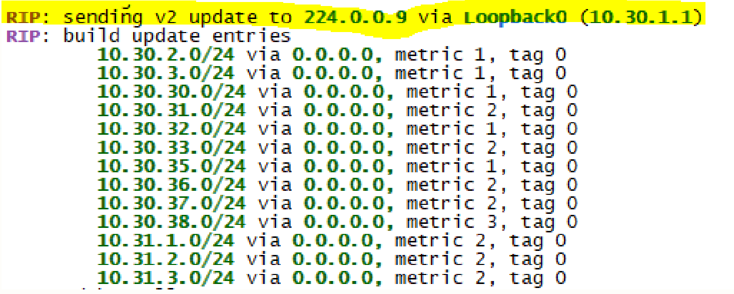


```
R31#show ip route rip
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 26 subnets, 2 masks
R        10.30.1.0/24 [120/1] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.30.2.0/24 [120/1] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.30.3.0/24 [120/1] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.30.32.0/24 [120/1] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.30.33.0/24 [120/1] via 10.30.31.2, 00:00:25, Ethernet0/1
R        10.30.35.0/24 [120/1] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.30.37.0/24 [120/2] via 10.30.31.2, 00:00:25, Ethernet0/1
                       [120/2] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.30.38.0/24 [120/1] via 10.30.31.2, 00:00:25, Ethernet0/1
R        10.32.1.0/24 [120/2] via 10.30.31.2, 00:00:25, Ethernet0/1
                      [120/2] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.32.2.0/24 [120/2] via 10.30.31.2, 00:00:25, Ethernet0/1
                      [120/2] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.32.3.0/24 [120/2] via 10.30.31.2, 00:00:25, Ethernet0/1
                      [120/2] via 10.30.30.1, 00:00:21, Ethernet0/0
R        10.33.1.0/24 [120/1] via 10.30.31.2, 00:00:25, Ethernet0/1
R        10.33.2.0/24 [120/1] via 10.30.31.2, 00:00:25, Ethernet0/1
R        10.33.3.0/24 [120/1] via 10.30.31.2, 00:00:25, Ethernet0/1
      20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
R        20.0.0.0/8 [120/1] via 10.30.30.1, 00:00:50, Ethernet0/0
R        20.30.1.0/24 [120/1] via 10.30.30.1, 00:00:21, Ethernet0/0
```



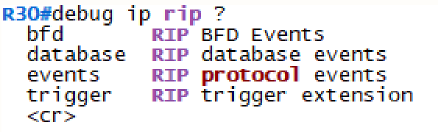
```
R30#show ip protocols
*** IP Routing is NSF aware ***

Routing Protocol is "application"
  Sending updates every 0 seconds
  Invalid after 0 seconds, hold down 0, flushed after 0
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Maximum path: 32
  Routing for Networks:
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: (default is 4)

Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 21 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Ethernet0/0           2     2
    Ethernet0/1           2     2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Ethernet0/2           2     2
    Loopback0             2     2
    Loopback1             2     2
    Loopback2             2     2
    Loopback20            2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
    20.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.30.30.2           120      00:00:01
    10.30.32.2           120      00:00:23
  Distance: (default is 120)
```

Rip updates learn from Multicast Address 224.0.0.9



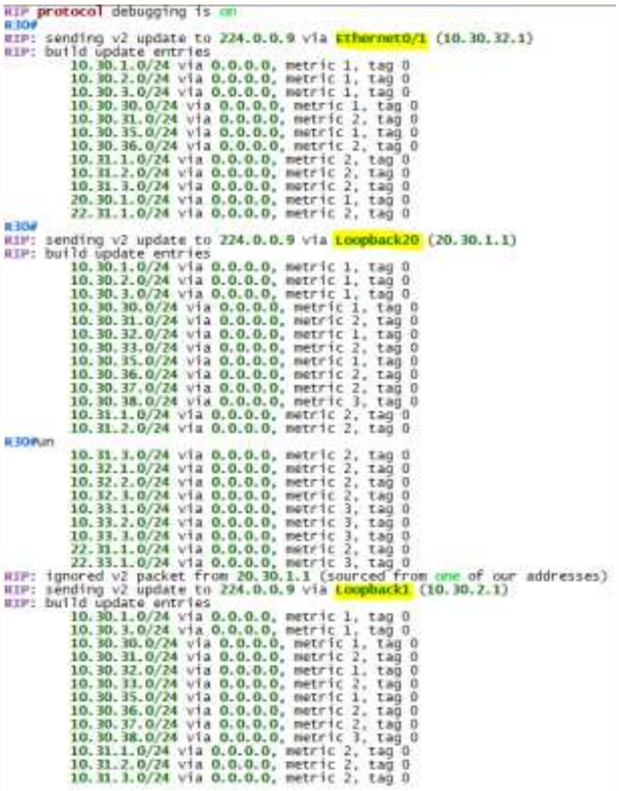
```
RIP: sending v2 update to 224.0.0.9 via Loopback0 (10.30.1.1)
RIP: build update entries
        10.30.2.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.3.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.30.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.31.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.32.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.33.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.35.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.36.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.37.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.38.0/24 via 0.0.0.0, metric 3, tag 0
        10.31.1.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.2.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.3.0/24 via 0.0.0.0, metric 2, tag 0
```

Available Debug option for RIP:



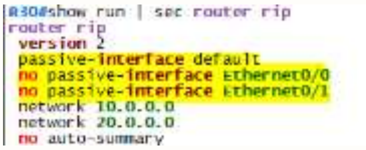
```
R30#debug ip rip ?
  bfd        RIP BFD Events
  database   RIP database events
  events     RIP protocol events
  trigger    RIP trigger extension
  <cr>
```

By Default, RIP send all the updates all the connected interface, some interface where there is no updates required, but still RIP send that information. To Optimise this, we need to make interface as passive interface so that update will not send updates.



```
RIP protocol debugging is on
R30#
RIP: sending v2 update to 224.0.0.9 via Ethernet0/1 (10.30.32.1)
RIP: build update entries
        10.30.1.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.2.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.3.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.30.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.31.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.35.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.36.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.1.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.2.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.3.0/24 via 0.0.0.0, metric 2, tag 0
        20.30.1.0/24 via 0.0.0.0, metric 1, tag 0
        22.31.1.0/24 via 0.0.0.0, metric 2, tag 0
R30#
RIP: sending v2 update to 224.0.0.9 via Loopback20 (20.30.1.1)
RIP: build update entries
        10.30.1.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.2.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.3.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.30.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.31.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.32.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.33.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.35.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.36.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.37.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.38.0/24 via 0.0.0.0, metric 3, tag 0
        10.31.1.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.2.0/24 via 0.0.0.0, metric 2, tag 0
R30#un
        10.31.3.0/24 via 0.0.0.0, metric 2, tag 0
        10.32.1.0/24 via 0.0.0.0, metric 2, tag 0
        10.32.2.0/24 via 0.0.0.0, metric 2, tag 0
        10.32.3.0/24 via 0.0.0.0, metric 2, tag 0
        10.33.1.0/24 via 0.0.0.0, metric 3, tag 0
        10.33.2.0/24 via 0.0.0.0, metric 3, tag 0
        10.33.3.0/24 via 0.0.0.0, metric 3, tag 0
        22.31.1.0/24 via 0.0.0.0, metric 2, tag 0
        22.33.1.0/24 via 0.0.0.0, metric 3, tag 0
RIP: ignored v2 packet from 20.30.1.1 (sourced from one of our addresses)
RIP: sending v2 update to 224.0.0.9 via Loopback1 (10.30.2.1)
RIP: build update entries
        10.30.1.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.3.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.30.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.31.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.32.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.33.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.35.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.36.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.37.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.38.0/24 via 0.0.0.0, metric 3, tag 0
        10.31.1.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.2.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.3.0/24 via 0.0.0.0, metric 2, tag 0
```

To disable this, we need to configure passive interface config

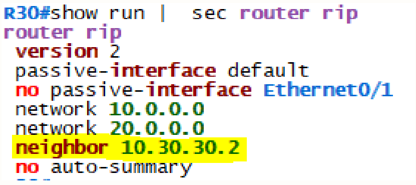


```
R30#show run | sec router rip
router rip
 version 2
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 network 10.0.0.0
 network 20.0.0.0
 no auto-summary
```

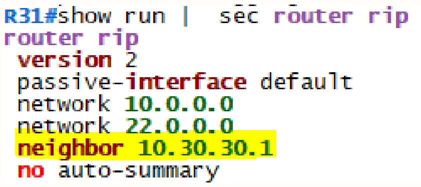
Here on R30 we only need Eth 0/0 and 0/1 have RIP neighbours.

RIP now using Multicast address to update the routes (224.0.0.9)

Instead of we can use Unicast to send updates towards R31

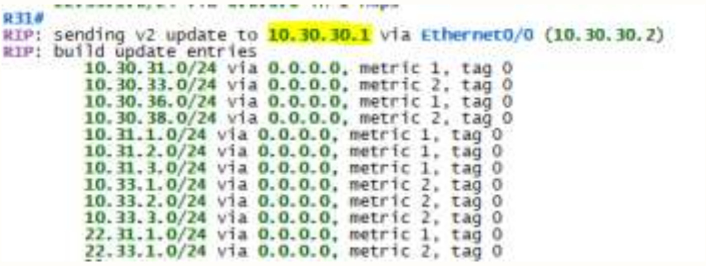


```
R30#show run |  sec router rip
router rip
 version 2
 passive-interface default
 no passive-interface Ethernet0/1
 network 10.0.0.0
 network 20.0.0.0
 neighbor 10.30.30.2
 no auto-summary
```



```
R31#show run |  sec router rip
router rip
 version 2
 passive-interface default
 network 10.0.0.0
 network 22.0.0.0
 neighbor 10.30.30.1
 no auto-summary
```

Now you can verify by enabling the Debug, you can observe unicast ip address to send updates to rip neighbour.



```
R31#
RIP: sending v2 update to 10.30.30.1 via Ethernet0/0 (10.30.30.2)
RIP: build update entries
        10.30.31.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.33.0/24 via 0.0.0.0, metric 2, tag 0
        10.30.36.0/24 via 0.0.0.0, metric 1, tag 0
        10.30.38.0/24 via 0.0.0.0, metric 2, tag 0
        10.31.1.0/24 via 0.0.0.0, metric 1, tag 0
        10.31.2.0/24 via 0.0.0.0, metric 1, tag 0
        10.31.3.0/24 via 0.0.0.0, metric 1, tag 0
        10.33.1.0/24 via 0.0.0.0, metric 2, tag 0
        10.33.2.0/24 via 0.0.0.0, metric 2, tag 0
        10.33.3.0/24 via 0.0.0.0, metric 2, tag 0
        22.31.1.0/24 via 0.0.0.0, metric 1, tag 0
        22.33.1.0/24 via 0.0.0.0, metric 2, tag 0
```

Authentication:

- Plain text
- MD5

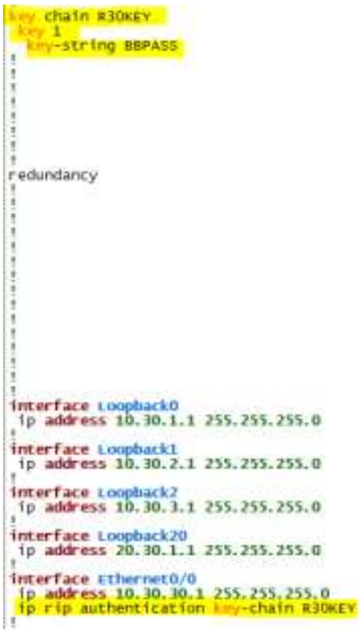
  - Create key chain
  - Apply to an interface

Create Key chain

  - Text authentication
  - Key ID and Key chain will be exchanged with Neighbour to check authentication
  - Key chain is locally significant.
  - No authentication mode means clear text

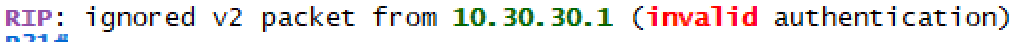
Let’s configuration authentication between R30 and R31 connected interface

R30 configure with authentication and R31 not configured, you can see the error as below



```
key chain R30KEY
 key 1
  key-string BBPASS
!
!
!
!
!
!
!
!
redundancy
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.30.1.1 255.255.255.0
!
interface Loopback1
 ip address 10.30.2.1 255.255.255.0
!
interface Loopback2
 ip address 10.30.3.1 255.255.255.0
!
interface Loopback20
 ip address 20.30.1.1 255.255.255.0
!
interface Ethernet0/0
 ip address 10.30.30.1 255.255.255.0
 ip rip authentication key-chain R30KEY
!
```

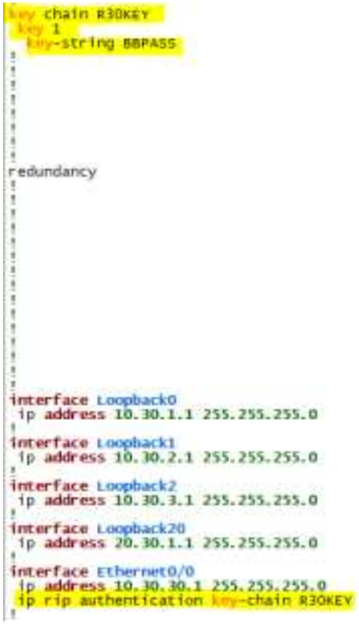
R31 have authentication log since it was not configured.



```
RIP: ignored v2 packet from 10.30.30.1 (invalid authentication)
```

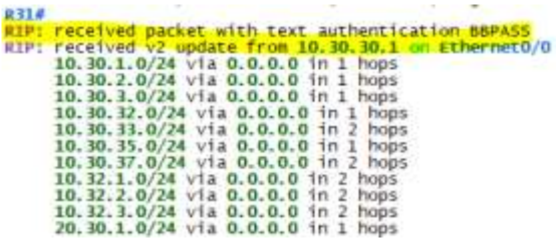
Now we configure same with R31 so authentication will be success and exchange the routes.

R31 config :



```
key chain R30KEY
 key 1
  key-string BBPASS
!
!
!
!
!
!
!
!
redundancy
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.30.1.1 255.255.255.0
!
interface Loopback1
 ip address 10.30.2.1 255.255.255.0
!
interface Loopback2
 ip address 10.30.3.1 255.255.255.0
!
interface Loopback20
 ip address 20.30.1.1 255.255.255.0
!
interface Ethernet0/0
 ip address 10.30.30.1 255.255.255.0
 ip rip authentication key-chain R30KEY
!
```

Now check the debug logs :

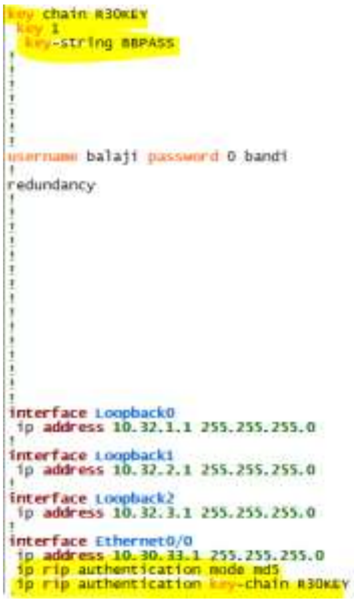


```
R31#
RIP: received packet with text authentication BBPASS
RIP: received v2 update from 10.30.30.1 on Ethernet0/0
     10.30.1.0/24 via 0.0.0.0 in 1 hops
     10.30.2.0/24 via 0.0.0.0 in 1 hops
     10.30.3.0/24 via 0.0.0.0 in 1 hops
     10.30.32.0/24 via 0.0.0.0 in 1 hops
     10.30.33.0/24 via 0.0.0.0 in 2 hops
     10.30.35.0/24 via 0.0.0.0 in 1 hops
     10.30.37.0/24 via 0.0.0.0 in 2 hops
     10.32.1.0/24 via 0.0.0.0 in 2 hops
     10.32.2.0/24 via 0.0.0.0 in 2 hops
     10.32.3.0/24 via 0.0.0.0 in 2 hops
     20.30.1.0/24 via 0.0.0.0 in 1 hops
```

MD5 authentication:

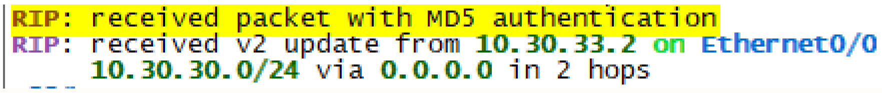
Same as above procedure, and we need to mention under interface authentication MD5

R32 to R33 connected interface configure MD5 authentication.



```
key chain R30KEY
 key 1
  key-string BBPASS
!
!
!
!
!
!
!
username balaji password 0 bandi
!
redundancy
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.32.1.1 255.255.255.0
!
interface Loopback1
 ip address 10.32.2.1 255.255.255.0
!
interface Loopback2
 ip address 10.32.3.1 255.255.255.0
!
interface Ethernet0/0
 ip address 10.30.33.1 255.255.255.0
 ip rip authentication mode md5
 ip rip authentication key-chain R30KEY
```

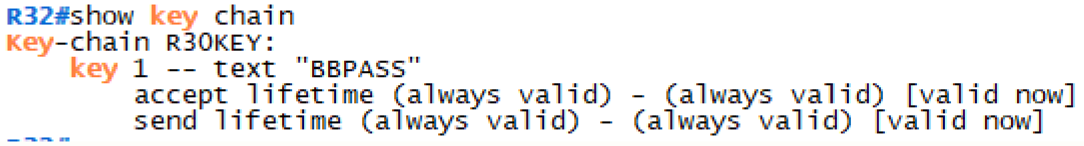
Debug ip rip will see MD5 authentication, rather clear text password in the logs.



```
RIP: received packet with MD5 authentication
RIP: received v2 update from 10.30.33.2 on Ethernet0/0
     10.30.30.0/24 via 0.0.0.0 in 2 hops
```

Whitespace counts as a valid character for key chain authentication. Use the show keychain command to ensure that a white space is not appended at the end of the authentication string.

Check the key chain with show commands



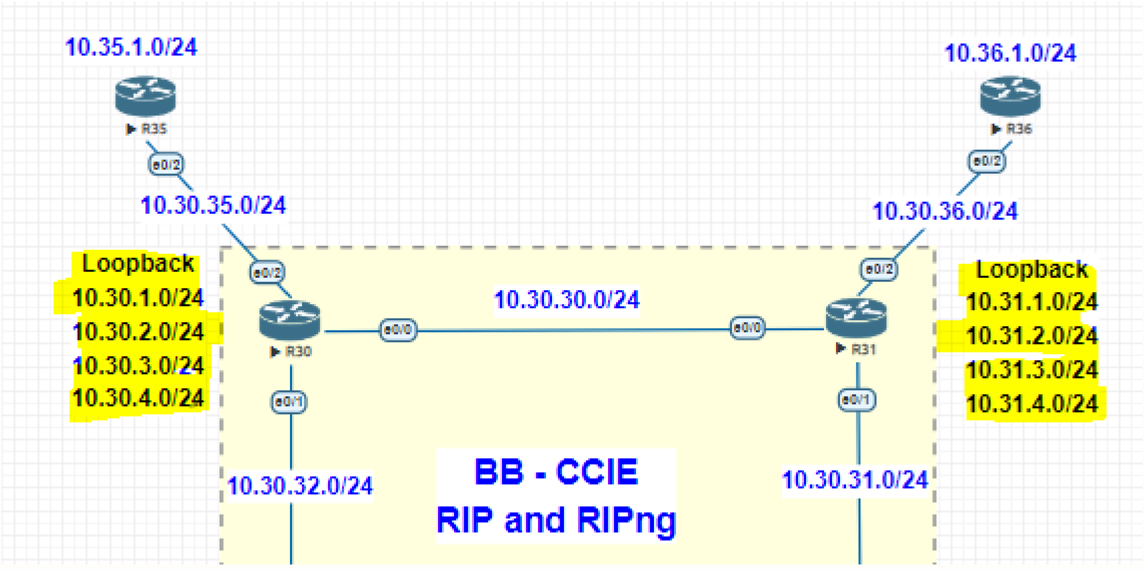
```
R32#show key chain
Key-chain R30KEY:
    key 1 -- text "BBPASS"
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (always valid) - (always valid) [valid now]
```

Summarisation:

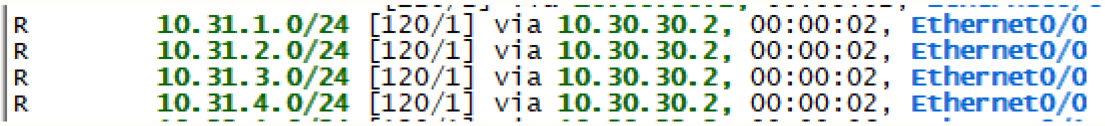
RIPv2 have default auto summarisation enabled – which will announce major network like /8
So to optimally summarise we need to manually summarise the network address space.
This will have advantage for the cut down the routing table entries.

Example for the manual summarisation in R30 and R31

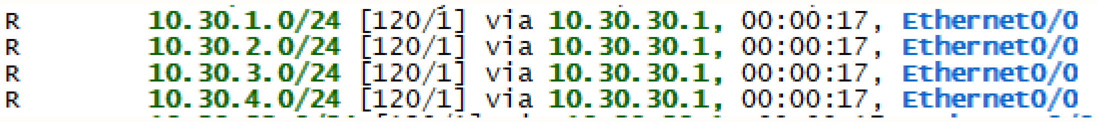
Here is the example network from R30 and R31



Now we look the Routing table before we summarise on R30 and R31



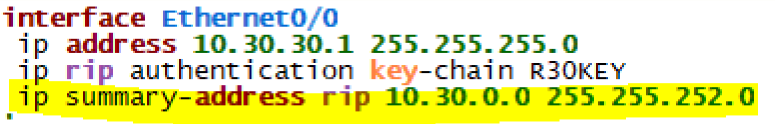
```
R        10.31.1.0/24 [120/1] via 10.30.30.2, 00:00:02, Ethernet0/0
R        10.31.2.0/24 [120/1] via 10.30.30.2, 00:00:02, Ethernet0/0
R        10.31.3.0/24 [120/1] via 10.30.30.2, 00:00:02, Ethernet0/0
R        10.31.4.0/24 [120/1] via 10.30.30.2, 00:00:02, Ethernet0/0
```



```
R        10.30.1.0/24 [120/1] via 10.30.30.1, 00:00:17, Ethernet0/0
R        10.30.2.0/24 [120/1] via 10.30.30.1, 00:00:17, Ethernet0/0
R        10.30.3.0/24 [120/1] via 10.30.30.1, 00:00:17, Ethernet0/0
R        10.30.4.0/24 [120/1] via 10.30.30.1, 00:00:17, Ethernet0/0
```

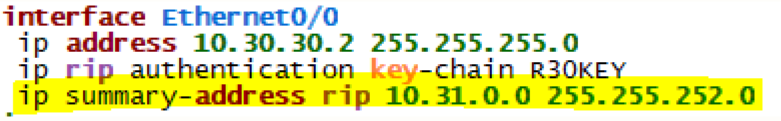
Now we configure Summary on R30 and R31 as below :

R30



```
interface Ethernet0/0
 ip address 10.30.30.1 255.255.255.0
 ip rip authentication key-chain R30KEY
 ip summary-address rip 10.30.0.0 255.255.252.0
```

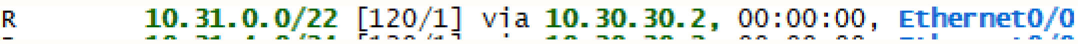
R31



```
interface Ethernet0/0
 ip address 10.30.30.2 255.255.255.0
 ip rip authentication key-chain R30KEY
 ip summary-address rip 10.31.0.0 255.255.252.0
```

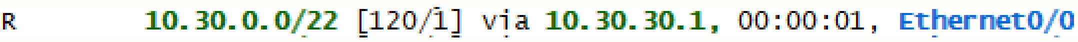
Now we check the routing table **show ip route rip** to verify on

R30



```
R        10.31.0.0/22 [120/1] via 10.30.30.2, 00:00:00, Ethernet0/0
```

R31



```
R        10.30.0.0/22 [120/1] via 10.30.30.1, 00:00:01, Ethernet0/0
```

Filtering:

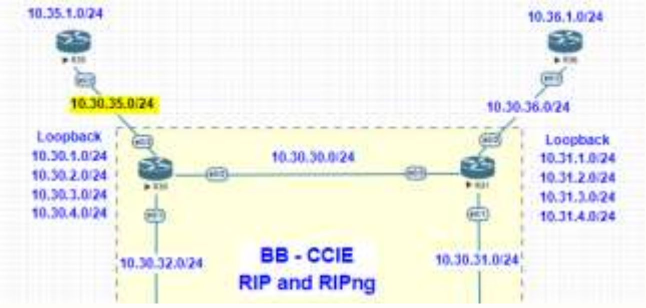
RIPv2 Filtering can be done outbound or inbound depends on requirement.
Filtering is done on the Interface level, you can also do Filtering globally.

RIPv2 there are several methods to filter routes received and advertised to other neighbours.

-Filtering using passive interface
-Filtering using prefix-lists
-Filtering using Standard access-lists
-Filtering using Extended access-lists
-Filtering using Administrative Distance
-Filtering using Offset Lists

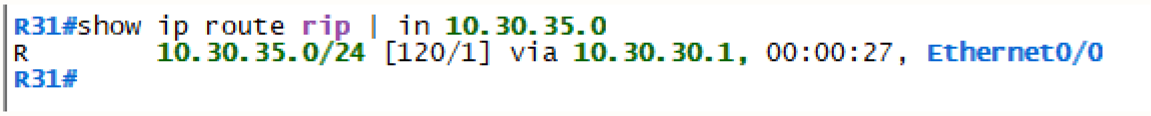
Example: Filtering using ACL

From R30 we will use ACL to stop sending routes 10.30.35.0/24 to R31 using interface Eth 0/0



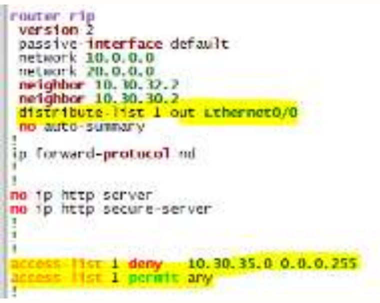
Before we apply ACL configuration lets verify the routing table.

R31 have route learned from R30



```
R31#show ip route rip | in 10.30.35.0
R        10.30.35.0/24 [120/1] via 10.30.30.1, 00:00:27, Ethernet0/0
R31#
```

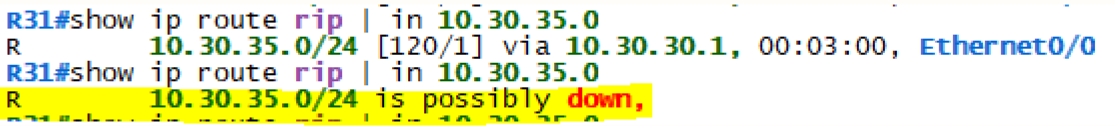
Now we configure Access-list and apply to RIP routing protocol as below :



```
router rip
 version 2
 passive-interface default
 network 10.0.0.0
 network 20.0.0.0
 neighbor 10.30.32.2
 neighbor 10.30.30.2
 distribute-list 1 out Ethernet0/0
 no auto-summary
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
!
access-list 1 deny   10.30.35.0 0.0.0.255
access-list 1 permit any
!
```

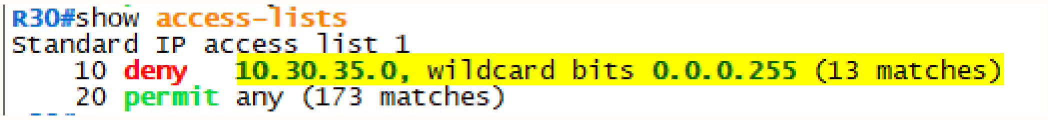
Now we will check the routing table on R1 see that filter working.

To get updates clear (you can do “clear ip route *” to get update quicket, if not RIP take 3min to get updates. After 3min the routes are down.



```
R31#show ip route rip | in 10.30.35.0
R        10.30.35.0/24 [120/1] via 10.30.30.1, 00:03:00, Ethernet0/0
R31#show ip route rip | in 10.30.35.0
R        10.30.35.0/24 is possibly down,
R31#show ip route rip | in 10.30.35.0
```

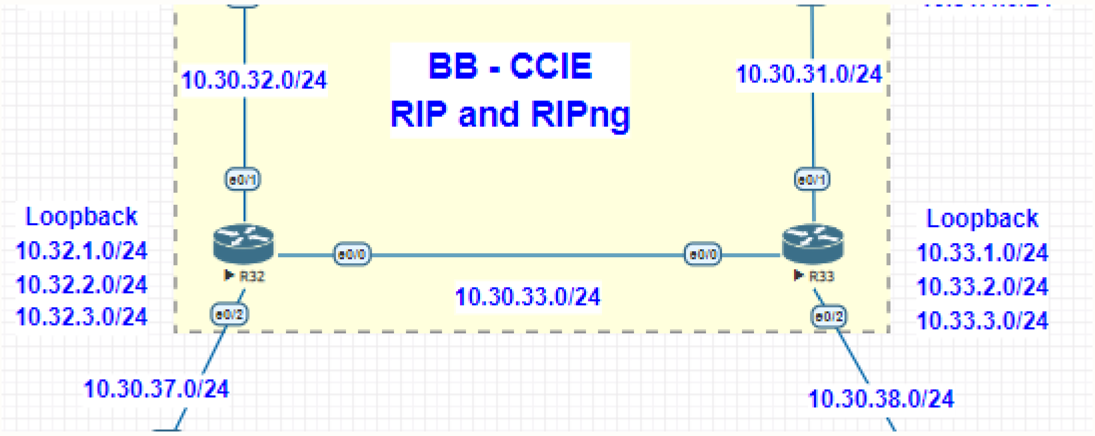
You can verify the Access-list hits

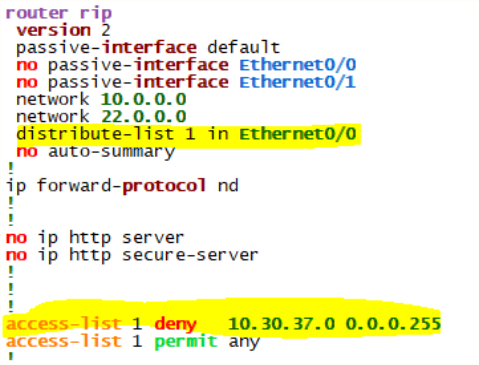


```
R30#show access-lists
Standard IP access list 1
    10 deny   10.30.35.0, wildcard bits 0.0.0.255 (13 matches)
    20 permit any (173 matches)
```

Now we will look incoming blocking using ACL

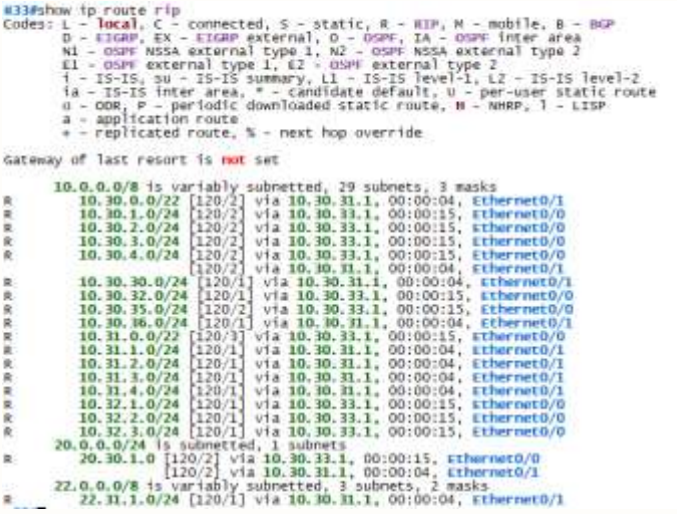
R33 looking to block this network to coming in 10.30.37.0/24



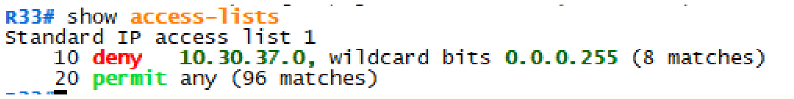


```
router rip
 version 2
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 network 10.0.0.0
 network 22.0.0.0
 distribute-list 1 in Ethernet0/0
 no auto-summary
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
!
access-list 1 deny   10.30.37.0 0.0.0.255
access-list 1 permit any
!
```

No networking routes related to 10.30.37.0/24



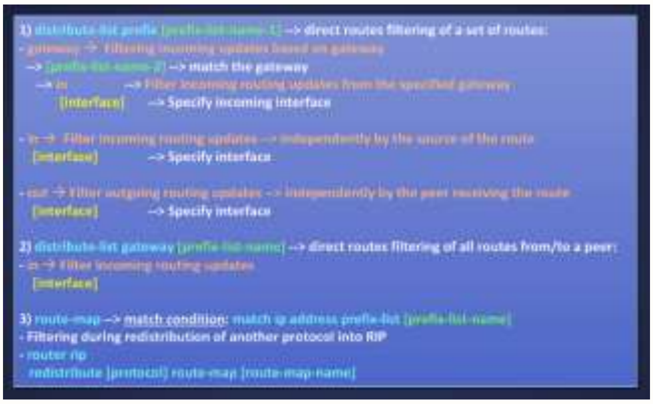
```
R33#show ip route rip
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 29 subnets, 3 masks
R        10.30.0.0/22 [120/2] via 10.30.31.1, 00:00:04, Ethernet0/1
R        10.30.1.0/24 [120/2] via 10.30.33.1, 00:00:15, Ethernet0/0
R        10.30.2.0/24 [120/2] via 10.30.33.1, 00:00:15, Ethernet0/0
R        10.30.3.0/24 [120/2] via 10.30.33.1, 00:00:15, Ethernet0/0
R        10.30.4.0/24 [120/2] via 10.30.33.1, 00:00:15, Ethernet0/0
                      [120/2] via 10.30.31.1, 00:00:04, Ethernet0/1
R        10.30.30.0/24 [120/1] via 10.30.31.1, 00:00:04, Ethernet0/1
R        10.30.32.0/24 [120/1] via 10.30.33.1, 00:00:15, Ethernet0/0
R        10.30.35.0/24 [120/2] via 10.30.33.1, 00:00:15, Ethernet0/0
R        10.30.36.0/24 [120/1] via 10.30.31.1, 00:00:04, Ethernet0/1
R        10.31.0.0/22 [120/3] via 10.30.33.1, 00:00:15, Ethernet0/0
R        10.31.1.0/24 [120/1] via 10.30.31.1, 00:00:04, Ethernet0/1
R        10.31.2.0/24 [120/1] via 10.30.31.1, 00:00:04, Ethernet0/1
R        10.31.3.0/24 [120/1] via 10.30.31.1, 00:00:04, Ethernet0/1
R        10.31.4.0/24 [120/1] via 10.30.31.1, 00:00:04, Ethernet0/1
R        10.32.1.0/24 [120/1] via 10.30.33.1, 00:00:15, Ethernet0/0
R        10.32.2.0/24 [120/1] via 10.30.33.1, 00:00:15, Ethernet0/0
R        10.32.3.0/24 [120/1] via 10.30.33.1, 00:00:15, Ethernet0/0
      20.0.0.0/24 is subnetted, 1 subnets
R        20.30.1.0 [120/2] via 10.30.33.1, 00:00:15, Ethernet0/0
                   [120/2] via 10.30.31.1, 00:00:04, Ethernet0/1
      22.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
R        22.31.1.0/24 [120/1] via 10.30.31.1, 00:00:04, Ethernet0/1
```



```
R33# show access-lists
Standard IP access list 1
    10 deny   10.30.37.0, wildcard bits 0.0.0.255 (8 matches)
    20 permit any (96 matches)
```

Prefix-List :

With a prefix-list we can filter a set of routes specifying a range of mask-length.
RIP can use a prefix-list calling it in these ways:



1) distribute-list prefix [prefix-list-name-1] --> direct routes filtering of a set of routes:
- gateway → Filtering incoming updates based on gateway
  --> [prefix-list-name-2] --> match the gateway
    --> in --> Filter incoming routing updates from the specified gateway
      [interface] --> Specify incoming interface

- in → Filter incoming routing updates --> independently by the source of the route
  [interface] --> Specify interface

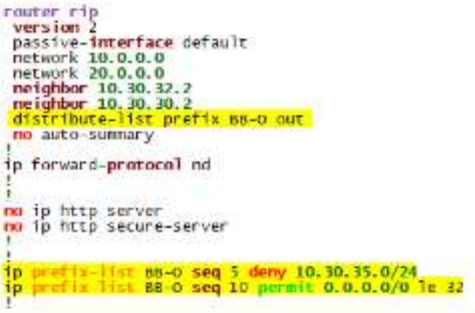
- out → Filter outgoing routing updates --> independently by the peer receiving the route
  [interface] --> Specify interface

2) distribute-list gateway [prefix-list-name] --> direct routes filtering of all routes from/to a peer:
- in → Filter incoming routing updates
  [interface]

3) route-map --> match condition: match ip address prefix-list [prefix-list-name]
- Filtering during redistribution of another protocol into RIP
- router rip
  redistribute [protocol] route-map [route-map-name]

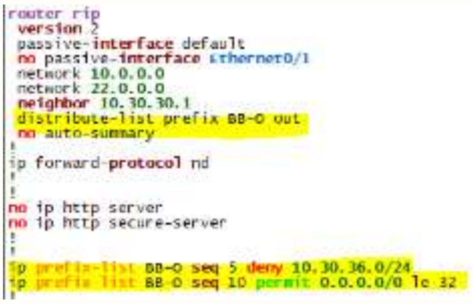
Now we block 10.30.35.0/24 from R30 and block 10.30.36.0/24 from R31

R30



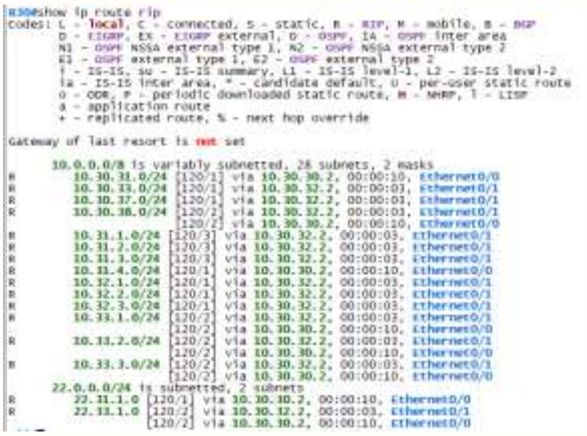
```
router rip
 version 2
 passive-interface default
 network 10.0.0.0
 network 20.0.0.0
 neighbor 10.30.32.2
 neighbor 10.30.30.2
 distribute-list prefix BB-O out
 no auto-summary
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
ip prefix-list BB-O seq 5 deny 10.30.35.0/24
ip prefix-list BB-O seq 10 permit 0.0.0.0/0 le 32
!
```

R31

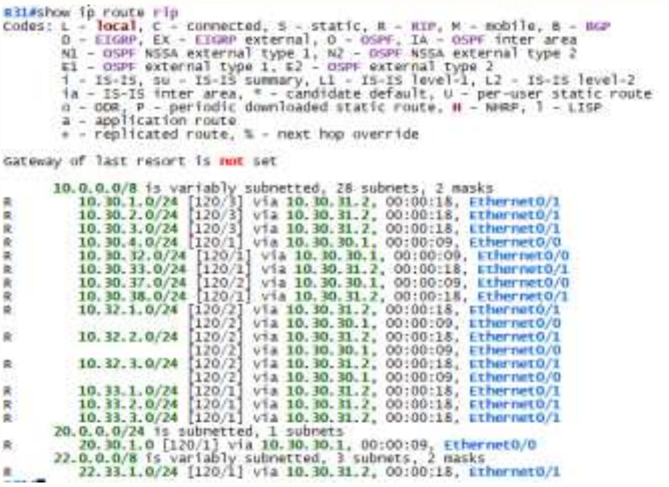


```
router rip
 version 2
 passive-interface default
 no passive-interface Ethernet0/1
 network 10.0.0.0
 network 22.0.0.0
 neighbor 10.30.30.1
 distribute-list prefix BB-O out
 no auto-summary
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
ip prefix-list BB-O seq 5 deny 10.30.36.0/24
ip prefix-list BB-O seq 10 permit 0.0.0.0/0 le 32
```

Verify on R30 and R31
On R30 we did not see network 10.30.36.0/24
Same On R31 we did not see network 10.30.35.0/24



```
R30#show ip route rip
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 28 subnets, 2 masks
R        10.30.31.0/24 [120/1] via 10.30.30.2, 00:00:10, Ethernet0/0
R        10.30.33.0/24 [120/1] via 10.30.32.2, 00:00:03, Ethernet0/1
R        10.30.37.0/24 [120/1] via 10.30.32.2, 00:00:03, Ethernet0/1
R        10.30.38.0/24 [120/2] via 10.30.32.2, 00:00:03, Ethernet0/1
                       [120/2] via 10.30.30.2, 00:00:10, Ethernet0/0
R        10.31.1.0/24 [120/3] via 10.30.32.2, 00:00:03, Ethernet0/1
R        10.31.2.0/24 [120/3] via 10.30.32.2, 00:00:03, Ethernet0/1
R        10.31.3.0/24 [120/3] via 10.30.32.2, 00:00:03, Ethernet0/1
R        10.31.4.0/24 [120/1] via 10.30.30.2, 00:00:10, Ethernet0/0
R        10.32.1.0/24 [120/1] via 10.30.32.2, 00:00:03, Ethernet0/1
R        10.32.2.0/24 [120/1] via 10.30.32.2, 00:00:03, Ethernet0/1
R        10.32.3.0/24 [120/1] via 10.30.32.2, 00:00:03, Ethernet0/1
R        10.33.1.0/24 [120/2] via 10.30.32.2, 00:00:03, Ethernet0/1
                      [120/2] via 10.30.30.2, 00:00:10, Ethernet0/0
R        10.33.2.0/24 [120/2] via 10.30.32.2, 00:00:03, Ethernet0/1
                      [120/2] via 10.30.30.2, 00:00:10, Ethernet0/0
R        10.33.3.0/24 [120/2] via 10.30.32.2, 00:00:03, Ethernet0/1
                      [120/2] via 10.30.30.2, 00:00:10, Ethernet0/0
      22.0.0.0/24 is subnetted, 2 subnets
R        22.31.1.0 [120/1] via 10.30.30.2, 00:00:10, Ethernet0/0
R        22.33.1.0 [120/2] via 10.30.32.2, 00:00:03, Ethernet0/1
                   [120/2] via 10.30.30.2, 00:00:10, Ethernet0/0
```

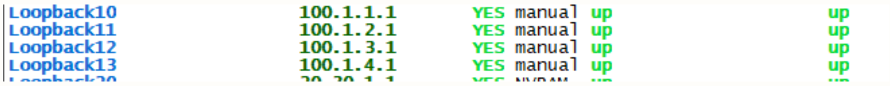


```
R31#show ip route rip
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 28 subnets, 2 masks
R        10.30.1.0/24 [120/3] via 10.30.31.2, 00:00:18, Ethernet0/1
R        10.30.2.0/24 [120/3] via 10.30.31.2, 00:00:18, Ethernet0/1
R        10.30.3.0/24 [120/3] via 10.30.31.2, 00:00:18, Ethernet0/1
R        10.30.4.0/24 [120/1] via 10.30.30.1, 00:00:09, Ethernet0/0
R        10.30.32.0/24 [120/1] via 10.30.30.1, 00:00:09, Ethernet0/0
R        10.30.33.0/24 [120/1] via 10.30.31.2, 00:00:18, Ethernet0/1
R        10.30.37.0/24 [120/2] via 10.30.30.1, 00:00:09, Ethernet0/0
R        10.30.38.0/24 [120/1] via 10.30.31.2, 00:00:18, Ethernet0/1
R        10.32.1.0/24 [120/2] via 10.30.31.2, 00:00:18, Ethernet0/1
                      [120/2] via 10.30.30.1, 00:00:09, Ethernet0/0
R        10.32.2.0/24 [120/2] via 10.30.31.2, 00:00:18, Ethernet0/1
                      [120/2] via 10.30.30.1, 00:00:09, Ethernet0/0
R        10.32.3.0/24 [120/2] via 10.30.31.2, 00:00:18, Ethernet0/1
                      [120/2] via 10.30.30.1, 00:00:09, Ethernet0/0
R        10.33.1.0/24 [120/1] via 10.30.31.2, 00:00:18, Ethernet0/1
R        10.33.2.0/24 [120/1] via 10.30.31.2, 00:00:18, Ethernet0/1
R        10.33.3.0/24 [120/1] via 10.30.31.2, 00:00:18, Ethernet0/1
      20.0.0.0/24 is subnetted, 1 subnets
R        20.30.1.0 [120/1] via 10.30.30.1, 00:00:09, Ethernet0/0
      22.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
R        22.33.1.0/24 [120/1] via 10.30.31.2, 00:00:18, Ethernet0/1
```

Now we add some different loop and try some advanced prefix list test.

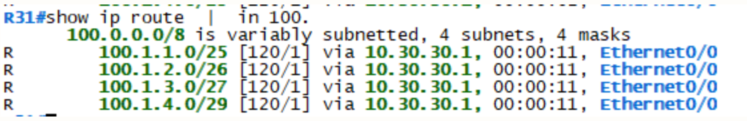
Going to add below network to R30 and advertise them in RIP and check routing populated and block the network greater than to /26 network

Loopback 10 with IP 100.1.1.1/25
Loopback 11 with IP 100.1.2.1/26
Loopback 12 with IP 100.1.3.1/27
Loopback 13 with IP 100.1.4.1/28



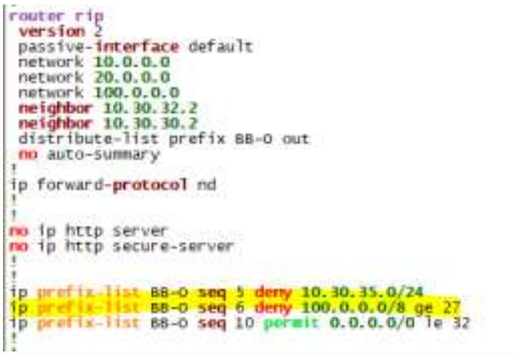
```
Loopback10                 100.1.1.1       YES manual up                    up
Loopback11                 100.1.2.1       YES manual up                    up
Loopback12                 100.1.3.1       YES manual up                    up
Loopback13                 100.1.4.1       YES manual up                    up
```

Check R1 Routes are received.



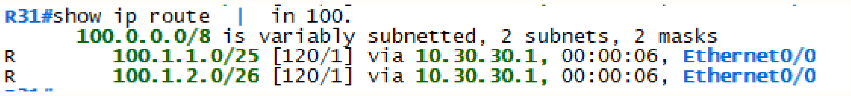
```
R31#show ip route  |  in 100.
      100.0.0.0/8 is variably subnetted, 4 subnets, 4 masks
R        100.1.1.0/25 [120/1] via 10.30.30.1, 00:00:11, Ethernet0/0
R        100.1.2.0/26 [120/1] via 10.30.30.1, 00:00:11, Ethernet0/0
R        100.1.3.0/27 [120/1] via 10.30.30.1, 00:00:11, Ethernet0/0
R        100.1.4.0/29 [120/1] via 10.30.30.1, 00:00:11, Ethernet0/0
```

Let’s make prefix list to block as mentioned above. I have added the deny statement to exiting prefix list as below



```
router rip
 version 2
 passive-interface default
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
 neighbor 10.30.32.2
 neighbor 10.30.30.2
 distribute-list prefix BB-O out
 no auto-summary
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
ip prefix-list BB-O seq 5 deny 10.30.35.0/24
ip prefix-list BB-O seq 6 deny 100.0.0.0/8 ge 27
ip prefix-list BB-O seq 10 permit 0.0.0.0/0 le 32
!
```

Now verify on R31 (we do not see greater than /26 routes in the routing table)

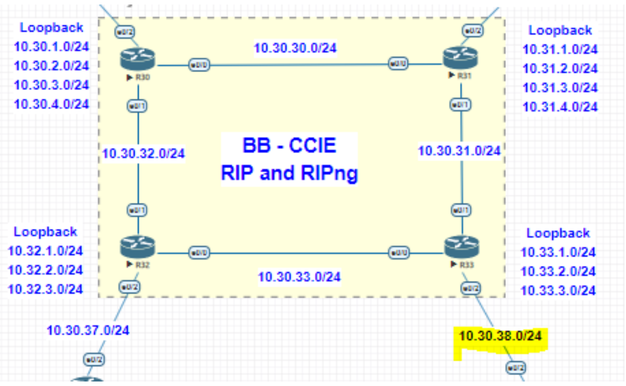


```
R31#show ip route | in 100.
      100.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
R        100.1.1.0/25 [120/1] via 10.30.30.1, 00:00:06, Ethernet0/0
R        100.1.2.0/26 [120/1] via 10.30.30.1, 00:00:06, Ethernet0/0
```

Offset-list:

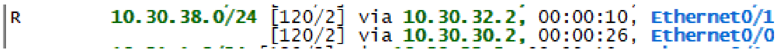
An offset-list is a filtering tool when used as an extrema ratio. By default, an offset-list is a tool used to INCREASE the metric of a route. Of course, if we increase the metric so it reaches 16 hops or more the route will become inaccessible and then discarded/filtered. Let's see the syntax:

offset-list keyword can invoke Standard-ACL (numbered or named)

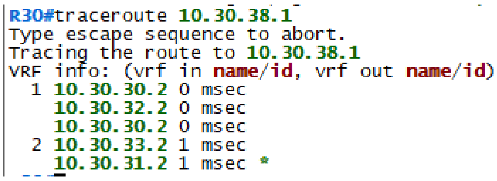


Now we can look, Fro R30 like to reach 10.30.38.1 it has 2 routes, one from R31 and another one from R32

Let us observe what routing table now R30 has and traceroute path to verify.



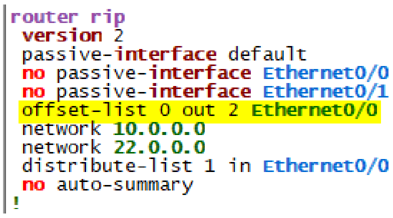
```
R        10.30.38.0/24 [120/2] via 10.30.32.2, 00:00:10, Ethernet0/1
                       [120/2] via 10.30.30.2, 00:00:26, Ethernet0/0
```



```
R30#traceroute 10.30.38.1
Type escape sequence to abort.
Tracing the route to 10.30.38.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.30.2 0 msec
    10.30.32.2 0 msec
    10.30.30.2 0 msec
  2 10.30.33.2 1 msec
    10.30.31.2 1 msec *
```

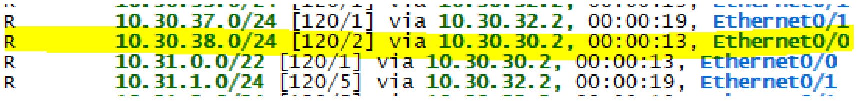
So now to reach 10.30.38.1 it using R32 as preferred path compare to R31, we will use off-list config to make R31 is preferred path compare to R32

On R33 I have added Off-list in the RIP config as below :



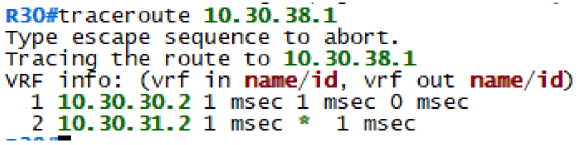
```
router rip
 version 2
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 offset-list 0 out 2 Ethernet0/0
 network 10.0.0.0
 network 22.0.0.0
 distribute-list 1 in Ethernet0/0
 no auto-summary
!
```

Now we observe the routing table R30 again and test the Traceroute.
R30 got only 1 Route in the routing table :



```
R        10.30.37.0/24 [120/1] via 10.30.32.2, 00:00:19, Ethernet0/1
R        10.30.38.0/24 [120/2] via 10.30.30.2, 00:00:13, Ethernet0/0
R        10.31.0.0/22 [120/1] via 10.30.30.2, 00:00:13, Ethernet0/0
R        10.31.1.0/24 [120/5] via 10.30.32.2, 00:00:19, Ethernet0/1
```

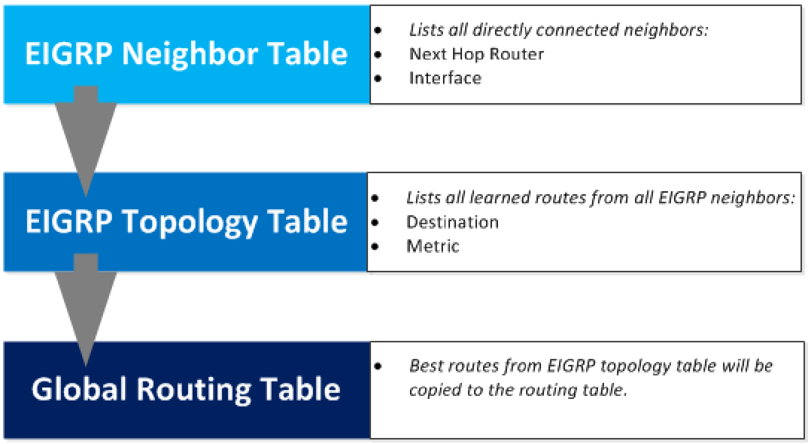
Traceroute will use Via R31 as preferred path now.



```
R30#traceroute 10.30.38.1
Type escape sequence to abort.
Tracing the route to 10.30.38.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.30.2 1 msec 1 msec 0 msec
  2 10.30.31.2 1 msec *  1 msec
```

2.5 EIGRP [for IPv4 and IPv6]

EIGRP (Enhanced Interior Gateway Routing Protocol) which is Cisco's routing protocol
EIGRP stands for Enhanced Interior Gateway Routing Protocol and is a routing protocol created by Cisco. Originally, it was only available on Cisco hardware but since a few years, it's now an open standard. EIGRP is called a hybrid or advanced distance vector protocol and most of the rules that apply to RIP also apply here:
Split Horizon
Route Poisoning
Poison Reverse

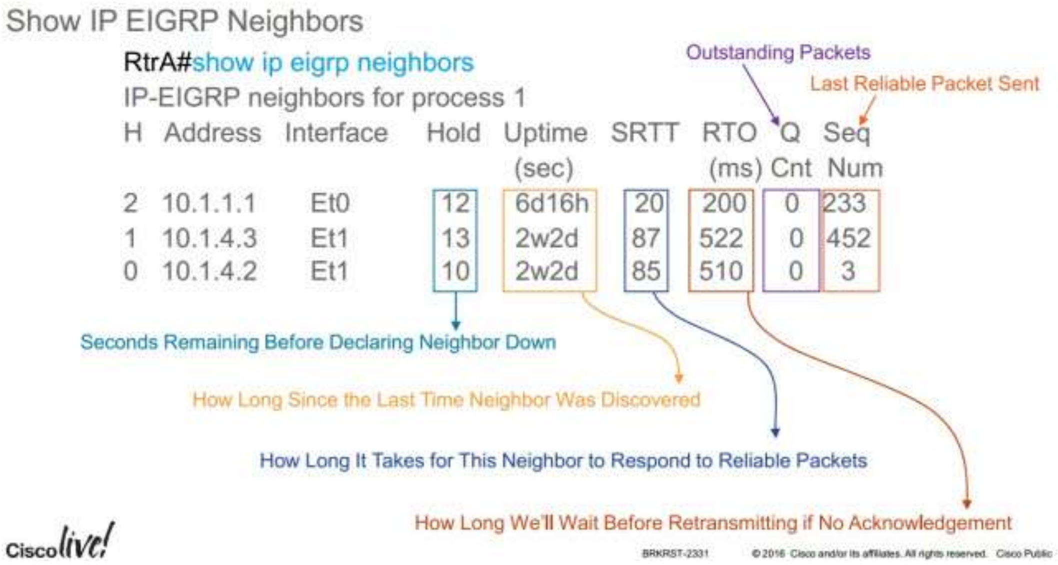


EIGRP Neighbor Table
- *Lists all directly connected neighbors:*
- Next Hop Router
- Interface

EIGRP Topology Table
- *Lists all learned routes from all EIGRP neighbors:*
- Destination
- Metric

Global Routing Table
- *Best routes from EIGRP topology table will be copied to the routing table.*

EIGRP routers will start sending hello packets to other routers just like OSPF does, if you send hello packets and you receive them you will become neighbors. EIGRP neighbors will exchange routing information which will be saved in the topology table. The best path from the topology table will be copied in the routing table.



Show IP EIGRP Neighbors

```
RtrA#show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H  Address   Interface   Hold  Uptime  SRTT  RTO  Q   Seq
                               (sec)         (ms) Cnt Num
2  10.1.1.1    Et0        12   6d16h   20   200   0   233
1  10.1.4.3    Et1        13   2w2d    87   522   0   452
0  10.1.4.2    Et1        10   2w2d    85   510   0   3
```

Outstanding Packets

Last Reliable Packet Sent

Seconds Remaining Before Declaring Neighbor Down

How Long Since the Last Time Neighbor Was Discovered

How Long It Takes for This Neighbor to Respond to Reliable Packets

How Long We'll Wait Before Retransmitting if No Acknowledgement

Cisco live!
BRKRST-2331
©2016 Cisco and/or its affiliates. All rights reserved. Cisco Public

EIGRP updates contain five metrics: **minimum bandwidth, delay, load, reliability, and maximum transmission unit (MTU).** Of these five metrics, by default, only minimum bandwidth and delay are used to compute best path. Unlike most metrics, minimum bandwidth is set to the minimum bandwidth of the entire path, and it does not reflect how many hops or low bandwidth links are in the path. Delay is a cumulative value which increases by the delay value of each segment in the path.

2.5.a Describe packet types

2.5.a [i] Packet types [hello, query, update, and such]
2.5.a [ii] Route types [internal, external]

Hello Identifies neighbors, exchanges parameters, and is sent periodically as a keepalive function
Update Informs neighbors about routing information
Ack Acknowledges Update, Query, and Response packets
Query Asks neighboring routers to verify their route to a particular subnet
Reply Sent by neighbors to reply to a Query
Goodbye Used by a router to notify its neighbors when the router is gracefully shutting down

EIGRP sends both unreliable and reliable packets:
– HELLOS and ACKS are **unreliable**
– UPDATES, QUERIES, REPLIES, SIA-queries and SIA-replies are **reliable**
Reliable packets are sequenced and require an acknowledgement.

2.5.b Implement and troubleshoot neighbor relationship

2.5.b [i] Multicast, unicast EIGRP peering.

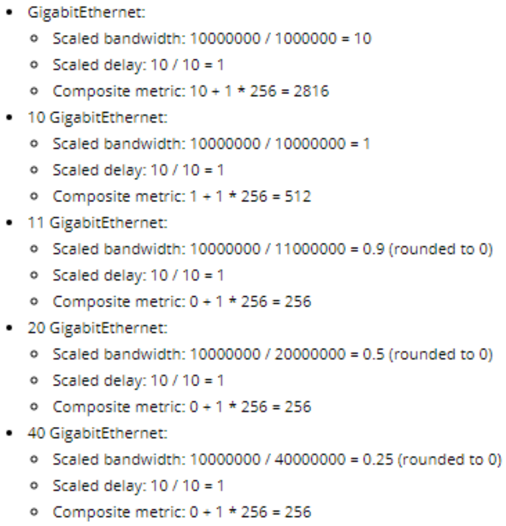
Updates are sent as multicasts but resends are unicast to neighbors who didn’t ACK the update before the RTO timer expired. 16 resends using unicast will be used before declaring a neighbor dead. The multicast flow timer is used for knowing when to switch to unicast packets instead of multicast for a neighbor.

2.5.c Implement and Troubleshoot Loop free path selection

2.5.c [i] RD, FD, FC, successor, feasible successor

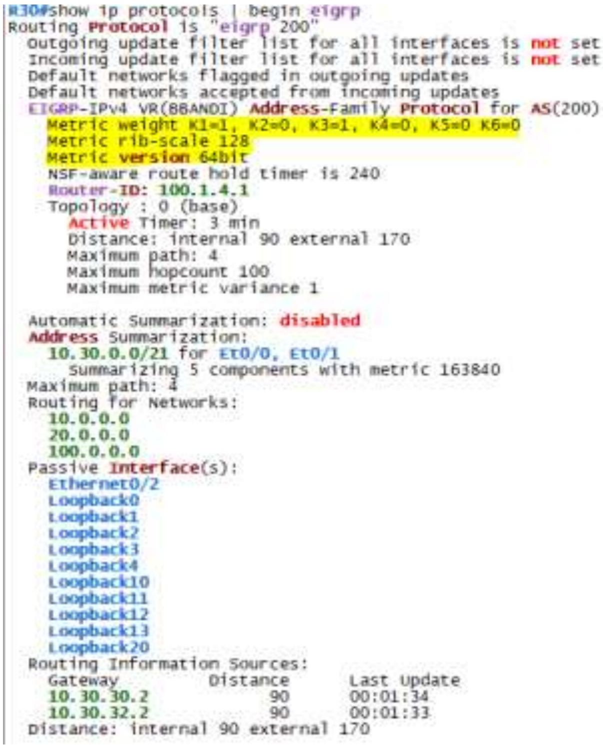
2.5.c [ii] Classic metric
2.5.c [iii] Wide metric

EIGRP uses the bandwidth, delay, reliability, load and K values to calculate a composite cost metric. The problem with this metric is that it doesn't scale for high bandwidth interfaces, It only Support EIGRP Name Mode and not have option to disable or enable, it will detect automatically based on the peer capabilities.



- GigabitEthernet:
  - Scaled bandwidth: 10000000 / 1000000 = 10
  - Scaled delay: 10 / 10 = 1
  - Composite metric: 10 + 1 * 256 = 2816
- 10 GigabitEthernet:
  - Scaled bandwidth: 10000000 / 10000000 = 1
  - Scaled delay: 10 / 10 = 1
  - Composite metric: 1 + 1 * 256 = 512
- 11 GigabitEthernet:
  - Scaled bandwidth: 10000000 / 11000000 = 0.9 (rounded to 0)
  - Scaled delay: 10 / 10 = 1
  - Composite metric: 0 + 1 * 256 = 256
- 20 GigabitEthernet:
  - Scaled bandwidth: 10000000 / 20000000 = 0.5 (rounded to 0)
  - Scaled delay: 10 / 10 = 1
  - Composite metric: 0 + 1 * 256 = 256
- 40 GigabitEthernet:
  - Scaled bandwidth: 10000000 / 40000000 = 0.25 (rounded to 0)
  - Scaled delay: 10 / 10 = 1
  - Composite metric: 0 + 1 * 256 = 256

Here is example :



```
R30#show ip protocols | begin eigrp
Routing Protocol is "eigrp 200"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP-IPv4 VR(BBANDI) Address-Family Protocol for AS(200)
    Metric weight K1=1, K2=0, K3=1, K4=0, K5=0 K6=0
    Metric rib-scale 128
    Metric version 64bit
    NSF-aware route hold timer is 240
    Router-ID: 100.1.4.1
    Topology : 0 (base)
      Active Timer: 3 min
      Distance: internal 90 external 170
      Maximum path: 4
      Maximum hopcount 100
      Maximum metric variance 1

  Automatic Summarization: disabled
  Address Summarization:
    10.30.0.0/21 for Et0/0, Et0/1
      Summarizing 5 components with metric 163840
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
    20.0.0.0
    100.0.0.0
  Passive Interface(s):
    Ethernet0/2
    Loopback0
    Loopback1
    Loopback2
    Loopback3
    Loopback4
    Loopback10
    Loopback11
    Loopback12
    Loopback13
    Loopback20
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.30.30.2            90      00:01:34
    10.30.32.2            90      00:01:33
  Distance: internal 90 external 170
```

Input events and local computation

When an input event occurs EIGRP needs to react, this could be an interface failing, a neighbor failing or an update for a new prefix. When the input event has occured EIGRP performs a local computation, EIGRP looks for a Feasible Successor (FS) route in its topology table and if it cannot find one it will actively query its neighbors for a route.

EIGRP algorithm

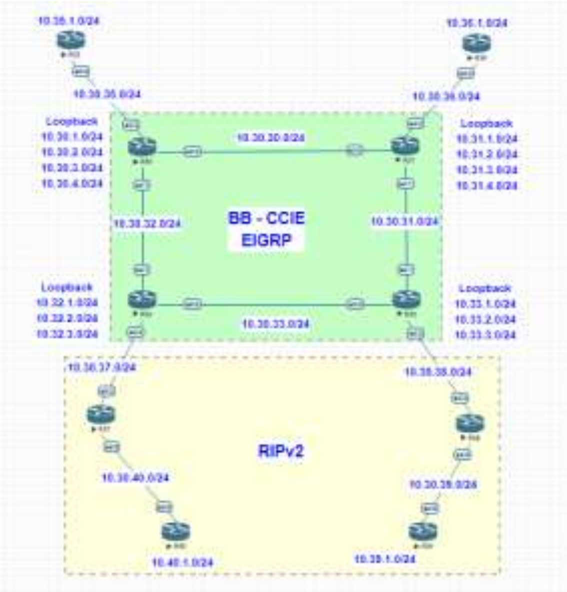
Uses the Diffusing Update ALgorithm (DUAL). Functioning routes are in a passive mode. Routes that no longer have a successor is in active mode since the route has to query its neighbors for a FS. The term Stuck In Active (SIA) means that an route has been active for too long, the active timer has expired. The active timer is set to 180 seconds by default, the active timer can also be disabled if needed.

Load balancing

EIGRP allows for up to 16 equal-metric routes to be installed in the routing table, the default is four. EIGRP also has something called variance. Variance allows for non equal-metric load balancing. The route still has to meet the feasibility condition to be considered for load balancing. The variance command is a multiplier, if the FD is 10000 for the current succcessor and there is a FS with a RD of 5000 and FD of 200000, variance 2 would make the router load balance between these two routes, variance 2 means the FD of the second best route can be twice as high as the best.

The load balancing can be done in a few different ways, traffic-share balanced means that the traffic will be distributed according to the metric, routes with lower metrics will see more traffic on them. Traffic-share min, install multiple routes but send only traffic on the one with the lowest metric. Traffic-share min across-interfaces, if more than one route has the same metric choose different outgoing interfaces for a better load balancing. The no traffic-share command will balance evenly across routes no matter what the metric is.

We use below topology for configuring.



HYBRID IGP

-Properties of both link-state and distance vector
-forms active adjacencies but still uses split-horizon (link state)
-Dual, guaranteed loop free but still “routing by rumor” (split-horizon)

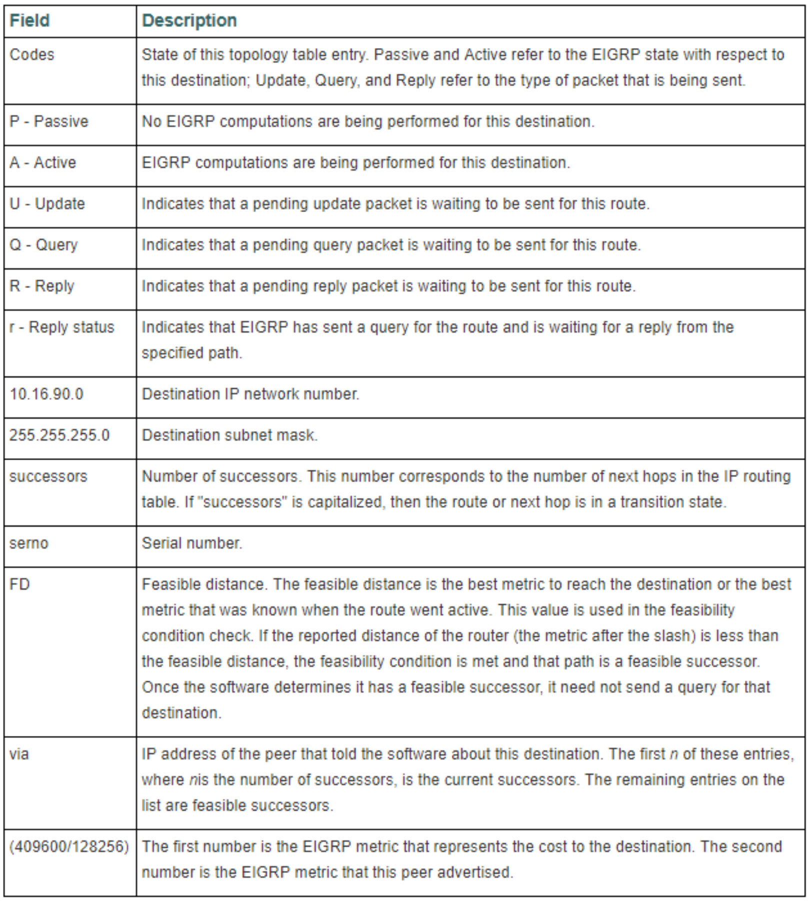
Uses its own transport protocol

-IP Protocol 88
-Multicast to 224.0.0.10 to establish adjacencies
-unicast and multicast to synchronize the topology

2.5.d Implement and troubleshoot operations

2.5.d [i] General operations

2.5.d [ii] Topology table, update, query, active, passive



Field	Description
Codes	State of this topology table entry. Passive and Active refer to the EIGRP state with respect to this destination; Update, Query, and Reply refer to the type of packet that is being sent.
P - Passive	No EIGRP computations are being performed for this destination.
A - Active	EIGRP computations are being performed for this destination.
U - Update	Indicates that a pending update packet is waiting to be sent for this route.
Q - Query	Indicates that a pending query packet is waiting to be sent for this route.
R - Reply	Indicates that a pending reply packet is waiting to be sent for this route.
r - Reply status	Indicates that EIGRP has sent a query for the route and is waiting for a reply from the specified path.
10.16.90.0	Destination IP network number.
255.255.255.0	Destination subnet mask.
successors	Number of successors. This number corresponds to the number of next hops in the IP routing table. If "successors" is capitalized, then the route or next hop is in a transition state.
serno	Serial number.
FD	Feasible distance. The feasible distance is the best metric to reach the destination or the best metric that was known when the route went active. This value is used in the feasibility condition check. If the reported distance of the router (the metric after the slash) is less than the feasible distance, the feasibility condition is met and that path is a feasible successor. Once the software determines it has a feasible successor, it need not send a query for that destination.
via	IP address of the peer that told the software about this destination. The first *n* of these entries, where *n*is the number of successors, is the current successors. The remaining entries on the list are feasible successors.
(409600/128256)	The first number is the EIGRP metric that represents the cost to the destination. The second number is the EIGRP metric that this peer advertised.

2.5.d [iii] Stuck in active

The EIGRP **Stuck In Active** event happens when a router that sends a Query message asking for a route does not receive a Reply from an adjacent in a certain amount of time. This time is called the Active time and is set to 180 seconds by default; at half of the active timer, the router that lost the route start sending SIA-Query in order to validate the availability of the adjacent neighbor, if after 90 seconds the router does not receive a SIA-Reply, then the adjacency is dropped.

2.5.d [iv] Graceful shutdown

With graceful shutdown, a goodbye message is broadcast when an eigrp routing process is shutdown, to inform adjacent peers about the impending topology change. This feature allows supporting EIGRP peers to synchronize and recalculate neighbour relationships more efficiently than would occur if the peers discovered the topology change after the hold time expired.

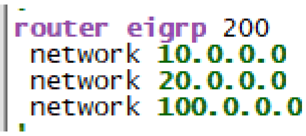
EIGRP sends an interface goodbye message with all K values set to 255 towards neighbours connected to its different interfaces. (these should have been advertised in EIGRP process)

If the IOS neighbour router supports this graceful shutdown, then it says "interface goodbye received". If not, it says K values mismatch.

NOTE: EIGRP router will not send a goodbye message if an interface is shutdown or the router is reloaded.

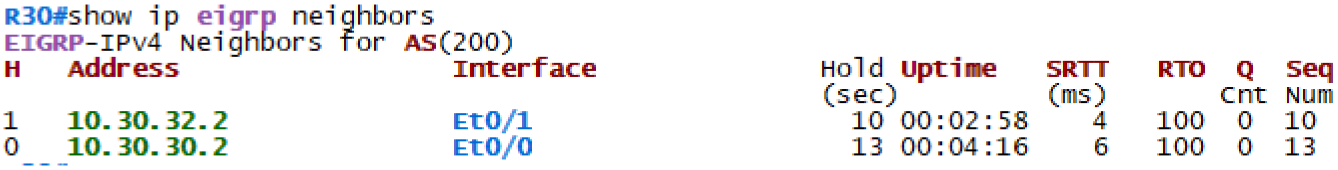
When the command **eigrp log-neighbor-changes** in EIGRP-configuration mode is enabled, routers that had adjacencies with the router sending the graceful shutdown may print out one of two things depending on what Cisco IOS Release is running.

EIGRP Basic configuration to enable Eigrp on R30 / R31 / R32 / R33



```
!
router eigrp 200
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
!
```

Check the neighbour on all routers:

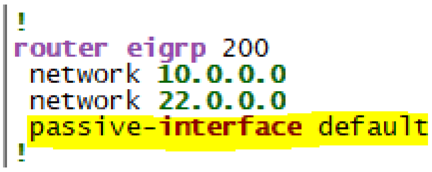


```
R30#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
1   10.30.32.2              Et0/1                    10 00:02:58    4   100  0  10
0   10.30.30.2              Et0/0                    13 00:04:16    6   100  0  13
```

Passive:

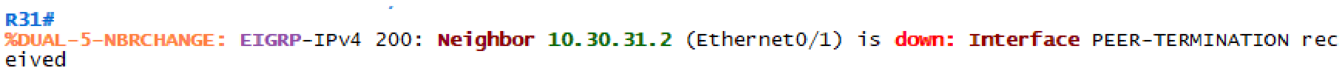
Passive interface command in EIGRP blocks both Multicast and Unicast (unlike RIP)

Let's enable passive interface on all routers by default
And Enable Multicast between R30 towards R31 and R32
Unicast between R33 Towards R31 and R32



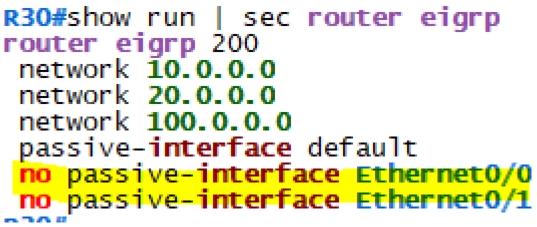
```
!
router eigrp 200
 network 10.0.0.0
 network 22.0.0.0
 passive-interface default
!
```

As soon we disable we can see below message on router:



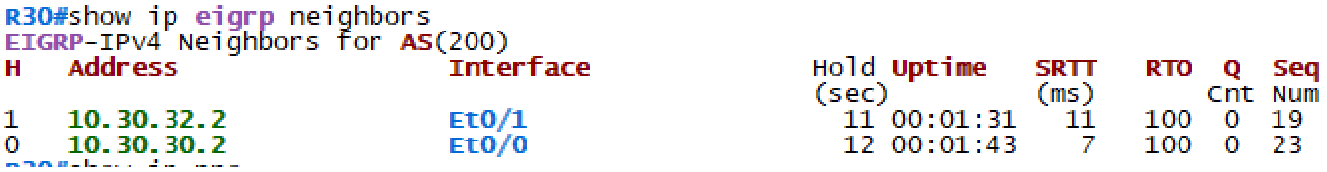
```
R31#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 10.30.31.2 (Ethernet0/1) is down: Interface PEER-TERMINATION rec
eived
```

Now we configure Eigrp Multicast enable on R30 router



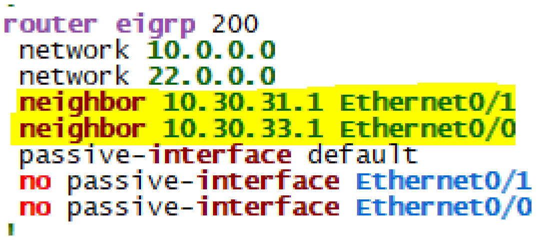
```
R30#show run | sec router eigrp
router eigrp 200
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
```

We can see the neighbours come up.



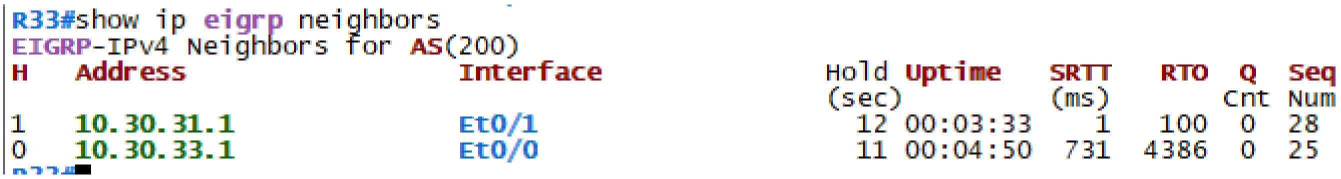
```
R30#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
1   10.30.32.2              Et0/1                    11 00:01:31   11   100  0  19
0   10.30.30.2              Et0/0                    12 00:01:43    7   100  0  23
```

Now we enable Unicast on R33



```
router eigrp 200
 network 10.0.0.0
 network 22.0.0.0
 neighbor 10.30.31.1 Ethernet0/1
 neighbor 10.30.33.1 Ethernet0/0
 passive-interface default
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/0
!
```

Now we can see neighbours.

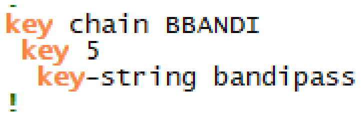


```
R33#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
1   10.30.31.1              Et0/1                    12 00:03:33    1   100  0  28
0   10.30.33.1              Et0/0                    11 00:04:50  731  4386  0  25
```

Authentication:

Only MD5 support.
Support SHA authentication in Named Mode (not on classic Version)

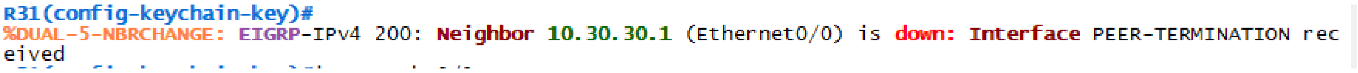
Now we enable authentication between R30 and R31



```
key chain BBANDI
 key 5
  key-string bandipass
!
```

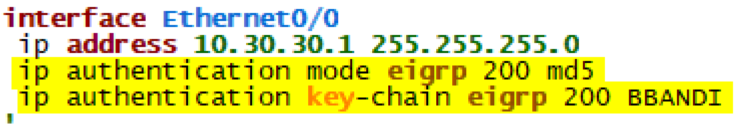
Enable key on interface eth 0/0 on both the side, (if no the neighbourship fail due to authentication fails)

Here is the message since R31 not configured MD5 authentication.



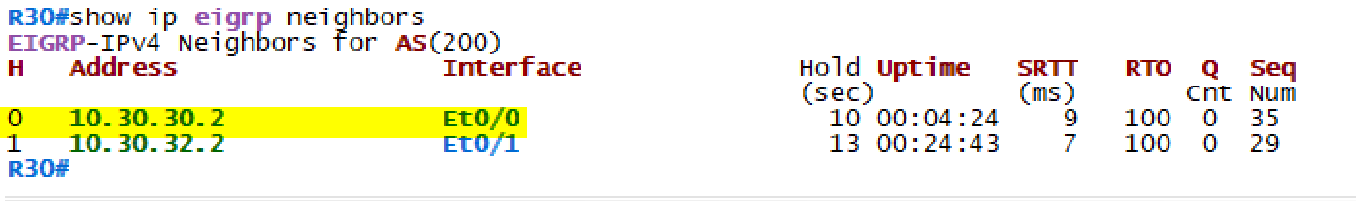
```
R31(config-keychain-key)#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 10.30.30.1 (Ethernet0/0) is down: Interface PEER-TERMINATION rec
eived
```

Once below configuration enabled both R30 and R31



```
interface Ethernet0/0
 ip address 10.30.30.1 255.255.255.0
 ip authentication mode eigrp 200 md5
 ip authentication key-chain eigrp 200 BBANDI
!
```

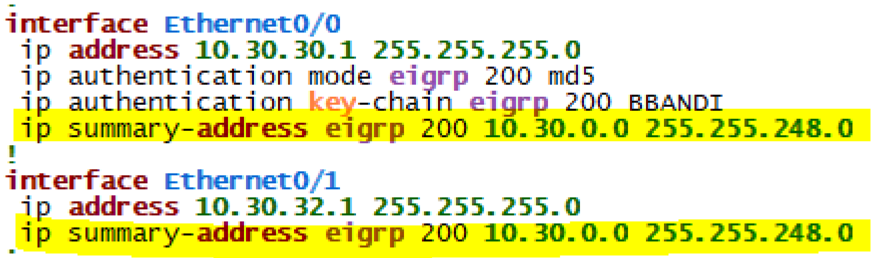
You can verify neighbourship :



```
R30#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
0   10.30.30.2              Et0/0                    10 00:04:24    9   100  0  35
1   10.30.32.2              Et0/1                    13 00:24:43    7   100  0  29
R30#
```

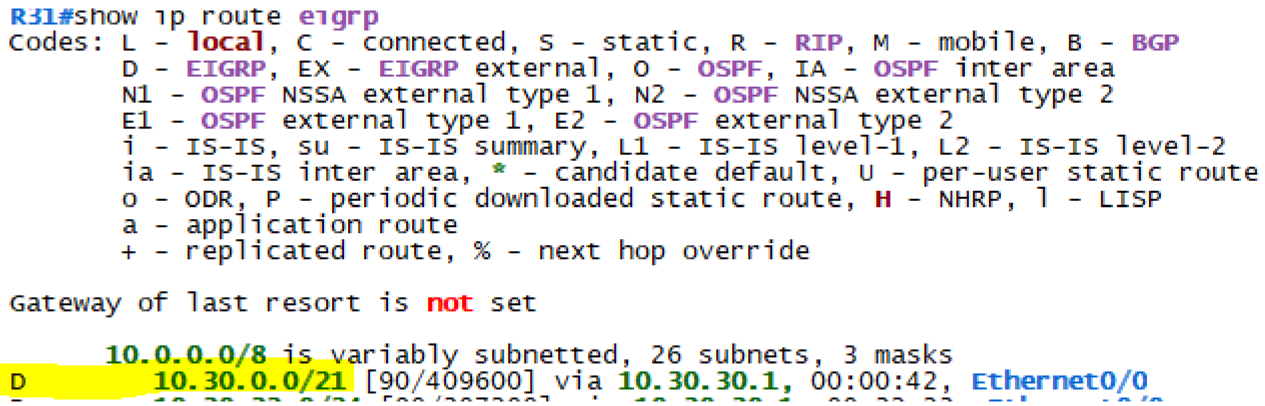
SUMMARISATION:

We will summarise 10.30.0.0/21 network connected in R30 Loopback address Towards R31 and R32

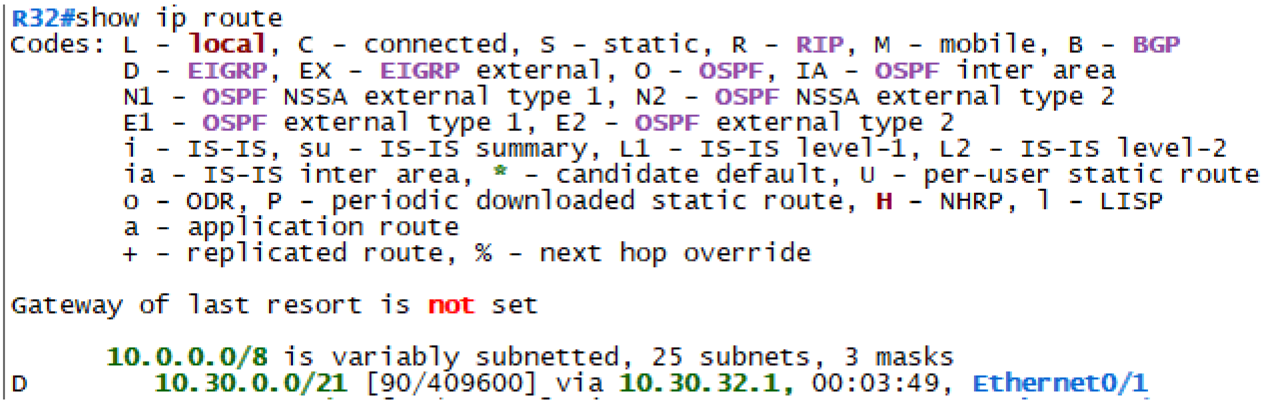


```
!
interface Ethernet0/0
 ip address 10.30.30.1 255.255.255.0
 ip authentication mode eigrp 200 md5
 ip authentication key-chain eigrp 200 BBANDI
 ip summary-address eigrp 200 10.30.0.0 255.255.248.0
!
interface Ethernet0/1
 ip address 10.30.32.1 255.255.255.0
 ip summary-address eigrp 200 10.30.0.0 255.255.248.0
!
```

We will verify the route on R31 and R32



```
R31#show ip route eigrp
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 26 subnets, 3 masks
D        10.30.0.0/21 [90/409600] via 10.30.30.1, 00:00:42, Ethernet0/0
```



```
R32#show ip route
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 25 subnets, 3 masks
D        10.30.0.0/21 [90/409600] via 10.30.32.1, 00:03:49, Ethernet0/1
```

Summary Leak Map :

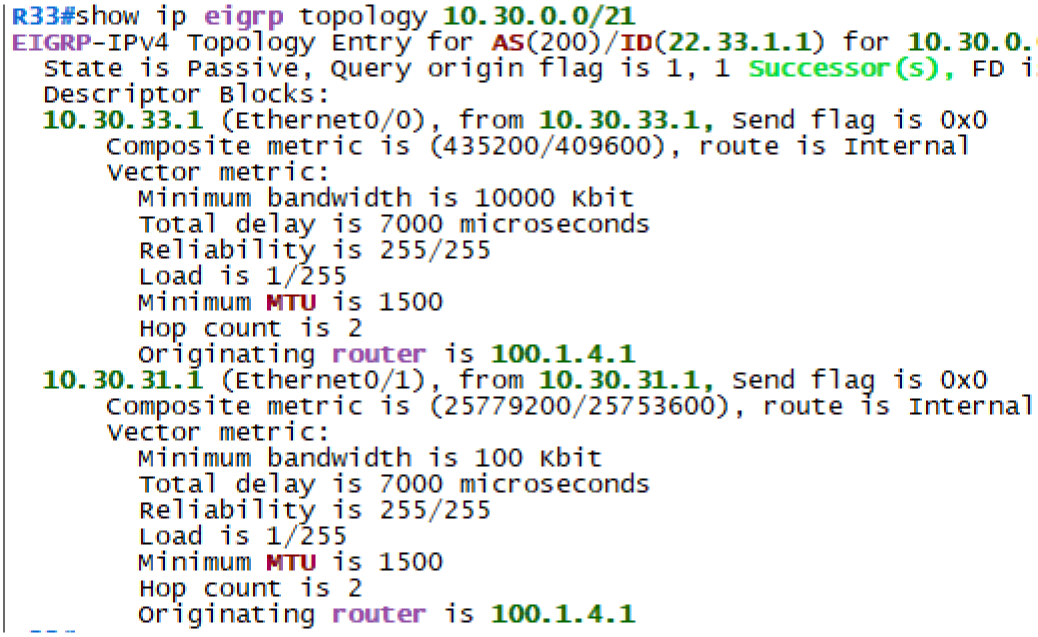
R30 to R31 and R33 have lower speed links
R30 to R32 and R33 have higher speed links

We have summarised so the path take always use higher bandwidth, and Lower bandwidth link will be not utilised at all.

So to optimise use bandwidth, we can send some subnet traffic towards R30 to R31 and R33

So we use Leak Map to leak the route towards R31.

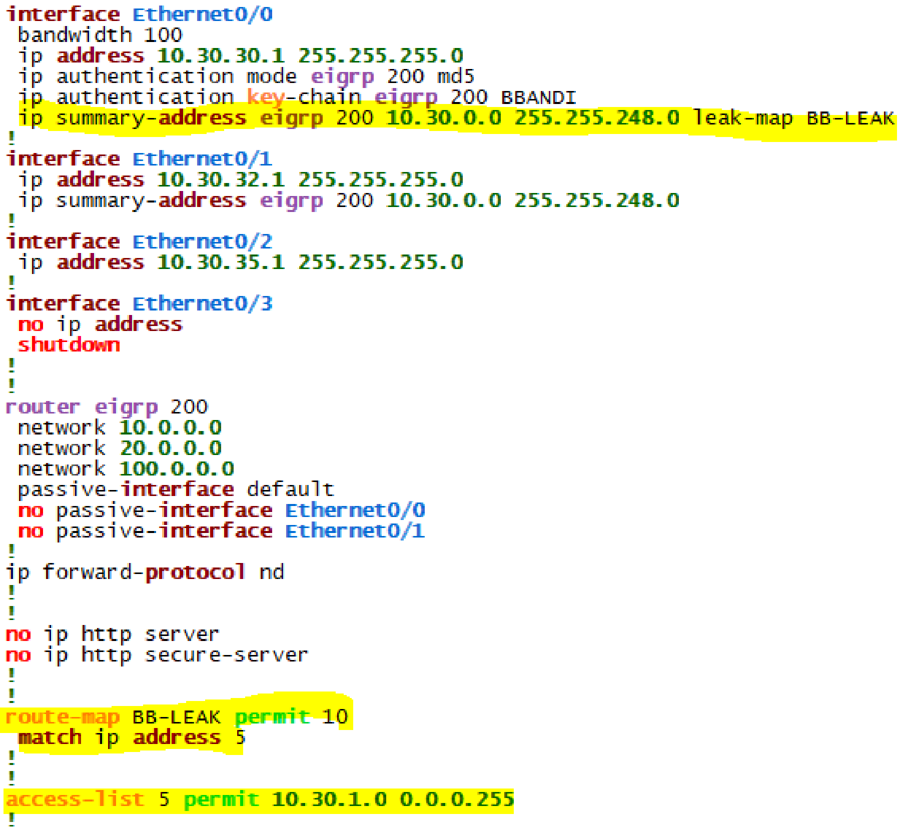
Summarised routes for 10.30.0.0/21



```
R33#show ip eigrp topology 10.30.0.0/21
EIGRP-IPv4 Topology Entry for AS(200)/ID(22.33.1.1) for 10.30.0.0/21
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 435200
  Descriptor Blocks:
  10.30.33.1 (Ethernet0/0), from 10.30.33.1, Send flag is 0x0
      Composite metric is (435200/409600), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 7000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 100.1.4.1
  10.30.31.1 (Ethernet0/1), from 10.30.31.1, Send flag is 0x0
      Composite metric is (25779200/25753600), route is Internal
      Vector metric:
        Minimum bandwidth is 100 Kbit
        Total delay is 7000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 100.1.4.1
```

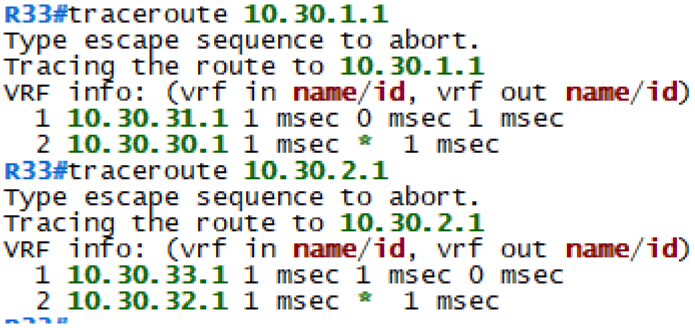
R33 reach 10.30.1.1 use R32 and R30 because of bandwidth.

Now we configure R33 to reach 10.30.1.1 to use R31 and R30 (rest 10.30.2.1 /3.1 /4.1 use high utilisation of Links)

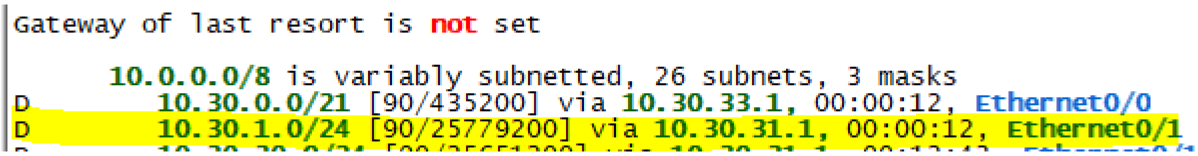


```
interface Ethernet0/0
 bandwidth 100
 ip address 10.30.30.1 255.255.255.0
 ip authentication mode eigrp 200 md5
 ip authentication key-chain eigrp 200 BBANDI
 ip summary-address eigrp 200 10.30.0.0 255.255.248.0 leak-map BB-LEAK
!
interface Ethernet0/1
 ip address 10.30.32.1 255.255.255.0
 ip summary-address eigrp 200 10.30.0.0 255.255.248.0
!
interface Ethernet0/2
 ip address 10.30.35.1 255.255.255.0
!
interface Ethernet0/3
 no ip address
 shutdown
!
!
router eigrp 200
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
route-map BB-LEAK permit 10
 match ip address 5
!
!
access-list 5 permit 10.30.1.0 0.0.0.255
!
```

Now check the traceroute from R33 for 10.30.1.1 and others



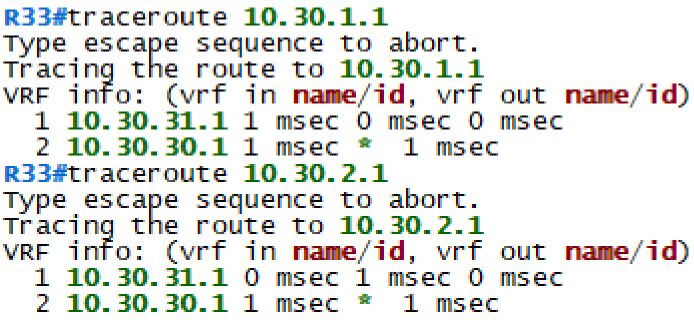
```
R33#traceroute 10.30.1.1
Type escape sequence to abort.
Tracing the route to 10.30.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.31.1 1 msec 0 msec 1 msec
  2 10.30.30.1 1 msec *  1 msec
R33#traceroute 10.30.2.1
Type escape sequence to abort.
Tracing the route to 10.30.2.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.33.1 1 msec 1 msec 0 msec
  2 10.30.32.1 1 msec *  1 msec
```



```
Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 26 subnets, 3 masks
D        10.30.0.0/21 [90/435200] via 10.30.33.1, 00:00:12, Ethernet0/0
D        10.30.1.0/24 [90/25779200] via 10.30.31.1, 00:00:12, Ethernet0/1
```

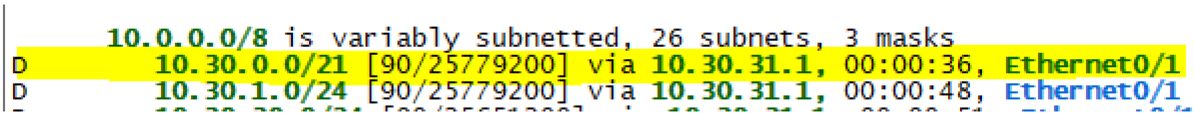
If Link go down netween R30 towards R32, all the routes will be used Lower bandwidth Link

Lets check



```
R33#traceroute 10.30.1.1
Type escape sequence to abort.
Tracing the route to 10.30.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.31.1 1 msec 0 msec 0 msec
  2 10.30.30.1 1 msec *  1 msec
R33#traceroute 10.30.2.1
Type escape sequence to abort.
Tracing the route to 10.30.2.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.31.1 0 msec 1 msec 0 msec
  2 10.30.30.1 1 msec *  1 msec
```

Verify the routes



```
      10.0.0.0/8 is variably subnetted, 26 subnets, 3 masks
D        10.30.0.0/21 [90/25779200] via 10.30.31.1, 00:00:36, Ethernet0/1
D        10.30.1.0/24 [90/25779200] via 10.30.31.1, 00:00:48, Ethernet0/1
```

FILTERING:

Eigrp inbound route filtering

Distribute-list
--standard access list
--extended acess-list
Source is route source, destination is prefix

Offset-List

Distance
-255=infinite
-can be per neighbor

route-map
-metric filter
-route tag filter

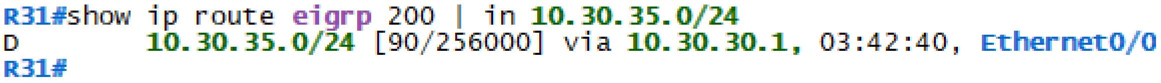
Filtering in EIGRP is similar to RIP, we can use standard or extended ACL to filter was we receive IN an interface or what we send OUT.

The extended ACL will not only match the route but the source of the route.
We can see the source of a route the following way.

Example : standard access list

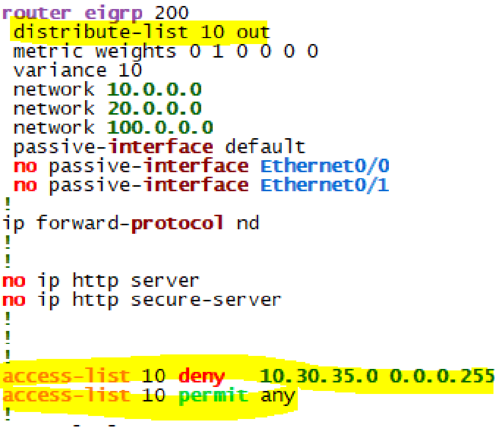
R30 having internally routes, which not required to know other routers, as this is internal network only, so let we block this network using standard access-list for 10.30.35.0/24

Before we apply lets verify on R31 we can see this routes



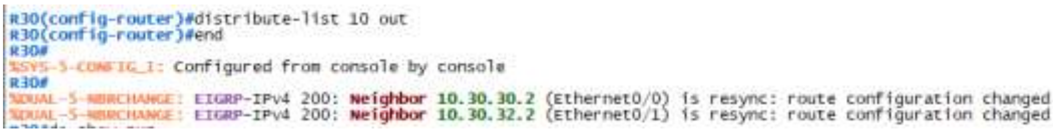
```
R31#show ip route eigrp 200 | in 10.30.35.0/24
D        10.30.35.0/24 [90/256000] via 10.30.30.1, 03:42:40, Ethernet0/0
R31#
```

Now we create an access-list for this network.



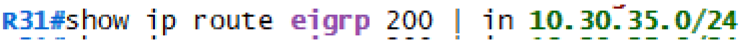
```
router eigrp 200
 distribute-list 10 out
 metric weights 0 1 0 0 0 0
 variance 10
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
!
access-list 10 deny   10.30.35.0 0.0.0.255
access-list 10 permit any
!
```

I have not mentioned any interface that means all the neighbour and interface will no get this route. As soon we apply this config, we can see eigrp re-convergence (very fast in the routing protocols)



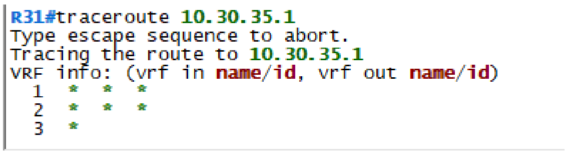
```
R30(config-router)#distribute-list 10 out
R30(config-router)#end
R30#
%SYS-5-CONFIG_I: Configured from console by console
R30#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 10.30.30.2 (Ethernet0/0) is resync: route configuration changed
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 10.30.32.2 (Ethernet0/1) is resync: route configuration changed
```

Now we verify the same routes in R31 and R32



```
R31#show ip route eigrp 200 | in 10.30.35.0/24
```

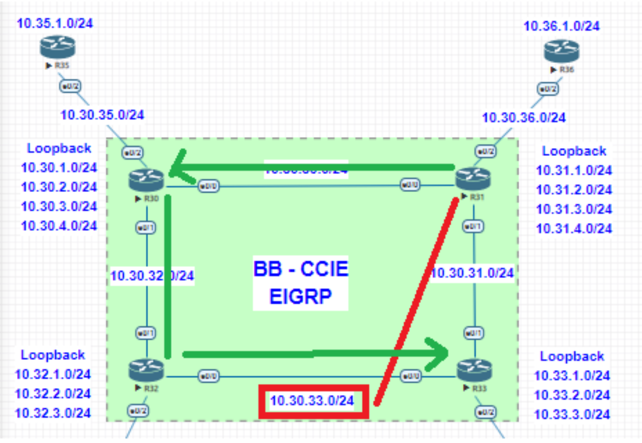
No routes found



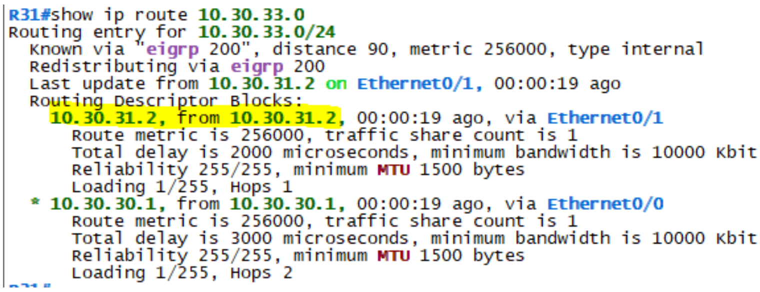
```
R31#traceroute 10.30.35.1
Type escape sequence to abort.
Tracing the route to 10.30.35.1
VRF info: (vrf in name/id, vrf out name/id)
  1  *  *  *
  2  *  *  *
  3  *
```

EXAMPLE: extended acess-list

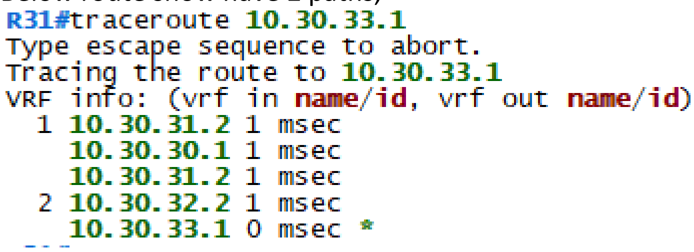
Now illustrate below example: any way we have blocked standard access-list for 10.30.35.0/24 from R30 Using extend ACL we going to block 10.30.33.0/24 advertising from R33 interface connected to R31



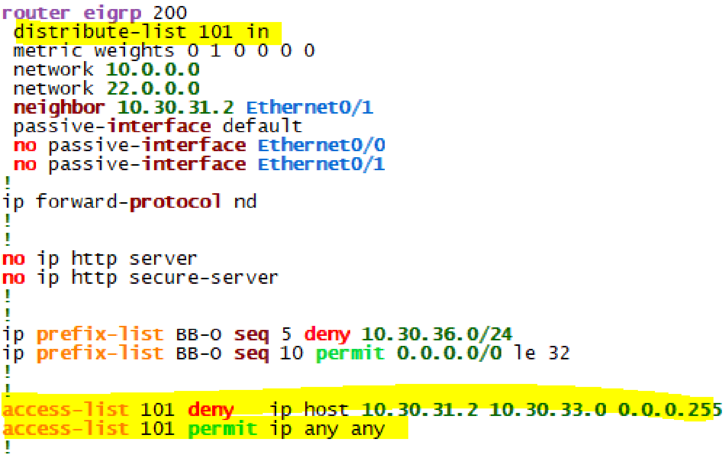
This subnet have 2 routes as below :


```
R31#show ip route 10.30.33.0
Routing entry for 10.30.33.0/24
  Known via "eigrp 200", distance 90, metric 256000, type internal
  Redistributing via eigrp 200
  Last update from 10.30.31.2 on Ethernet0/1, 00:00:19 ago
  Routing Descriptor Blocks:
    10.30.31.2, from 10.30.31.2, 00:00:19 ago, via Ethernet0/1
        Route metric is 256000, traffic share count is 1
        Total delay is 2000 microseconds, minimum bandwidth is 10000 Kbit
        Reliability 255/255, minimum MTU 1500 bytes
        Loading 1/255, Hops 1
  * 10.30.30.1, from 10.30.30.1, 00:00:19 ago, via Ethernet0/0
        Route metric is 256000, traffic share count is 1
        Total delay is 3000 microseconds, minimum bandwidth is 10000 Kbit
        Reliability 255/255, minimum MTU 1500 bytes
        Loading 1/255, Hops 2
```

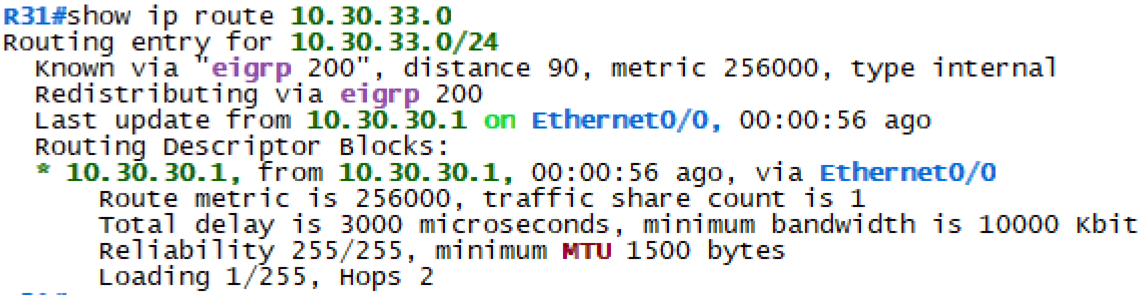
Below route show have 2 paths,


```
R31#traceroute 10.30.33.1
Type escape sequence to abort.
Tracing the route to 10.30.33.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.31.2 1 msec
    10.30.30.1 1 msec
    10.30.31.2 1 msec
  2 10.30.32.2 1 msec
    10.30.33.1 0 msec *
```

Now we configure extend access-list as below and apply to EIGRP process.

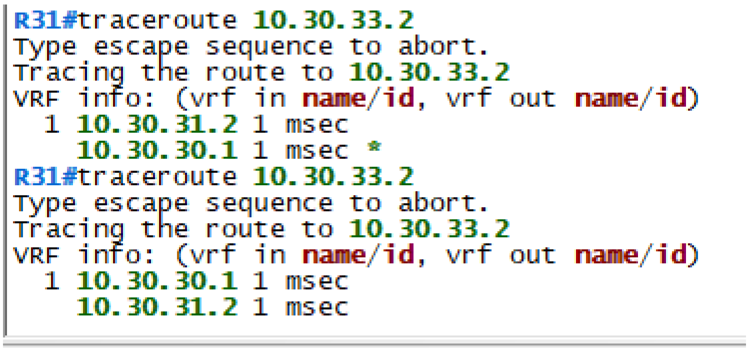

```
router eigrp 200
 distribute-list 101 in
 metric weights 0 1 0 0 0 0
 network 10.0.0.0
 network 22.0.0.0
 neighbor 10.30.31.2 Ethernet0/1
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
ip prefix-list BB-O seq 5 deny 10.30.36.0/24
ip prefix-list BB-O seq 10 permit 0.0.0.0/0 le 32
!
!
access-list 101 deny   ip host 10.30.31.2 10.30.33.0 0.0.0.255
access-list 101 permit ip any any
!
```

Now we will verify routes and path by using trace route



```
R31#show ip route 10.30.33.0
Routing entry for 10.30.33.0/24
  Known via "eigrp 200", distance 90, metric 256000, type internal
  Redistributing via eigrp 200
  Last update from 10.30.30.1 on Ethernet0/0, 00:00:56 ago
  Routing Descriptor Blocks:
  * 10.30.30.1, from 10.30.30.1, 00:00:56 ago, via Ethernet0/0
      Route metric is 256000, traffic share count is 1
      Total delay is 3000 microseconds, minimum bandwidth is 10000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

Below traceroute always use R30 to reach 10.30.33.2

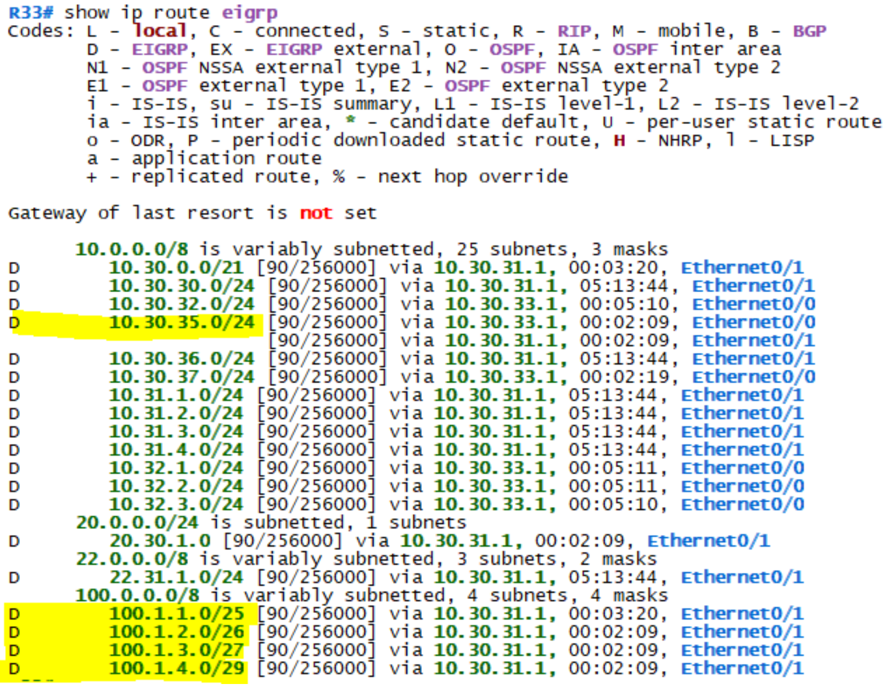


```
R31#traceroute 10.30.33.2
Type escape sequence to abort.
Tracing the route to 10.30.33.2
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.31.2 1 msec
    10.30.30.1 1 msec *
R31#traceroute 10.30.33.2
Type escape sequence to abort.
Tracing the route to 10.30.33.2
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.30.1 1 msec
    10.30.31.2 1 msec
```

Example :
route-map
-metric filter
-route tag filter

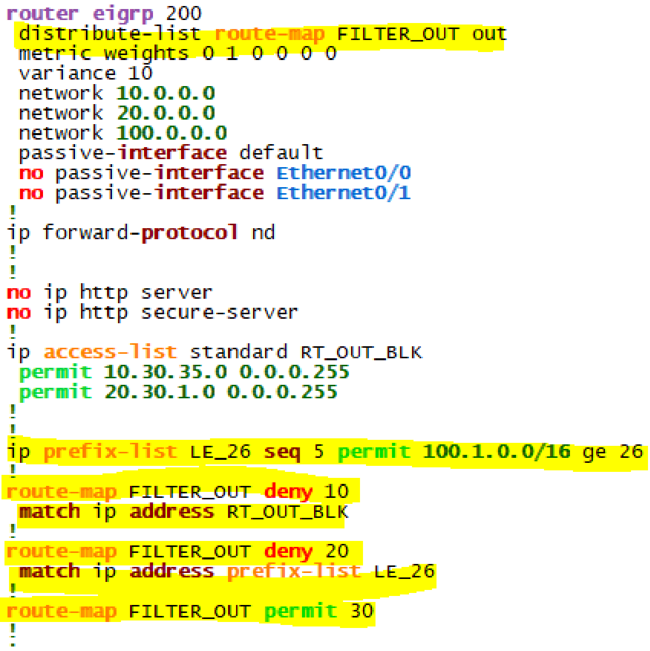
From R30 we going to block outbound 10.30.35.0/24 and 100.x.x.x network Greater than Equal to 26

Verify before we apply the rule in R33.



```
R33# show ip route eigrp
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 25 subnets, 3 masks
D        10.30.0.0/21 [90/256000] via 10.30.31.1, 00:03:20, Ethernet0/1
D        10.30.30.0/24 [90/256000] via 10.30.31.1, 05:13:44, Ethernet0/1
D        10.30.32.0/24 [90/256000] via 10.30.33.1, 00:05:10, Ethernet0/0
D        10.30.35.0/24 [90/256000] via 10.30.33.1, 00:02:09, Ethernet0/0
                       [90/256000] via 10.30.31.1, 00:02:09, Ethernet0/1
D        10.30.36.0/24 [90/256000] via 10.30.31.1, 05:13:44, Ethernet0/1
D        10.30.37.0/24 [90/256000] via 10.30.33.1, 00:02:19, Ethernet0/0
D        10.31.1.0/24 [90/256000] via 10.30.31.1, 05:13:44, Ethernet0/1
D        10.31.2.0/24 [90/256000] via 10.30.31.1, 05:13:44, Ethernet0/1
D        10.31.3.0/24 [90/256000] via 10.30.31.1, 05:13:44, Ethernet0/1
D        10.31.4.0/24 [90/256000] via 10.30.31.1, 05:13:44, Ethernet0/1
D        10.32.1.0/24 [90/256000] via 10.30.33.1, 00:05:11, Ethernet0/0
D        10.32.2.0/24 [90/256000] via 10.30.33.1, 00:05:11, Ethernet0/0
D        10.32.3.0/24 [90/256000] via 10.30.33.1, 00:05:10, Ethernet0/0
      20.0.0.0/24 is subnetted, 1 subnets
D        20.30.1.0 [90/256000] via 10.30.31.1, 00:02:09, Ethernet0/1
      22.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
D        22.31.1.0/24 [90/256000] via 10.30.31.1, 05:13:44, Ethernet0/1
      100.0.0.0/8 is variably subnetted, 4 subnets, 4 masks
D        100.1.1.0/25 [90/256000] via 10.30.31.1, 00:03:20, Ethernet0/1
D        100.1.2.0/26 [90/256000] via 10.30.31.1, 00:02:09, Ethernet0/1
D        100.1.3.0/27 [90/256000] via 10.30.31.1, 00:02:09, Ethernet0/1
D        100.1.4.0/29 [90/256000] via 10.30.31.1, 00:02:09, Ethernet0/1
```

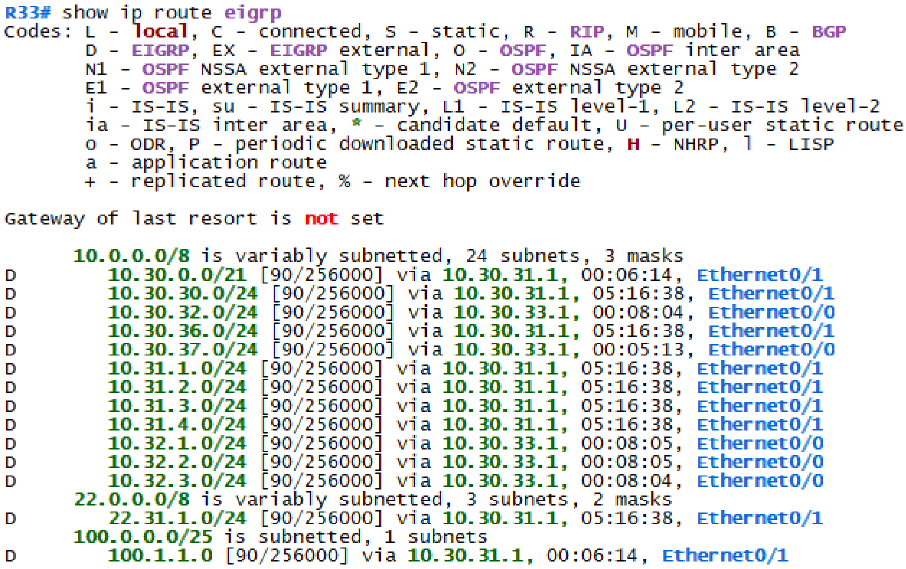
Now we configure as below in R30 with route-map and prefix-list.



```
router eigrp 200
 distribute-list route-map FILTER_OUT out
 metric weights 0 1 0 0 0 0
 variance 10
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
ip access-list standard RT_OUT_BLK
 permit 10.30.35.0 0.0.0.255
 permit 20.30.1.0 0.0.0.255
!
!
ip prefix-list LE_26 seq 5 permit 100.1.0.0/16 ge 26
!
route-map FILTER_OUT deny 10
 match ip address RT_OUT_BLK
!
route-map FILTER_OUT deny 20
 match ip address prefix-list LE_26
!
route-map FILTER_OUT permit 30
!
```

Now we verify the same on R33

We do not see the route entries what we have blocked.



```
R33# show ip route eigrp
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 24 subnets, 3 masks
D        10.30.0.0/21 [90/256000] via 10.30.31.1, 00:06:14, Ethernet0/1
D        10.30.30.0/24 [90/256000] via 10.30.31.1, 05:16:38, Ethernet0/1
D        10.30.32.0/24 [90/256000] via 10.30.33.1, 00:08:04, Ethernet0/0
D        10.30.36.0/24 [90/256000] via 10.30.31.1, 05:16:38, Ethernet0/1
D        10.30.37.0/24 [90/256000] via 10.30.33.1, 00:05:13, Ethernet0/0
D        10.31.1.0/24 [90/256000] via 10.30.31.1, 05:16:38, Ethernet0/1
D        10.31.2.0/24 [90/256000] via 10.30.31.1, 05:16:38, Ethernet0/1
D        10.31.3.0/24 [90/256000] via 10.30.31.1, 05:16:38, Ethernet0/1
D        10.31.4.0/24 [90/256000] via 10.30.31.1, 05:16:38, Ethernet0/1
D        10.32.1.0/24 [90/256000] via 10.30.33.1, 00:08:05, Ethernet0/0
D        10.32.2.0/24 [90/256000] via 10.30.33.1, 00:08:05, Ethernet0/0
D        10.32.3.0/24 [90/256000] via 10.30.33.1, 00:08:04, Ethernet0/0
      22.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
D        22.31.1.0/24 [90/256000] via 10.30.31.1, 05:16:38, Ethernet0/1
      100.0.0.0/25 is subnetted, 1 subnets
D        100.1.1.0 [90/256000] via 10.30.31.1, 00:06:14, Ethernet0/1
```

EIGRP Offset Lists:

In certain situations, we may need to manipulate EIGRP metrics, and one method is to use Offset Lists to increase both the AD (Advertised Distance) and FD (Feasible Distance) of a route by a certain value – the offset.

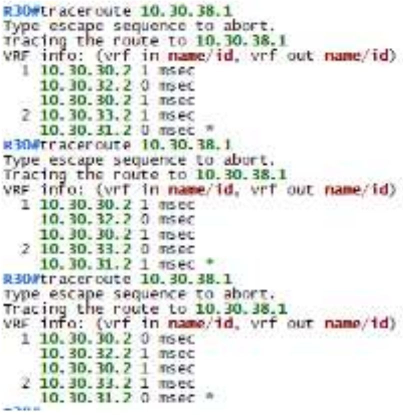
Distribute list functions to control the Routes which are advertised or received while **Offset List function is to modify the advertised/received metric of Routes**.

Offset List configurations define the following:

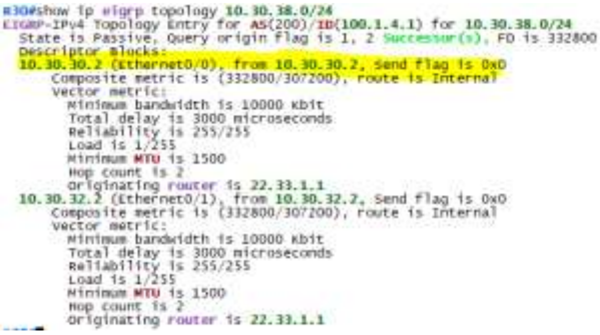
- route(s) that we want to amend the metric for (matching an ACL)
- direction of the updates being sent or received
- interface on which updates are sent or received
- offset integer value

From R30 we have 2 routes for 10.30.38.0/24 network as below, it use R31 and R33 to each always.
So we make use of R32 as primary path of that link do down we use R31 as secondary path with offlist config

Verify before we modify :

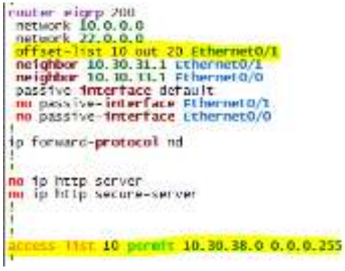


```
R30#traceroute 10.30.38.1
Type escape sequence to abort.
Tracing the route to 10.30.38.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.30.2 1 msec
    10.30.32.2 0 msec
    10.30.30.2 1 msec
  2 10.30.33.2 1 msec
    10.30.31.2 0 msec *
R30#traceroute 10.30.38.1
Type escape sequence to abort.
Tracing the route to 10.30.38.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.30.2 1 msec
    10.30.32.2 0 msec
    10.30.30.2 1 msec
  2 10.30.33.2 0 msec
    10.30.31.2 1 msec *
R30#traceroute 10.30.38.1
Type escape sequence to abort.
Tracing the route to 10.30.38.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.30.2 0 msec
    10.30.32.2 1 msec
    10.30.30.2 1 msec
  2 10.30.33.2 1 msec
    10.30.31.2 0 msec *
```



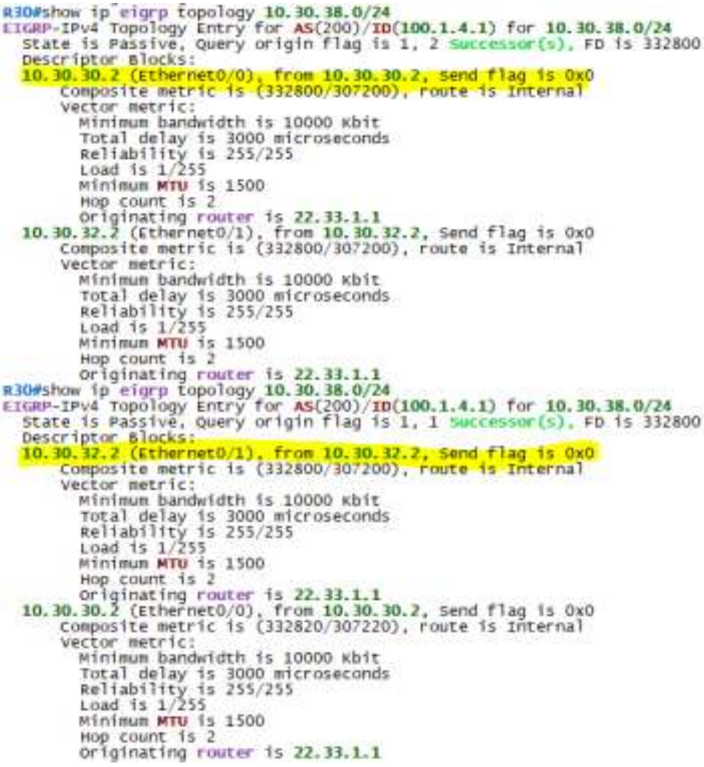
```
R30#show ip eigrp topology 10.30.38.0/24
EIGRP-IPv4 Topology Entry for AS(200)/ID(100.1.4.1) for 10.30.38.0/24
  State is Passive, Query origin flag is 1, 2 Successor(s), FD is 332800
  Descriptor Blocks:
  10.30.30.2 (Ethernet0/0), from 10.30.30.2, Send flag is 0x0
      Composite metric is (332800/307200), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
  10.30.32.2 (Ethernet0/1), from 10.30.32.2, Send flag is 0x0
      Composite metric is (332800/307200), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
```

Now we use offset config on R33 as below :



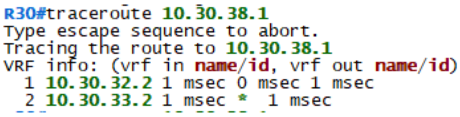
```
router eigrp 200
 network 10.0.0.0
 network 22.0.0.0
 offset-list 10 out 20 Ethernet0/1
 neighbor 10.30.31.1 Ethernet0/1
 neighbor 10.30.33.1 Ethernet0/0
 passive-interface default
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/0
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
!
access-list 10 permit 10.30.38.0 0.0.0.255
!
```

And Verify the results



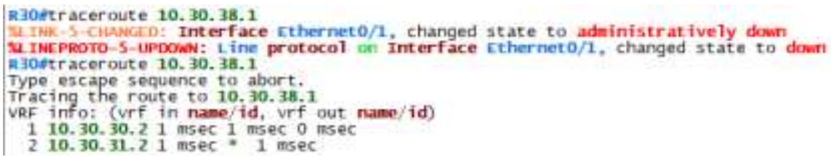
```
R30#show ip eigrp topology 10.30.38.0/24
EIGRP-IPv4 Topology Entry for AS(200)/ID(100.1.4.1) for 10.30.38.0/24
  State is Passive, Query origin flag is 1, 2 Successor(s), FD is 332800
  Descriptor Blocks:
  10.30.30.2 (Ethernet0/0), from 10.30.30.2, Send flag is 0x0
      Composite metric is (332800/307200), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
  10.30.32.2 (Ethernet0/1), from 10.30.32.2, Send flag is 0x0
      Composite metric is (332800/307200), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
R30#show ip eigrp topology 10.30.38.0/24
EIGRP-IPv4 Topology Entry for AS(200)/ID(100.1.4.1) for 10.30.38.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 332800
  Descriptor Blocks:
  10.30.32.2 (Ethernet0/1), from 10.30.32.2, Send flag is 0x0
      Composite metric is (332800/307200), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
  10.30.30.2 (Ethernet0/0), from 10.30.30.2, Send flag is 0x0
      Composite metric is (332820/307220), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
```

From R31 traceroute, we see it going via R32 no load-balancing.



```
R30#traceroute 10.30.38.1
Type escape sequence to abort.
Tracing the route to 10.30.38.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.32.2 1 msec 0 msec 1 msec
  2 10.30.33.2 1 msec *  1 msec
```

Lets check in case the link go down between R30 and R32, it will use R32 path to reach 10.30.38.0/24



```
R30#traceroute 10.30.38.1
%LINK-5-CHANGED: Interface Ethernet0/1, changed state to administratively down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/1, changed state to down
R30#traceroute 10.30.38.1
Type escape sequence to abort.
Tracing the route to 10.30.38.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.30.2 1 msec 1 msec 0 msec
  2 10.30.31.2 1 msec *  1 msec
```

Cisco reference:
https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/13673-14.html

2.5.e Implement and troubleshoot EIGRP stub

2.5.e [i] stub

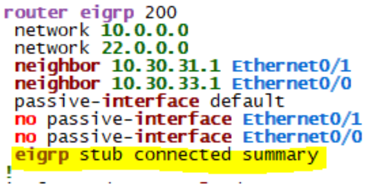
Stub Routers in EIGRP have two purposes:

1. to prevent a router from advertising any routes it has learnt via EIGRP to neighbouring routers
2. to limit the scope of query messages when a route goes “active”

Here are the flavors we have:

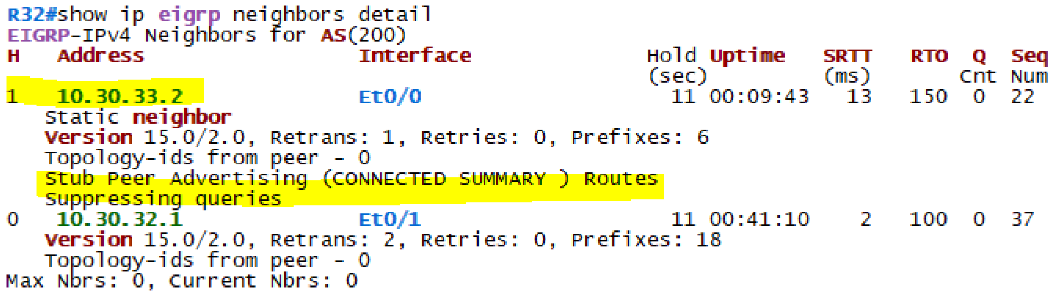
- Receive-only: The stub router will not advertise any network.
- Connected: allows the stub router to advertise directly connected networks.
- Static: allows the stub router to advertise static routes (you have to redistribute them).
- Summary: allows the stub router to advertise summary routes.
- Redistribute:allows the stub router to advertise redistributed routes.

Now we will make R33 as Stub router and verify the output



```
router eigrp 200
 network 10.0.0.0
 network 22.0.0.0
 neighbor 10.30.31.1 Ethernet0/1
 neighbor 10.30.33.1 Ethernet0/0
 passive-interface default
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/0
 eigrp stub connected summary
!
```

Verify the neighbour:



```
R32#show ip eigrp neighbors detail
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
1   10.30.33.2              Et0/0                    11 00:09:43   13   150  0  22
   Static neighbor
   Version 15.0/2.0, Retrans: 1, Retries: 0, Prefixes: 6
   Topology-ids from peer - 0
   Stub Peer Advertising (CONNECTED SUMMARY ) Routes
   Suppressing queries
0   10.30.32.1              Et0/1                    11 00:41:10    2   100  0  37
   Version 15.0/2.0, Retrans: 2, Retries: 0, Prefixes: 18
   Topology-ids from peer - 0
Max Nbrs: 0, Current Nbrs: 0
```

2.5.e [ii] leak-map

Same as above summary route leak map

2.5.f Implement and troubleshoot load-balancing

2.5.f [i] equal-cost

EIGRP Path Selection, EIGRP Metric Weights, EIGRP Traffic Engineering

EIGRP chooses the path with the lowest composite metric based on:

Bandwidth

--inverse lowest bandwidth along path in Kbps scaled by 10^7 * 256

Delay

--cumulative delay along path in tens of microseconds scaled by 256.

Load

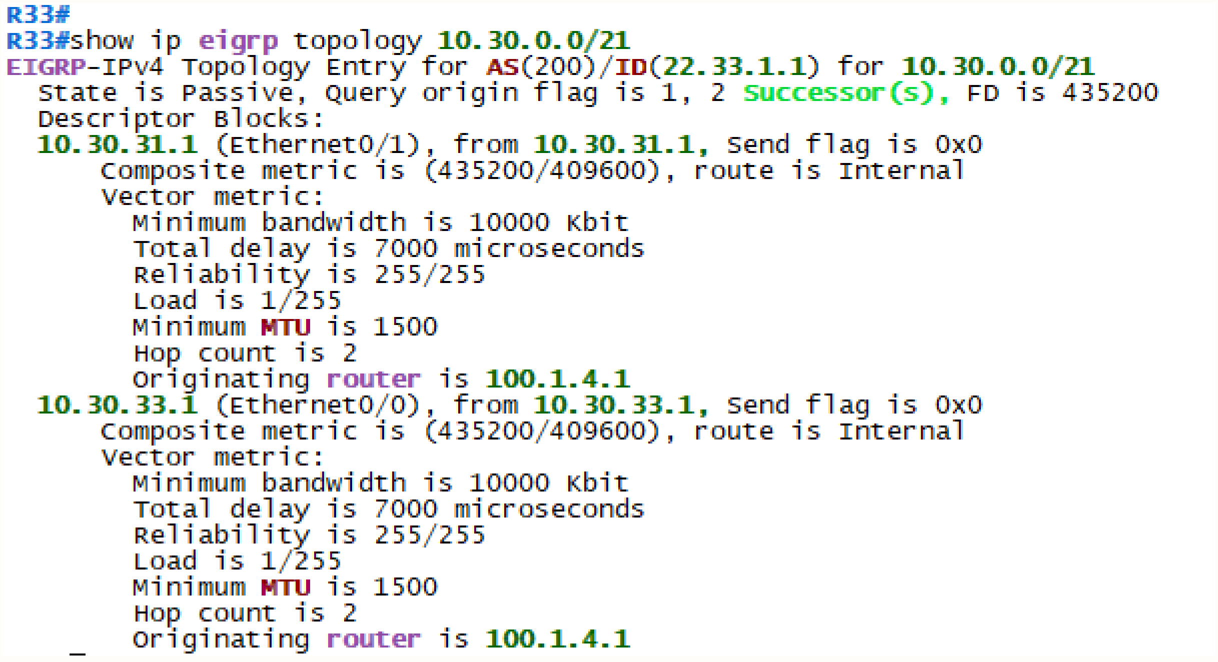
--Highest load along path

Reliability

--lowest reliability along path

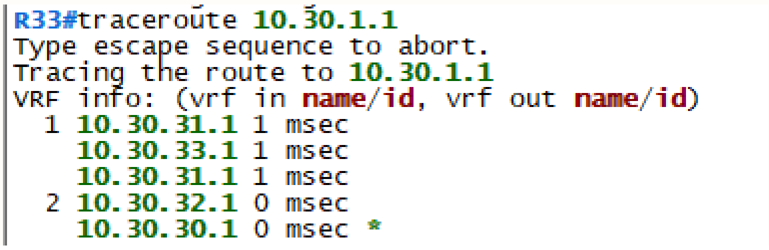
Easiest way on EIGRP Traffic Engineering with Delay is cumulative on a hop-by-hop basis
--easier to influence path selection with “Delay” interface command.

Now From R30 we got summary Route 10.30.0.0/21
Now we observe this summary route on R33 and we do Traffic engineer using delay adjustment



```
R33#
R33#show ip eigrp topology 10.30.0.0/21
EIGRP-IPv4 Topology Entry for AS(200)/ID(22.33.1.1) for 10.30.0.0/21
  State is Passive, Query origin flag is 1, 2 Successor(s), FD is 435200
  Descriptor Blocks:
  10.30.31.1 (Ethernet0/1), from 10.30.31.1, Send flag is 0x0
      Composite metric is (435200/409600), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 7000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 100.1.4.1
  10.30.33.1 (Ethernet0/0), from 10.30.33.1, Send flag is 0x0
      Composite metric is (435200/409600), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 7000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
    _   Originating router is 100.1.4.1
```

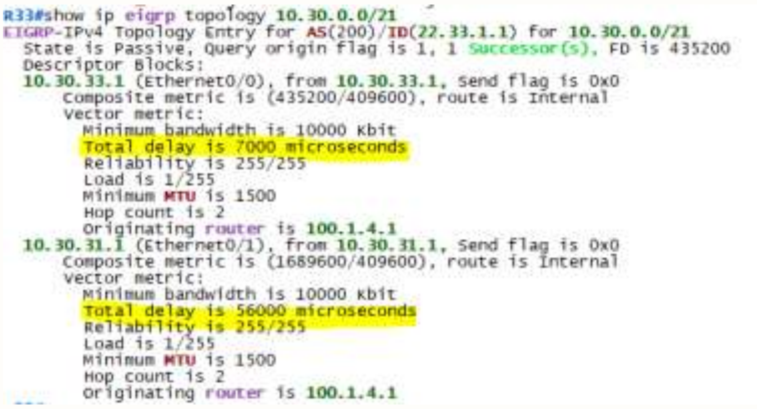
Lets test traceroute from R33 to 10.30.1.1



```
R33#traceroute 10.30.1.1
Type escape sequence to abort.
Tracing the route to 10.30.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.31.1 1 msec
    10.30.33.1 1 msec
    10.30.31.1 1 msec
  2 10.30.32.1 0 msec
    10.30.30.1 0 msec *
```

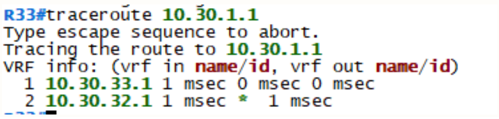
Now it Go from R33 to R31 then R30 was the final destination.

On R33 Eth 0/1 Link lets increase the delay another 5000, so traffic path will take from R33 to R32 then R30.



```
R33#show ip eigrp topology 10.30.0.0/21
EIGRP-IPv4 Topology Entry for AS(200)/ID(22.33.1.1) for 10.30.0.0/21
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 435200
  Descriptor Blocks:
  10.30.33.1 (Ethernet0/0), from 10.30.33.1, Send flag is 0x0
      Composite metric is (435200/409600), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 7000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 100.1.4.1
  10.30.31.1 (Ethernet0/1), from 10.30.31.1, Send flag is 0x0
      Composite metric is (1689600/409600), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 56000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 100.1.4.1
```

Now we do the same traceroute



```
R33#traceroute 10.30.1.1
Type escape sequence to abort.
Tracing the route to 10.30.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.33.1 1 msec 0 msec 0 msec
  2 10.30.32.1 1 msec *  1 msec
```

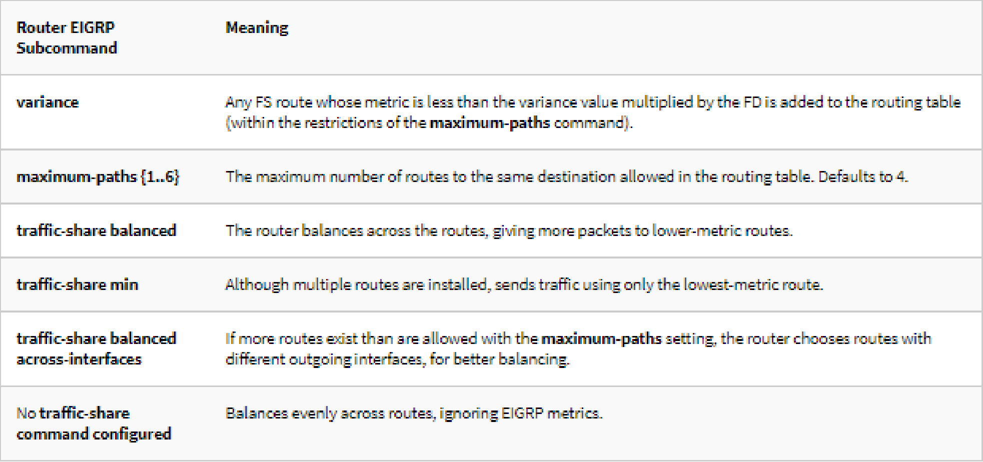
2.5.f [ii] unequal-cost

Routing would be more efficient if we balanced the load from those two available paths. The metrics of those two paths are equal or nearby. Since we cannot do un-equal path load-balance other link will get wasted by leaving idle. while letting the other path handle the full load for traffic.

That's where EIGRP unequal-cost load balancing comes in. EIGRP runs equal-cost load balancing by default, but unequal-cost balancing requires a little configuration, a little math, and an eye for details and unexpected results. We'll use all of those as we proceed with this lab.

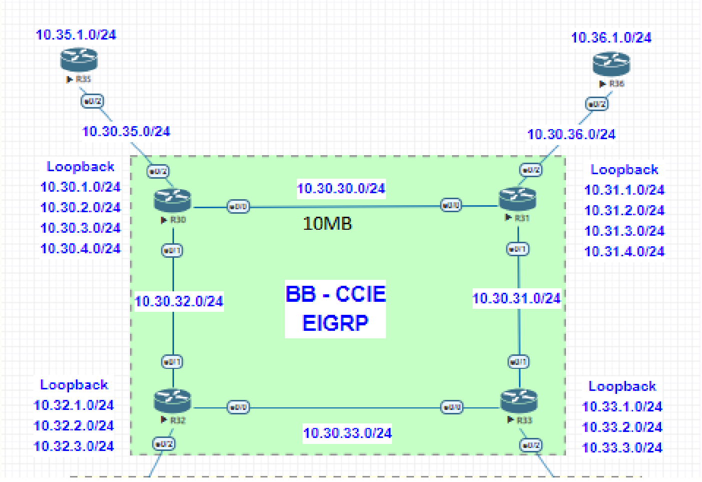
One magic word enables unequal-cost load balancing in EIGRP. The **variance** command is a multiplier and has a default value of 1. Using variance is a two-step process:

The router will multiple the Feasible Distance of the route by the variance value.
Any feasible successor with a metric less than that new value will be entered into the routing table.

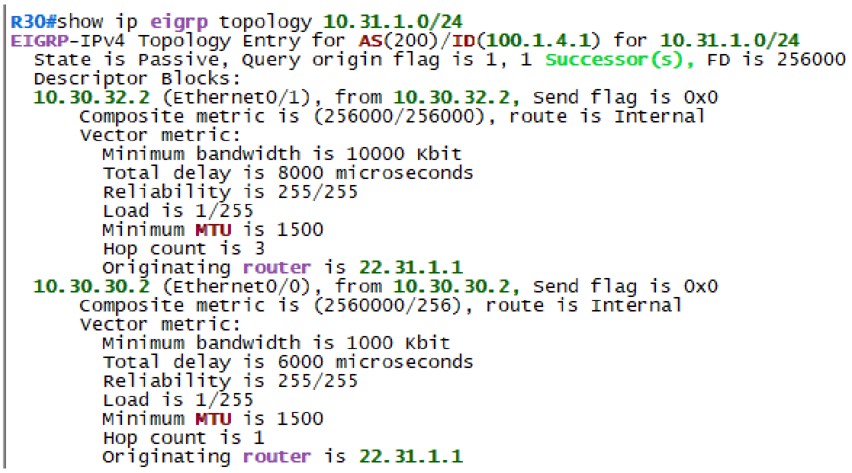


Router EIGRP Subcommand	Meaning
variance	Any FS route whose metric is less than the variance value multiplied by the FD is added to the routing table (within the restrictions of the **maximum-paths** command).
maximum-paths {1..6}	The maximum number of routes to the same destination allowed in the routing table. Defaults to 4.
traffic-share balanced	The router balances across the routes, giving more packets to lower-metric routes.
traffic-share min	Although multiple routes are installed, sends traffic using only the lowest-metric route.
traffic-share balanced across-interfaces	If more routes exist than are allowed with the **maximum-paths** setting, the router chooses routes with different outgoing interfaces, for better balancing.
No **traffic-share command configured**	Balances evenly across routes, ignoring EIGRP metrics.

Example: (in this topology we make between R30 R31 link speed 10MB and work with Variance to make un-equal path load share.

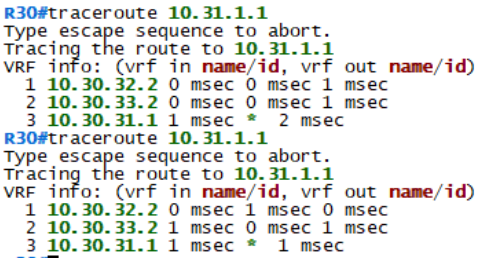


Check the Eigrp topology and traceroute for 10.31.1.1



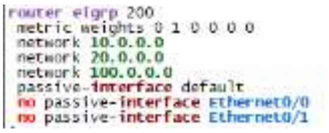
```
R30#show ip eigrp topology 10.31.1.0/24
EIGRP-IPv4 Topology Entry for AS(200)/ID(100.1.4.1) for 10.31.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 256000
  Descriptor Blocks:
  10.30.32.2 (Ethernet0/1), from 10.30.32.2, Send flag is 0x0
      Composite metric is (256000/256000), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 8000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 3
        Originating router is 22.31.1.1
  10.30.30.2 (Ethernet0/0), from 10.30.30.2, Send flag is 0x0
      Composite metric is (2560000/256), route is Internal
      Vector metric:
        Minimum bandwidth is 1000 Kbit
        Total delay is 6000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
        Originating router is 22.31.1.1
```

Traceroute always use 100MB link even though R31 is next hop.



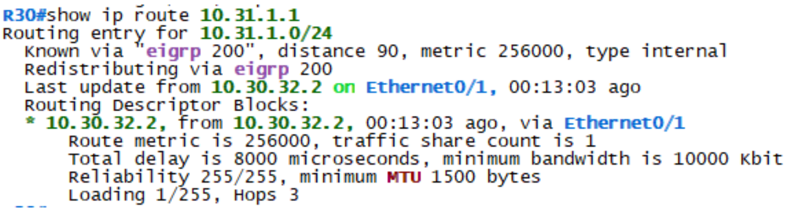
```
R30#traceroute 10.31.1.1
Type escape sequence to abort.
Tracing the route to 10.31.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.32.2 0 msec 0 msec 1 msec
  2 10.30.33.2 0 msec 0 msec 1 msec
  3 10.30.31.1 1 msec *  2 msec
R30#traceroute 10.31.1.1
Type escape sequence to abort.
Tracing the route to 10.31.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.32.2 0 msec 1 msec 0 msec
  2 10.30.33.2 1 msec 0 msec 1 msec
  3 10.30.31.1 1 msec *  1 msec
```

Since EIGRP rely to calculate feasible success based on the formula.
For simplicity I have configured EIGRP to only use the **Bandwidth K** value when calculating Feasible Distances.



```
router eigrp 200
 metric weights 0 1 0 0 0 0
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
```

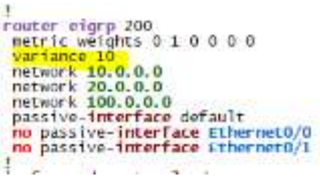
Now before variance added lets check the routing table.



```
R30#show ip route 10.31.1.1
Routing entry for 10.31.1.0/24
  Known via "eigrp 200", distance 90, metric 256000, type internal
  Redistributing via eigrp 200
  Last update from 10.30.32.2 on Ethernet0/1, 00:13:03 ago
  Routing Descriptor Blocks:
  * 10.30.32.2, from 10.30.32.2, 00:13:03 ago, via Ethernet0/1
      Route metric is 256000, traffic share count is 1
      Total delay is 8000 microseconds, minimum bandwidth is 10000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 3
```

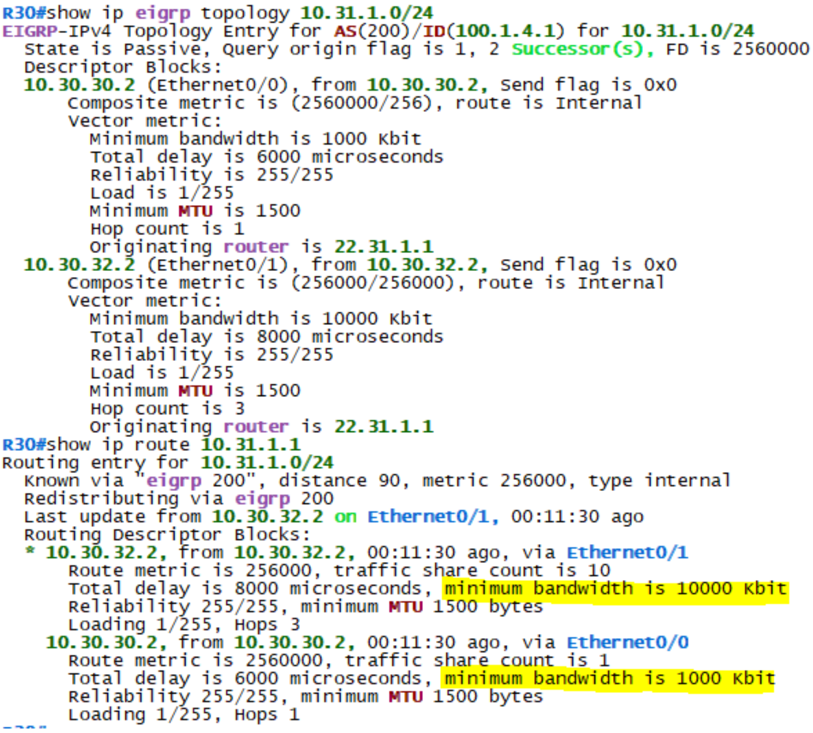
This still used R32 as route path to reach R31 Loopback interface.

I have just added Variance 10 on R30 Eigrp process



```
!
router eigrp 200
 metric weights 0 1 0 0 0 0
 variance 10
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
!
```

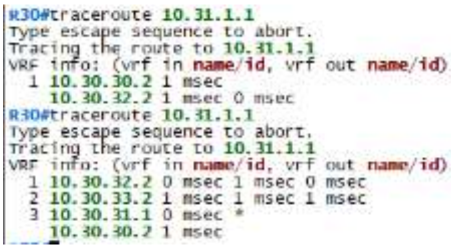
How lets verify:



```
R30#show ip eigrp topology 10.31.1.0/24
EIGRP-IPv4 Topology Entry for AS(200)/ID(100.1.4.1) for 10.31.1.0/24
  State is Passive, Query origin flag is 1, 2 Successor(s), FD is 2560000
  Descriptor Blocks:
  10.30.30.2 (Ethernet0/0), from 10.30.30.2, Send flag is 0x0
      Composite metric is (2560000/256), route is Internal
      Vector metric:
        Minimum bandwidth is 1000 Kbit
        Total delay is 6000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
        Originating router is 22.31.1.1
  10.30.32.2 (Ethernet0/1), from 10.30.32.2, Send flag is 0x0
      Composite metric is (256000/256000), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 8000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 3
        Originating router is 22.31.1.1
R30#show ip route 10.31.1.1
Routing entry for 10.31.1.0/24
  Known via "eigrp 200", distance 90, metric 256000, type internal
  Redistributing via eigrp 200
  Last update from 10.30.32.2 on Ethernet0/1, 00:11:30 ago
  Routing Descriptor Blocks:
  * 10.30.32.2, from 10.30.32.2, 00:11:30 ago, via Ethernet0/1
      Route metric is 256000, traffic share count is 10
      Total delay is 8000 microseconds, minimum bandwidth is 10000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 3
    10.30.30.2, from 10.30.30.2, 00:11:30 ago, via Ethernet0/0
      Route metric is 2560000, traffic share count is 1
      Total delay is 6000 microseconds, minimum bandwidth is 1000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1
```

Let’s see traceroute load-balance to 10.31.1.1

We can observe now taking 2 paths and loadbalancing between R31 and R32



```
R30#traceroute 10.31.1.1
Type escape sequence to abort.
Tracing the route to 10.31.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.30.2 1 msec
    10.30.32.2 1 msec 0 msec
R30#traceroute 10.31.1.1
Type escape sequence to abort.
Tracing the route to 10.31.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.30.32.2 0 msec 1 msec 0 msec
  2 10.30.33.2 1 msec 1 msec 1 msec
  3 10.30.31.1 0 msec *
    10.30.30.2 1 msec
```

2.5.f [iii] add-path (Covered in DMVPN)

The EIGRP Add Path Support is a feature that allows the hub in a DMVPN topology to advertise multiple best paths to its spoke routers. This feature is needed if you have two or more spoke routers that advertise the same subnet. The hub router will learn about the subnet from both spoke routers so it can use ECMP (Equal Cost Multipath) with both spoke routers. EIGRP however, will only advertise one path to other spoke routers. Without this feature, you can't use ECMP between spoke routers and when one of your spoke routers fail, EIGRP has to re-converge.

2.5.g Implement EIGRP [multi-address] named mode

2.5.g [i] Types of families

2.5.g [ii] IPv4 address-family

2.5.g [iii] IPv6 address-family

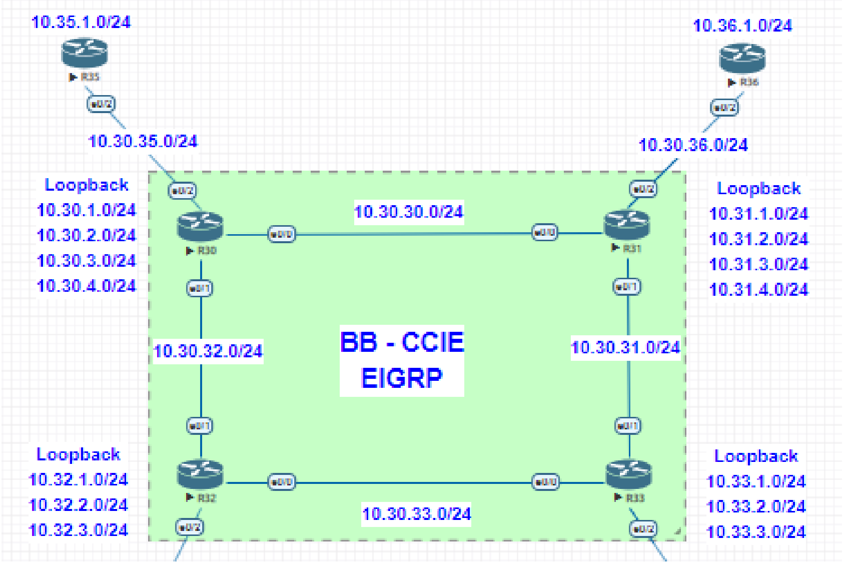
The traditional way to configure EIGRP requires various parameters to be configured under the interface and EIGRP configuration mode. In order to configure EIGRP IPV4 and IPv6, it is required to configure separate EIGRP instances. Traditional EIGRP does not support Virtual Routing and Forwarding (VRF) in IPv6 EIGRP implementations.
With Named mode EIGRP, everything is configured at a single place under the EIGRP configuration and there are no restrictions as mentioned previously.

EIGRP Named Mode or Multi-AF Mode is a new development in EIGRP starting in Version 15.x
EIGRP name mode have backward compatible with Classic mode neighbour also.

Only a single instance of EIGRP needs to be created. It can be used for all address family types. It also supports multiple VRFs limited only by available system resources. One thing to be aware of in regards to the named mode is that configuration of the address-family does not enable IPv4 routing as a traditional configuration of IPv4 EIGRP. A 'no shut' is required in order to start the process:

router eigrp [virtual-instance-name | asystem]
[no] shutdown

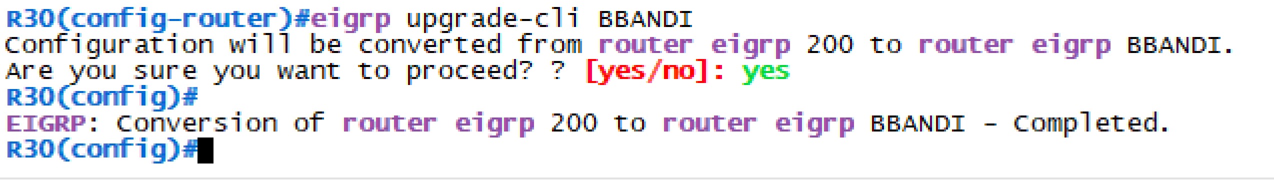
On this topology we run EIGRP Named mode on R30 and R33 for testing, and R31 and R32 remain Classic Mode



Named EIGRP has three modes under which the bulk of the configuration is completed. These are:

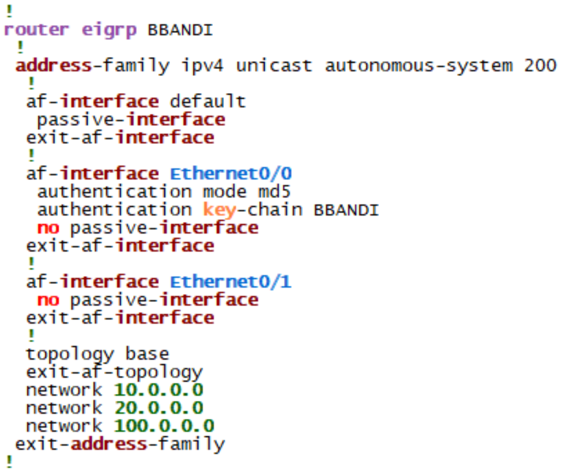
- address-family configuration mode - (config-router-af)#
- address-family interface configuration mode - (config-router-af-interface)#
- address-family topology configuration mode - (config-router-af-topology)#

Since R30 already have classic mode config, we upgrade to Named mode with existing config. As below :

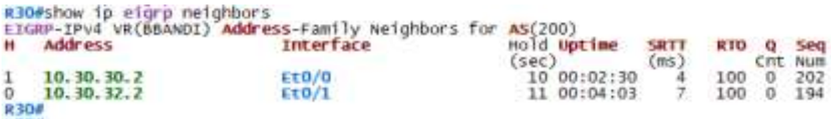


```
R30(config-router)#eigrp upgrade-cli BBANDI
Configuration will be converted from router eigrp 200 to router eigrp BBANDI.
Are you sure you want to proceed? ? [yes/no]: yes
R30(config)#
EIGRP: Conversion of router eigrp 200 to router eigrp BBANDI - Completed.
R30(config)#
```

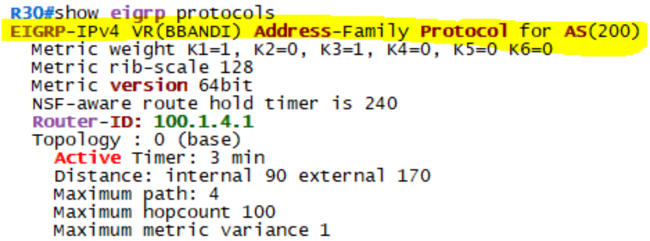
Verify the config and neighbours:



```
!
router eigrp BBANDI
 !
 address-family ipv4 unicast autonomous-system 200
  !
  af-interface default
   passive-interface
  exit-af-interface
  !
  af-interface Ethernet0/0
   authentication mode md5
   authentication key-chain BBANDI
   no passive-interface
  exit-af-interface
  !
  af-interface Ethernet0/1
   no passive-interface
  exit-af-interface
  !
  topology base
  exit-af-topology
  network 10.0.0.0
  network 20.0.0.0
  network 100.0.0.0
 exit-address-family
!
```

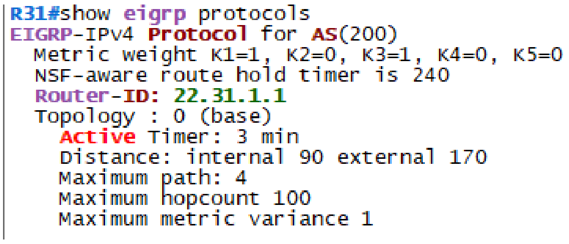


```
R30#show ip eigrp neighbors
EIGRP-IPv4 VR(BBANDI) Address-Family Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
1   10.30.30.2              Et0/0                    10 00:02:30    4   100  0  202
0   10.30.32.2              Et0/1                    11 00:04:03    7   100  0  194
R30#
```



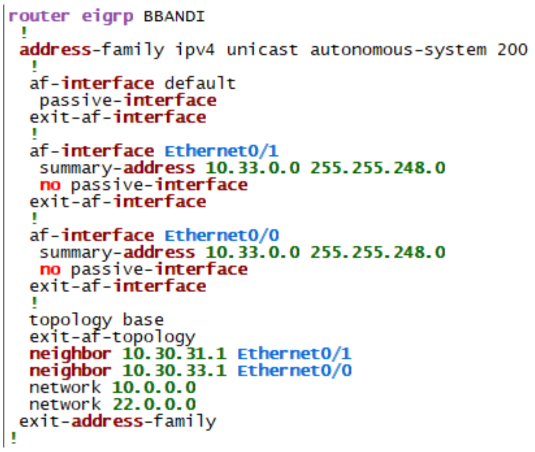
```
R30#show eigrp protocols
EIGRP-IPv4 VR(BBANDI) Address-Family Protocol for AS(200)
  Metric weight K1=1, K2=0, K3=1, K4=0, K5=0 K6=0
  Metric rib-scale 128
  Metric version 64bit
  NSF-aware route hold timer is 240
  Router-ID: 100.1.4.1
  Topology : 0 (base)
    Active Timer: 3 min
    Distance: internal 90 external 170
    Maximum path: 4
    Maximum hopcount 100
    Maximum metric variance 1
```

Classic mode as below :

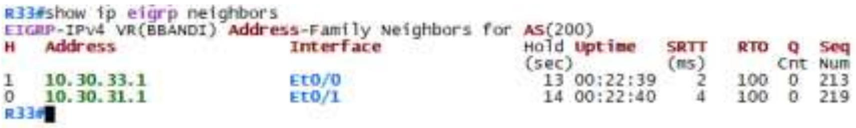


```
R31#show eigrp protocols
EIGRP-IPv4 Protocol for AS(200)
  Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  NSF-aware route hold timer is 240
  Router-ID: 22.31.1.1
  Topology : 0 (base)
    Active Timer: 3 min
    Distance: internal 90 external 170
    Maximum path: 4
    Maximum hopcount 100
    Maximum metric variance 1
```

Now we configure R33 with summarisation and verify

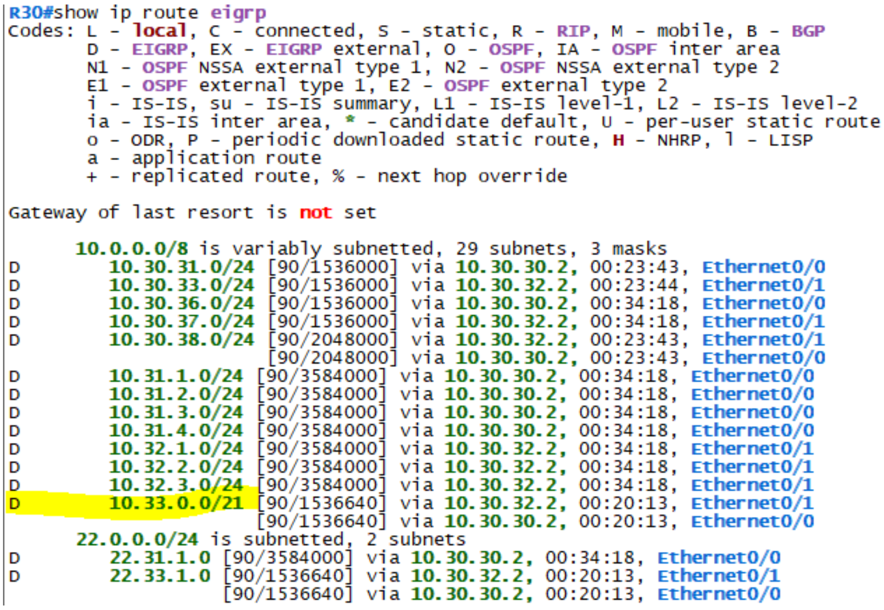


```
router eigrp BBANDI
 !
 address-family ipv4 unicast autonomous-system 200
  !
  af-interface default
   passive-interface
  exit-af-interface
  !
  af-interface Ethernet0/1
   summary-address 10.33.0.0 255.255.248.0
   no passive-interface
  exit-af-interface
  !
  af-interface Ethernet0/0
   summary-address 10.33.0.0 255.255.248.0
   no passive-interface
  exit-af-interface
  !
  topology base
  exit-af-topology
  neighbor 10.30.31.1 Ethernet0/1
  neighbor 10.30.33.1 Ethernet0/0
  network 10.0.0.0
  network 22.0.0.0
 exit-address-family
!
```



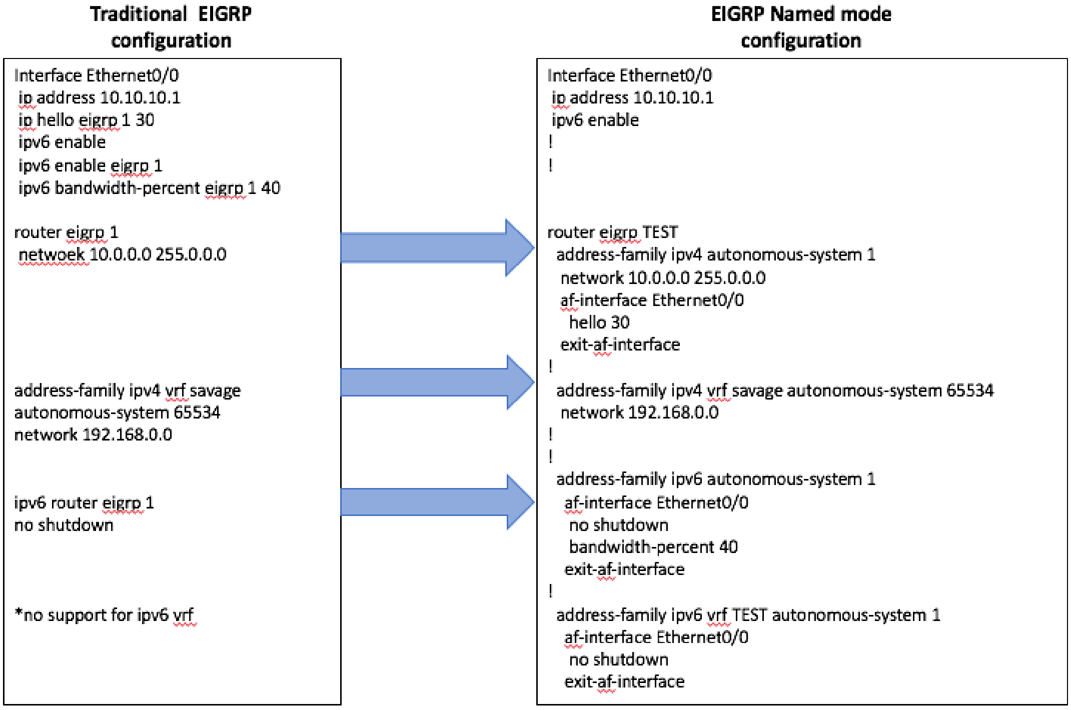
```
R33#show ip eigrp neighbors
EIGRP-IPv4 VR(BBANDI) Address-Family Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
1   10.30.33.1              Et0/0                    13 00:22:39    2   100  0  213
0   10.30.31.1              Et0/1                    14 00:22:40    4   100  0  219
R33#
```

On R30 lets check the summarisation for 10.33.0.0/21



```
R30#show ip route eigrp
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 29 subnets, 3 masks
D        10.30.31.0/24 [90/1536000] via 10.30.30.2, 00:23:43, Ethernet0/0
D        10.30.33.0/24 [90/1536000] via 10.30.32.2, 00:23:44, Ethernet0/1
D        10.30.36.0/24 [90/1536000] via 10.30.30.2, 00:34:18, Ethernet0/0
D        10.30.37.0/24 [90/1536000] via 10.30.32.2, 00:34:18, Ethernet0/1
D        10.30.38.0/24 [90/2048000] via 10.30.32.2, 00:23:43, Ethernet0/1
                       [90/2048000] via 10.30.30.2, 00:23:43, Ethernet0/0
D        10.31.1.0/24 [90/3584000] via 10.30.30.2, 00:34:18, Ethernet0/0
D        10.31.2.0/24 [90/3584000] via 10.30.30.2, 00:34:18, Ethernet0/0
D        10.31.3.0/24 [90/3584000] via 10.30.30.2, 00:34:18, Ethernet0/0
D        10.31.4.0/24 [90/3584000] via 10.30.30.2, 00:34:18, Ethernet0/0
D        10.32.1.0/24 [90/3584000] via 10.30.32.2, 00:34:18, Ethernet0/1
D        10.32.2.0/24 [90/3584000] via 10.30.32.2, 00:34:18, Ethernet0/1
D        10.32.3.0/24 [90/3584000] via 10.30.32.2, 00:34:18, Ethernet0/1
D        10.33.0.0/21 [90/1536640] via 10.30.32.2, 00:20:13, Ethernet0/1
                      [90/1536640] via 10.30.30.2, 00:20:13, Ethernet0/0
      22.0.0.0/24 is subnetted, 2 subnets
D        22.31.1.0 [90/3584000] via 10.30.30.2, 00:34:18, Ethernet0/0
D        22.33.1.0 [90/1536640] via 10.30.32.2, 00:20:13, Ethernet0/1
                   [90/1536640] via 10.30.30.2, 00:20:13, Ethernet0/0
```

Here is table conversion for reference:



Traditional EIGRP configuration

```
Interface Ethernet0/0
 ip address 10.10.10.1
 ip hello eigrp 1 30
 ipv6 enable
 ipv6 enable eigrp 1
 ipv6 bandwidth-percent eigrp 1 40

router eigrp 1
 netwoek 10.0.0.0 255.0.0.0



address-family ipv4 vrf savage
autonomous-system 65534
network 192.168.0.0


ipv6 router eigrp 1
no shutdown


*no support for ipv6 vrf
```

EIGRP Named mode configuration

```
Interface Ethernet0/0
 ip address 10.10.10.1
 ipv6 enable
!
!

router eigrp TEST
  address-family ipv4 autonomous-system 1
   network 10.0.0.0 255.0.0.0
   af-interface Ethernet0/0
     hello 30
   exit-af-interface
!
  address-family ipv4 vrf savage autonomous-system 65534
   network 192.168.0.0
!
!
  address-family ipv6 autonomous-system 1
   af-interface Ethernet0/0
    no shutdown
    bandwidth-percent 40
   exit-af-interface
!
  address-family ipv6 vrf TEST autonomous-system 1
   af-interface Ethernet0/0
    no shutdown
   exit-af-interface
```

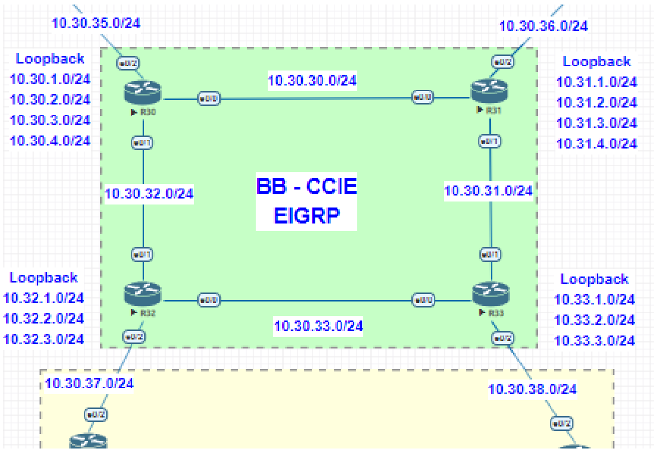
2.5.h Implement, troubleshoot and optimize EIGRP convergence and scalability

2.5.h [i] Describe fast convergence requirements

2.5.h [ii] Control query boundaries

2.5.h [iii] IP FRR/fast reroute [single hop]

EIGRP Loop-Free Alternate (LFA) Fast Reroute (FRR) is a feature that allows EIGRP to switch to a backup path in less than 50 ms. Fast reroute means we switch to another next hop, Loop-free alternate is an alternative path in the network that is loop free.



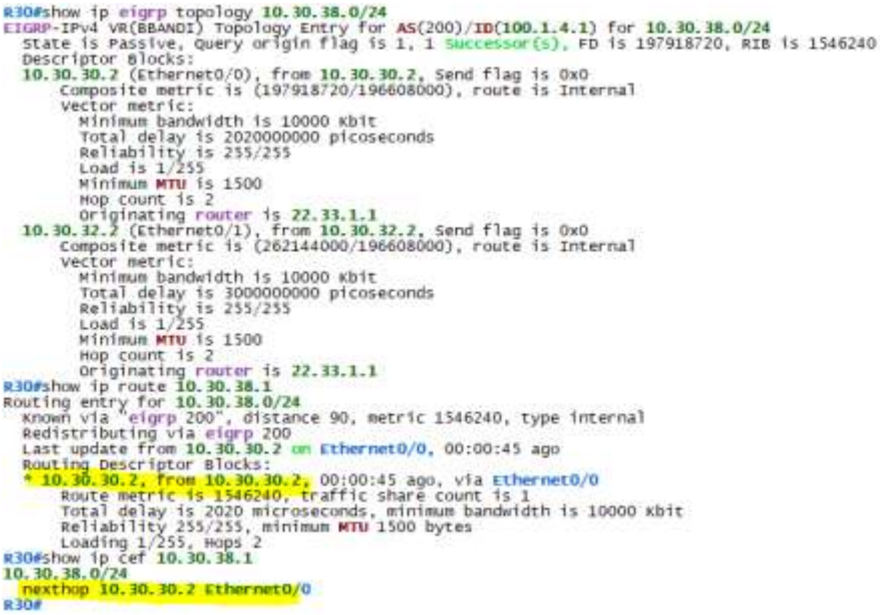
In this example R31 to reach 10.30.38.1 It has 2 routes, from R31 and R32.

```
R30#show ip eigrp topology 10.30.38.0/24
EIGRP-IPv4 VR(BBANDI) Topology Entry for AS(200)/ID(100.1.4.1) for 10.30.38.0/24
  State is Passive, Query origin flag is 1, 2 Successor(s), FD is 197918720, RIB is 1546240
  Descriptor Blocks:
  10.30.30.2 (Ethernet0/0), from 10.30.30.2, Send flag is 0x0
      Composite metric is (262144000/196608000), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000000000 picoseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
  10.30.32.2 (Ethernet0/1), from 10.30.32.2, Send flag is 0x0
      Composite metric is (262144000/196608000), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000000000 picoseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
R30#show ip route 10.30.38.1
Routing entry for 10.30.38.0/24
  Known via "eigrp 200", distance 90, metric 2048000, type internal
  Redistributing via eigrp 200
  Last update from 10.30.32.2 on Ethernet0/1, 00:00:11 ago
  Routing Descriptor Blocks:
    10.30.32.2, from 10.30.32.2, 00:00:11 ago, via Ethernet0/1
      Route metric is 2048000, traffic share count is 1
      Total delay is 3000 microseconds, minimum bandwidth is 10000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
  * 10.30.30.2, from 10.30.30.2, 00:00:11 ago, via Ethernet0/0
      Route metric is 2048000, traffic share count is 1
      Total delay is 3000 microseconds, minimum bandwidth is 10000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
R30#show ip cef 10.30.38.1
10.30.38.0/24
  nexthop 10.30.30.2 Ethernet0/0
  nexthop 10.30.32.2 Ethernet0/1
R30#
```

Now we increase delay on R30 Interface E0/1

```
R30#show run interface ethernet 0/0
Building configuration...

Current configuration : 75 bytes
!
interface Ethernet0/0
 ip address 10.30.30.1 255.255.255.0
 delay 2
end
```

Now check the same output again : (now we have only 1 path to reach 10.30.38.1)



```
R30#show ip eigrp topology 10.30.38.0/24
EIGRP-IPv4 VR(BBANDI) Topology Entry for AS(200)/ID(100.1.4.1) for 10.30.38.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 197918720, RIB is 1546240
  Descriptor Blocks:
  10.30.30.2 (Ethernet0/0), from 10.30.30.2, Send flag is 0x0
      Composite metric is (197918720/196608000), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 2020000000 picoseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
  10.30.32.2 (Ethernet0/1), from 10.30.32.2, Send flag is 0x0
      Composite metric is (262144000/196608000), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 3000000000 picoseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 22.33.1.1
R30#show ip route 10.30.38.1
Routing entry for 10.30.38.0/24
  Known via "eigrp 200", distance 90, metric 1546240, type internal
  Redistributing via eigrp 200
  Last update from 10.30.30.2 on Ethernet0/0, 00:00:45 ago
  Routing Descriptor Blocks:
  * 10.30.30.2, from 10.30.30.2, 00:00:45 ago, via Ethernet0/0
      Route metric is 1546240, traffic share count is 1
      Total delay is 2020 microseconds, minimum bandwidth is 10000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
R30#show ip cef 10.30.38.1
10.30.38.0/24
  nexthop 10.30.30.2 Ethernet0/0
R30#
```

FRR (this feature not tested - only works on CSRV1000)

```
router eigrp BBANDI
address-family ipv4 unicast autonomous-system 200
topology base
fast-reroute per-prefix all
```

2.5.h [iv] Summary leak-map

Allows for more specific routes to be advertised that normally would be suppressed by the summary route

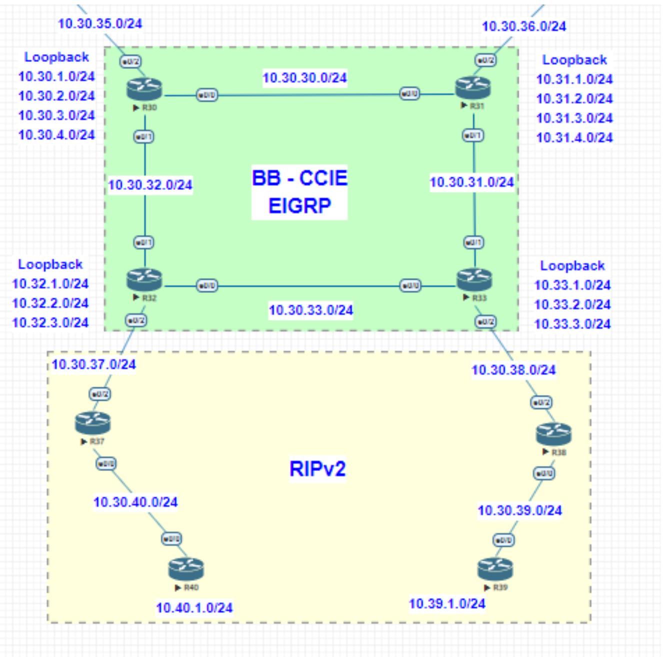
2.5.h [v] Summary metric

By default, summary routes use the lowest metric among the existing routes. If this metric changes, the summary route will also be updated.

The summary metric can be manually configured under the EIGRP process

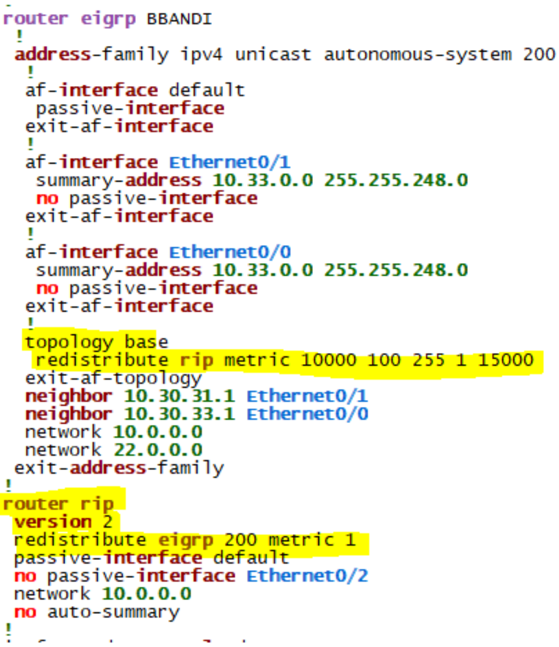
```
#router eigrp 200
#summary-metric 10.0.0.0/16 10000 200 255 0 1500
```

Redistribution from RIP to EIGRIP and EIGRP to RIP with below Network Topology:



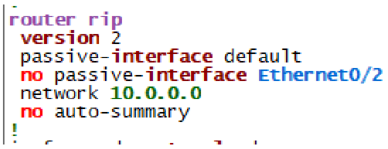
Configure R33 with EIGRIP and RIP and Reattribute the network and Verify:

Configure RIP on R33 and R38 and redistribute and very the output



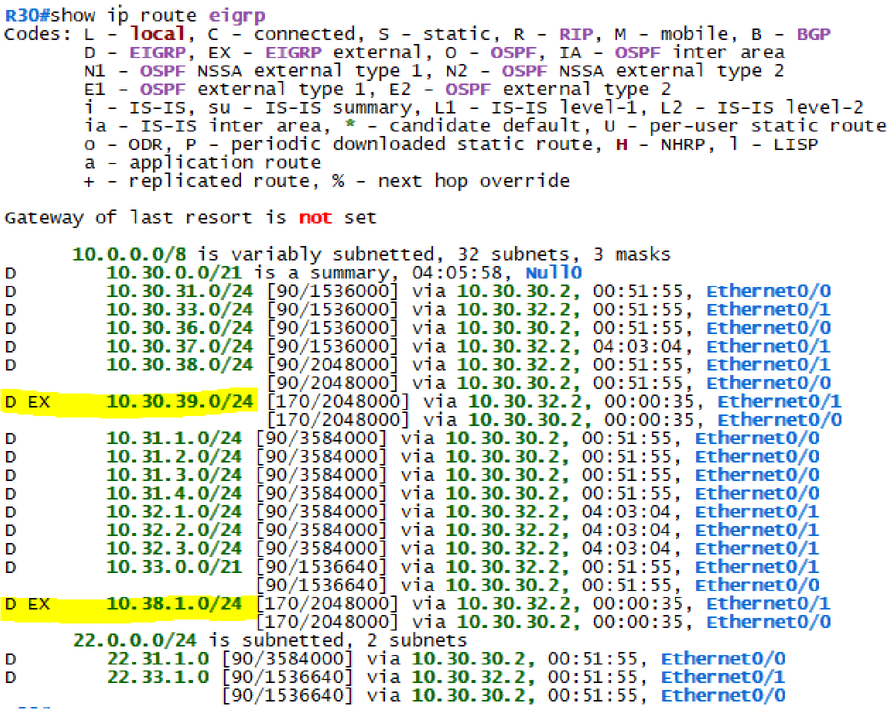
```
router eigrp BBANDI
 !
 address-family ipv4 unicast autonomous-system 200
  !
  af-interface default
   passive-interface
  exit-af-interface
  !
  af-interface Ethernet0/1
   summary-address 10.33.0.0 255.255.248.0
   no passive-interface
  exit-af-interface
  !
  af-interface Ethernet0/0
   summary-address 10.33.0.0 255.255.248.0
   no passive-interface
  exit-af-interface
  !
  topology base
   redistribute rip metric 10000 100 255 1 15000
  exit-af-topology
  neighbor 10.30.31.1 Ethernet0/1
  neighbor 10.30.33.1 Ethernet0/0
  network 10.0.0.0
  network 22.0.0.0
 exit-address-family
!
router rip
 version 2
 redistribute eigrp 200 metric 1
 passive-interface default
 no passive-interface Ethernet0/2
 network 10.0.0.0
 no auto-summary
!
```

R38



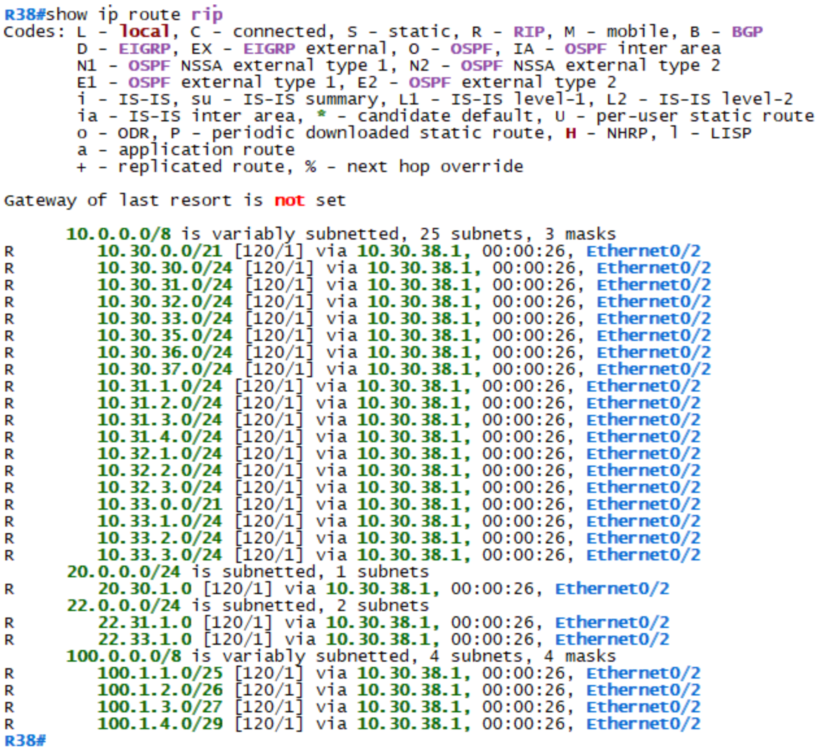
```
router rip
 version 2
 passive-interface default
 no passive-interface Ethernet0/2
 network 10.0.0.0
 no auto-summary
!
```

Check the route on R30 and R38



```
R30#show ip route eigrp
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 32 subnets, 3 masks
D        10.30.0.0/21 is a summary, 04:05:58, Null0
D        10.30.31.0/24 [90/1536000] via 10.30.30.2, 00:51:55, Ethernet0/0
D        10.30.33.0/24 [90/1536000] via 10.30.32.2, 00:51:55, Ethernet0/1
D        10.30.36.0/24 [90/1536000] via 10.30.30.2, 00:51:55, Ethernet0/0
D        10.30.37.0/24 [90/1536000] via 10.30.32.2, 04:03:04, Ethernet0/1
D        10.30.38.0/24 [90/2048000] via 10.30.32.2, 00:51:55, Ethernet0/1
                       [90/2048000] via 10.30.30.2, 00:51:55, Ethernet0/0
D EX     10.30.39.0/24 [170/2048000] via 10.30.32.2, 00:00:35, Ethernet0/1
                       [170/2048000] via 10.30.30.2, 00:00:35, Ethernet0/0
D        10.31.1.0/24 [90/3584000] via 10.30.30.2, 00:51:55, Ethernet0/0
D        10.31.2.0/24 [90/3584000] via 10.30.30.2, 00:51:55, Ethernet0/0
D        10.31.3.0/24 [90/3584000] via 10.30.30.2, 00:51:55, Ethernet0/0
D        10.31.4.0/24 [90/3584000] via 10.30.30.2, 00:51:55, Ethernet0/0
D        10.32.1.0/24 [90/3584000] via 10.30.32.2, 04:03:04, Ethernet0/1
D        10.32.2.0/24 [90/3584000] via 10.30.32.2, 04:03:04, Ethernet0/1
D        10.32.3.0/24 [90/3584000] via 10.30.32.2, 04:03:04, Ethernet0/1
D        10.33.0.0/21 [90/1536640] via 10.30.32.2, 00:51:55, Ethernet0/1
                      [90/1536640] via 10.30.30.2, 00:51:55, Ethernet0/0
D EX     10.38.1.0/24 [170/2048000] via 10.30.32.2, 00:00:35, Ethernet0/1
                      [170/2048000] via 10.30.30.2, 00:00:35, Ethernet0/0
      22.0.0.0/24 is subnetted, 2 subnets
D        22.31.1.0 [90/3584000] via 10.30.30.2, 00:51:55, Ethernet0/0
D        22.33.1.0 [90/1536640] via 10.30.32.2, 00:51:55, Ethernet0/1
                   [90/1536640] via 10.30.30.2, 00:51:55, Ethernet0/0
```

R38 we can see all routes from Network



```
R38#show ip route rip
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 25 subnets, 3 masks
R        10.30.0.0/21 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.30.30.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.30.31.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.30.32.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.30.33.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.30.35.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.30.36.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.30.37.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.31.1.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.31.2.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.31.3.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.31.4.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.32.1.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.32.2.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.32.3.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.33.0.0/21 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.33.1.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.33.2.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        10.33.3.0/24 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
      20.0.0.0/24 is subnetted, 1 subnets
R        20.30.1.0 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
      22.0.0.0/24 is subnetted, 2 subnets
R        22.31.1.0 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        22.33.1.0 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
      100.0.0.0/8 is variably subnetted, 4 subnets, 4 masks
R        100.1.1.0/25 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        100.1.2.0/26 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        100.1.3.0/27 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R        100.1.4.0/29 [120/1] via 10.30.38.1, 00:00:26, Ethernet0/2
R38#
```

EIGRP helpful commands

show running-config eigrp	Displays EIGRP running configuration information.
show ip eigrp topology	command to determine Diffusing Update Algorithm (DUAL) states and to debug possible DUAL problems.
show ip eigrp topology all-links	all the entries in the EIGRP topology table
show ip eigrp topology detail-links	display the detailed information for all entries in the EIGRP topology table:
show ip eigrp topology summary	display a summary of the topology table:
show ip eigrp topology active	how to display the active entries in the topology table
show ip eigrp topology zero-successors	display zero-successors in the topology table:
show ip eigrp topology pending	display pending entries:

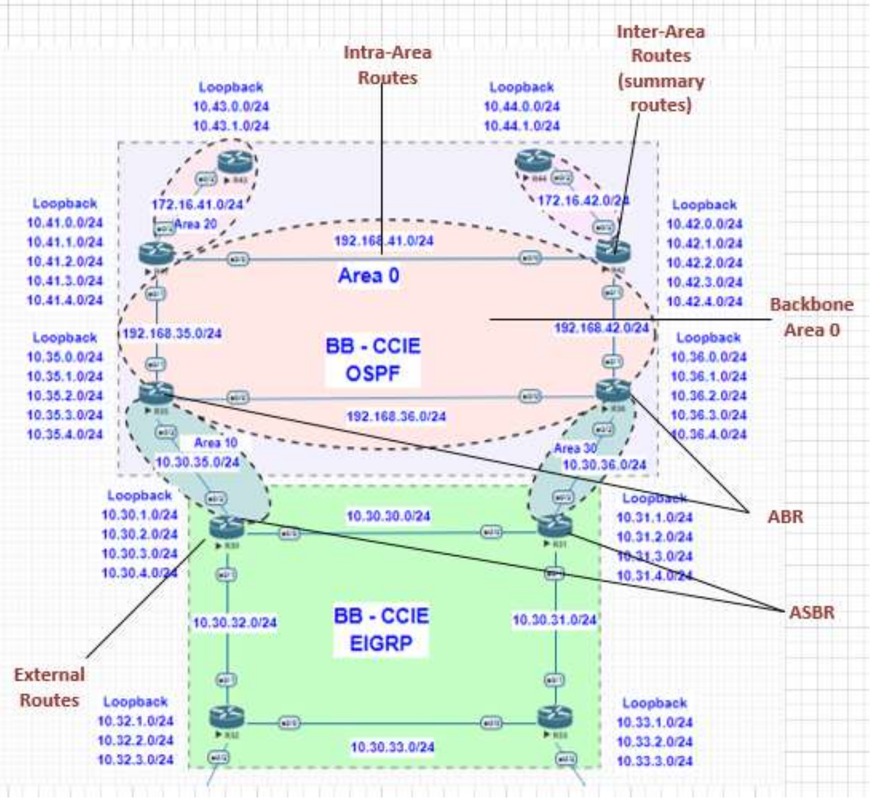
2.6 OSPF [v2 and v3]

https://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/7039-1.html#t6

OSPF in brief

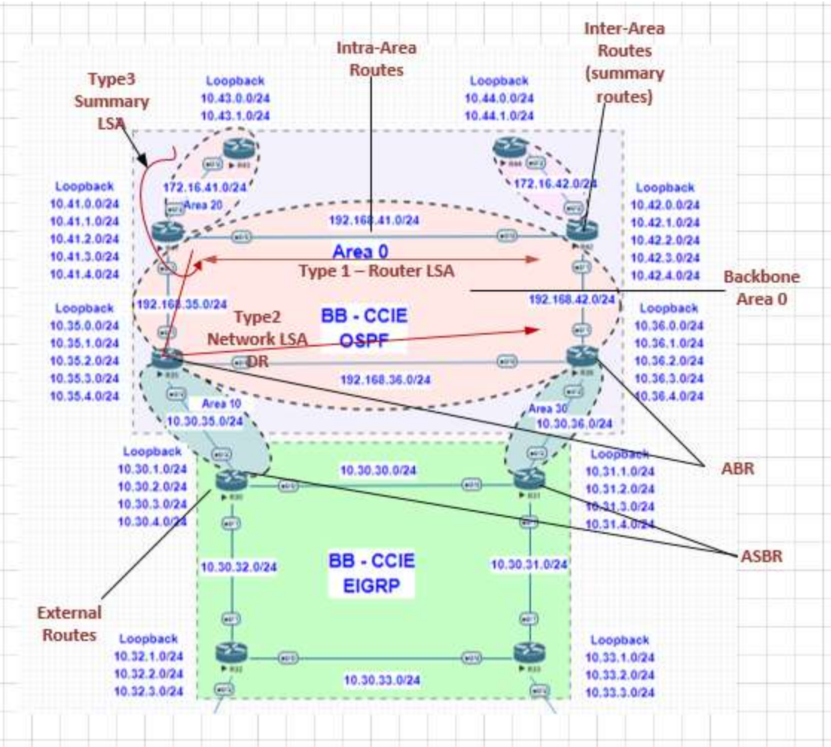
- With OSPF, there is no limitation on the hop count.
- The intelligent use of VLSM is very useful in IP address allocation.
- OSPF uses IP multicast to send link-state updates. This ensures less processing on routers that are not listening to OSPF packets. Also, updates are only sent in case routing changes occur instead of periodically. This ensures a better use of bandwidth.
- OSPF has better convergence than RIP. This is because routing changes are propagated instantaneously and not periodically.
- OSPF allows for better load balancing.
- OSPF allows for a logical definition of networks where routers can be divided into areas. This limits the explosion of link state updates over the whole network. This also provides a mechanism for aggregating routes and cutting down on the unnecessary propagation of subnet information.
- OSPF allows for routing authentication by using different methods of password authentication.
- OSPF allows for the transfer and tagging of external routes injected into an Autonomous System. This keeps track of external routes injected by exterior protocols such as BGP.

Areas and Border Routers - In the Lab diagram show R35 and R36 are ABR, R30 and R31 act as ASBR



2.6.a Describe packet types

2.6.a [i] LSA types [1, 2, 3, 4, 5, 7, 9]



LSA type 1

Router LSA: the router advertises its presence along with the links to other routers or networks and their metrics within the same area. Type 1 LSAs are flooded only in their own area. The originating router provides the link state id for type 1 link state advertisements.

LSA type 2

Network LSA: the DR of a broadcast segment lists the routers joined by the segment. Type 2LSAs are flooded only in their own area. the link state id is the ip interface address of the DR.

LSA type 3

Summary Network LSA: originated from an ABR to advertise the subnets in an area to routers outside that area. When an ABR receives a Type1 or 2 LSA, it generates a Type 3 LSA for the networks learned from the Type 1 or 2 LSA to other areas. the link state id is the ip address of the subnet/s being advertised.

LSA type 4

Summary ASBR LSA: similar to type 3, however, it advertises a host route used to reach the ASBR. the link state id is the RID of the ASBR. not flooded to stub areas.

LSA TYPE 5

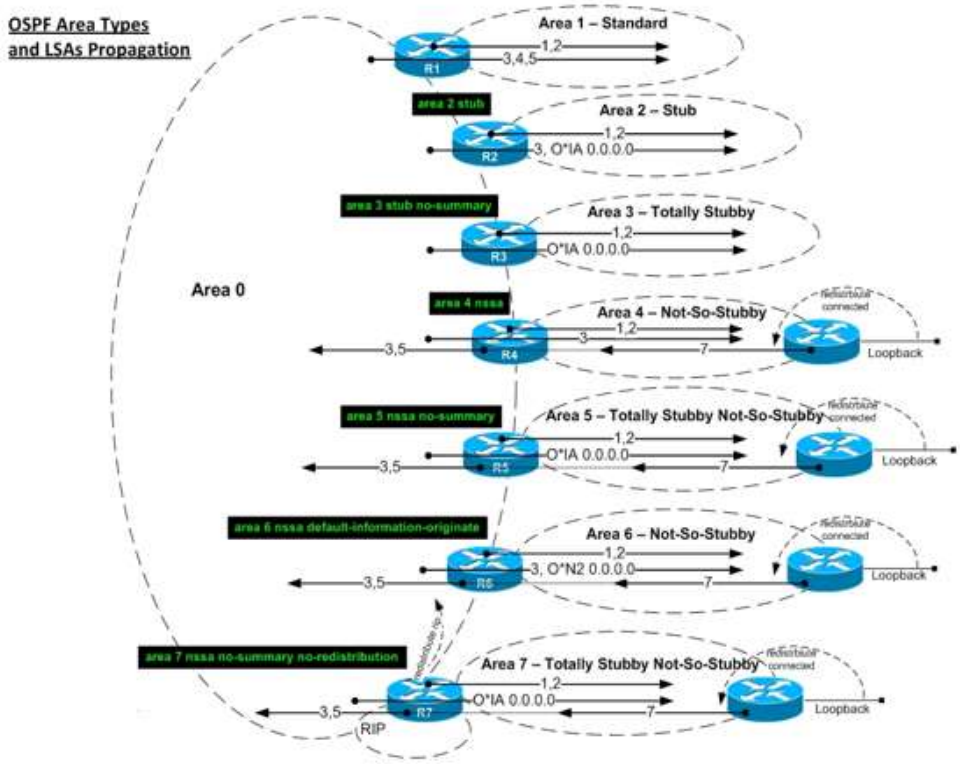
External: as the name implies, type 5 LSA's contain information about other routing processes being imported to OSPF. the link state id is the external network number. not flooded to stub areas.

LSA type 7

NSSA: creates a special type of link-state advertisement, which can only exist in an NSSA area. An NSSA ASBR generates this LSA and an NSSA ABR translates it into a type 5 LSA, which gets propagated into the OSPF domain.

LSA type 9

Link-local Opaque: for OSPFv2, and Intra-Area-Prefix LSA for OSPFv3. It is the OSPFv3 LSA that contains prefixes for stub and transit networks in the link-state ID.



2.6.a [ii] Route types [N1, N2, E1, E2]

Intra-Area (O)
Inter-Area (O IA)
External Type 1 (E1)
External Type 2 (E2)
NSSA Type 1 (N1)
NSSA Type 2 (N2)

- Type-1.....Router LSA ....Intra-Area.....................(O)
- Type-2.....Network LSA(Learned from a DR) ....Intra-Area...................(O)
- Type-3.....Summary LSA...Inter-Area....................(O IA)
- Type-5.....External LSA..External......................(E1 or E2)
- Type-7.....NSSA External LSA..NSSA External............(N1 or N2)

OSPF has enough information to select the best path in this order:

1. Intra-Area
2. Inter-Area
3. E1 or N1 (determined by metric - if equal E1 is chosen)
4. E2 or N2 (determined by forward metric - if equal E2 is chosen)

2.6.b Implement and troubleshoot neighbor relationship.

When OSPF adjacency is formed, a router goes through several state changes before it becomes fully adjacent with its neighbor. Those states are :

Down
Attempt
Init
2-Way
Exstart
Exchange
Loading
Full

Down

This is the first OSPF neighbor state. It means that no information (hellos) has been received from this neighbor, but hello packets can still be sent to the neighbor in this state.

During the fully adjacent neighbor state, if a router doesn't receive hello packet from a neighbor within the RouterDeadInterval time (RouterDeadInterval = 4*HelloInterval by default) or if the manually configured neighbor is being removed from the configuration, then the neighbor state changes from Full to Down.

Attempt

This state is only valid for manually configured neighbors in an NBMA environment. In Attempt state, the router sends unicast hello packets every poll interval to the neighbor, from which hellos have not been received within the dead interval.

Init

This state specifies that the router has received a hello packet from its neighbor, but the receiving router's ID was not included in the hello packet. When a router receives a hello packet from a neighbor, it should list the sender's router ID in its hello packet as an acknowledgment that it received a valid hello packet.

2-Way

This state designates that bi-directional communication has been established between two routers. Bi-directional means that each router has seen the other's hello packet. This state is attained when the router receiving the hello packet sees its own Router ID within the received hello packet's neighbor field. At this state, a router decides whether to become adjacent with this neighbor. On broadcast media and non-broadcast multiaccess networks, a router becomes full only with the designated router (DR) and the backup designated router (BDR); it stays in the 2-way state with all other neighbors. On Point-to-point and Point-to-multipoint networks, a router becomes full with all connected routers.

At the end of this stage, the DR and BDR for broadcast and non-broadcast multiacess networks are elected. For more information on the DR election process, refer to DR Election.

Note: Receiving a Database Descriptor (DBD) packet from a neighbor in the init state will also a cause a transition to 2-way state.

Exstart

Once the DR and BDR are elected, the actual process of exchanging link state information can start between the routers and their DR and BDR.

In this state, the routers and their DR and BDR establish a master-slave relationship and choose the initial sequence number for adjacency formation. The router with the higher router ID becomes the master and starts the exchange, and as such, is the only router that can increment the sequence number. Note that one would logically conclude that the DR/BDR with the highest router ID will become the master during this process of master-slave relation. Remember that the DR/BDR election might be purely by virtue of a higher priority configured on the router instead of highest router ID. Thus, it is possible that a DR plays the role of slave. And also note that master/slave election is on a per-neighbor basis.

Exchange

In the exchange state, OSPF routers exchange database descriptor (DBD) packets. Database descriptors contain link-state advertisement (LSA) headers only and describe the contents of the entire link-state database. Each DBD packet has a sequence number which can be incremented only by master which is explicitly acknowledged by slave. Routers also send link-state request packets and link-state update packets (which contain the entire LSA) in this state. The contents of the DBD received are compared to the information contained in the routers link-state database to check if new or more current link-state information is available with the neighbor.

Loading

In this state, the actual exchange of link state information occurs. Based on the information provided by the DBDs, routers send link-state request packets. The neighbor then provides the requested link-state information in link-state update packets. During the adjacency, if a router receives an outdated or missing LSA, it requests that LSA by sending a link-state request packet. All link-state update packets are acknowledged.

Full

In this state, routers are fully adjacent with each other. All the router and network LSAs are exchanged and the routers' databases are fully synchronized.

Full is the normal state for an OSPF router. If a router is stuck in another state, it is an indication that there are problems in forming adjacencies. The only exception to this is the 2-way state, which is normal in a broadcast network. Routers achieve the FULL state with their DR and BDR in NBMA/broadcast media and FULL state with every neighbor in the remaining media such as point-to-point and point-to-multipoint.

Note: The DR and BDR that achieve FULL state with every router on the segment will display FULL/DROTHER when you enter the show ip ospf neighbor command on either a DR or BDR. This simply means that the neighbor is not a DR or BDR, but since the router on which the command was entered is either a DR or BDR, this shows the neighbor as FULL/DROTHER.

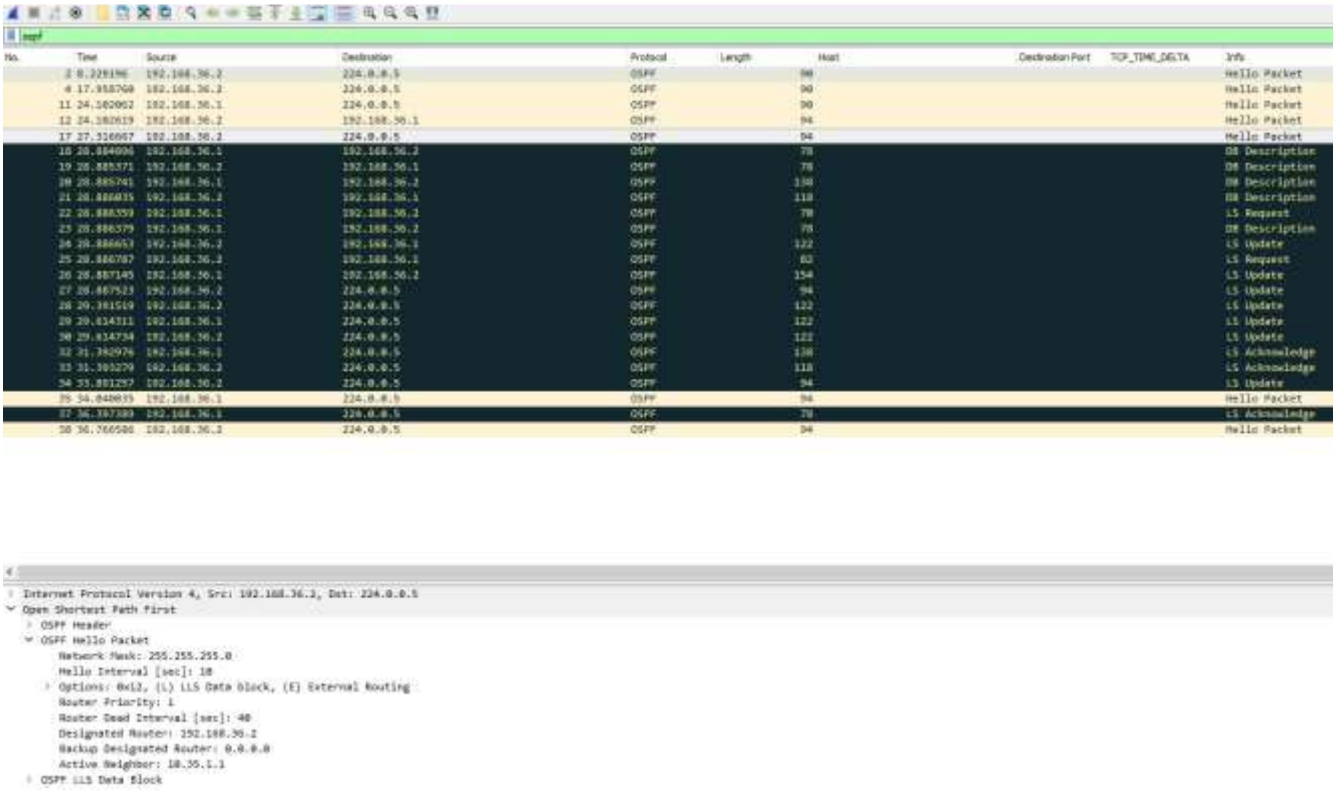
2.6.c Implement and troubleshoot OSPFv3 address-family support

2.6.c [i] IPv4 address-family

2.6.c [ii] IPv6 address-family

2.6.d Implement and troubleshoot network types, area types and router types

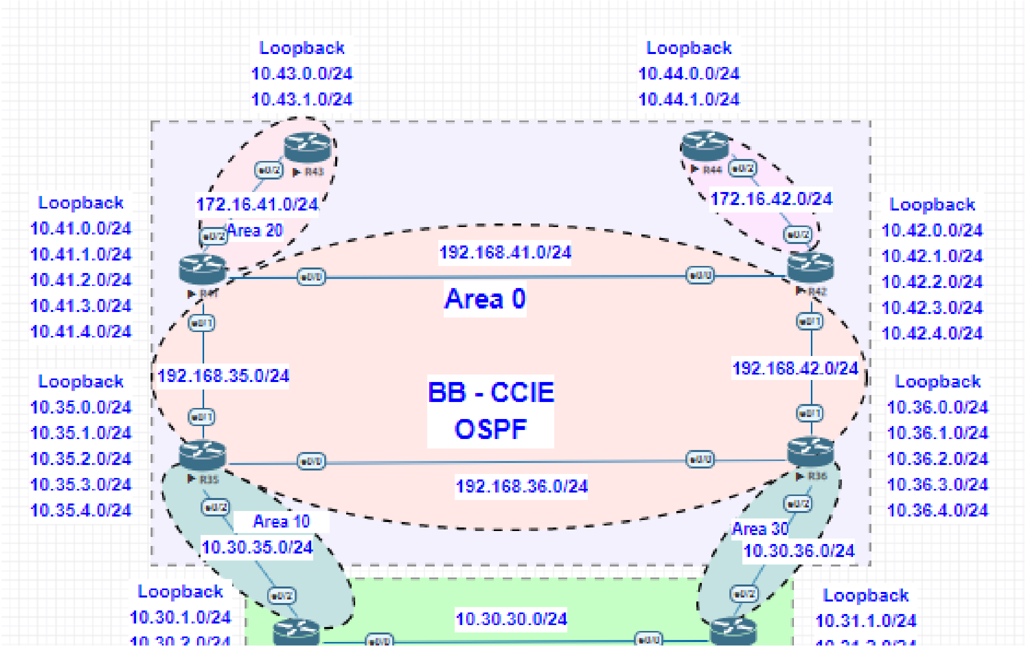
Wireshark capture for OSPF neighbour forming.



```
ospf
No.   Time       Source        Destination    Protocol  Length  Host  Destination Port  TCP_TIME_DELTA  Info
    2 0.229196   192.168.36.2  224.0.0.5      OSPF          90                                          Hello Packet
    4 17.958760  192.168.36.2  224.0.0.5      OSPF          90                                          Hello Packet
   11 24.182062  192.168.36.1  224.0.0.5      OSPF          90                                          Hello Packet
   12 24.182619  192.168.36.2  192.168.36.1   OSPF          94                                          Hello Packet
   17 27.316667  192.168.36.2  224.0.0.5      OSPF          94                                          Hello Packet
   18 28.884096  192.168.36.1  192.168.36.2   OSPF          78                                          DB Description
   19 28.885371  192.168.36.2  192.168.36.1   OSPF          78                                          DB Description
   20 28.885741  192.168.36.1  192.168.36.2   OSPF         138                                          DB Description
   21 28.886035  192.168.36.2  192.168.36.1   OSPF         118                                          DB Description
   22 28.886359  192.168.36.1  192.168.36.2   OSPF          78                                          LS Request
   23 28.886379  192.168.36.1  192.168.36.2   OSPF          78                                          DB Description
   24 28.886653  192.168.36.2  192.168.36.1   OSPF         122                                          LS Update
   25 28.886787  192.168.36.2  192.168.36.1   OSPF          82                                          LS Request
   26 28.887145  192.168.36.1  192.168.36.2   OSPF         154                                          LS Update
   27 28.887523  192.168.36.2  224.0.0.5      OSPF          94                                          LS Update
   28 29.381519  192.168.36.2  224.0.0.5      OSPF         122                                          LS Update
   29 29.614311  192.168.36.1  224.0.0.5      OSPF         122                                          LS Update
   30 29.614734  192.168.36.2  224.0.0.5      OSPF         122                                          LS Update
   32 31.382976  192.168.36.1  224.0.0.5      OSPF         138                                          LS Acknowledge
   33 31.383279  192.168.36.2  224.0.0.5      OSPF         118                                          LS Acknowledge
   34 33.881297  192.168.36.2  224.0.0.5      OSPF          94                                          LS Update
   35 34.040035  192.168.36.1  224.0.0.5      OSPF          94                                          Hello Packet
   37 36.397389  192.168.36.1  224.0.0.5      OSPF          78                                          LS Acknowledge
   38 36.766500  192.168.36.2  224.0.0.5      OSPF          94                                          Hello Packet
```

```
> Internet Protocol Version 4, Src: 192.168.36.2, Dst: 224.0.0.5
v Open Shortest Path First
  > OSPF Header
  v OSPF Hello Packet
      Network Mask: 255.255.255.0
      Hello Interval [sec]: 10
    > Options: 0x12, (L) LLS Data block, (E) External Routing
      Router Priority: 1
      Router Dead Interval [sec]: 40
      Designated Router: 192.168.36.2
      Backup Designated Router: 0.0.0.0
      Active Neighbor: 10.35.1.1
  > OSPF LLS Data Block
```

We use below topology for our testing:



2.6.d [i] Point-to-point, multipoint, broadcast, non-broadcast

BROADCAST:

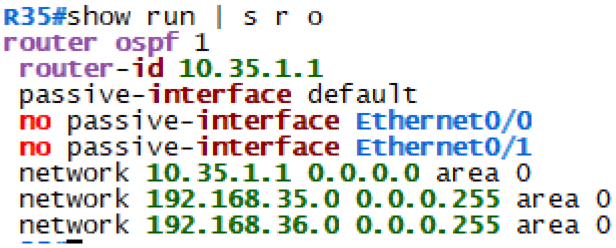
-Default on mulit-access broadcast medias
  - Ethernet, Token Ring, & FDDI
-Sends hellos and updates as multicast
-224.0.0.5 (ALL SPF Routers)
-224.0.0.6 (ALL DRouters)
Note: Both of these multicast addresses are going to use the OSPF transport protocol 89

Performs Designated Routerd (DR) and Backup Designated Router (BDR) Elections.
DR and BDR – are used for two things.
1 – It will cut down on the number of adjacencies and LSA flooding
2 – Is for the extra recursive step in the database for building the graph of the network in side and individual area.

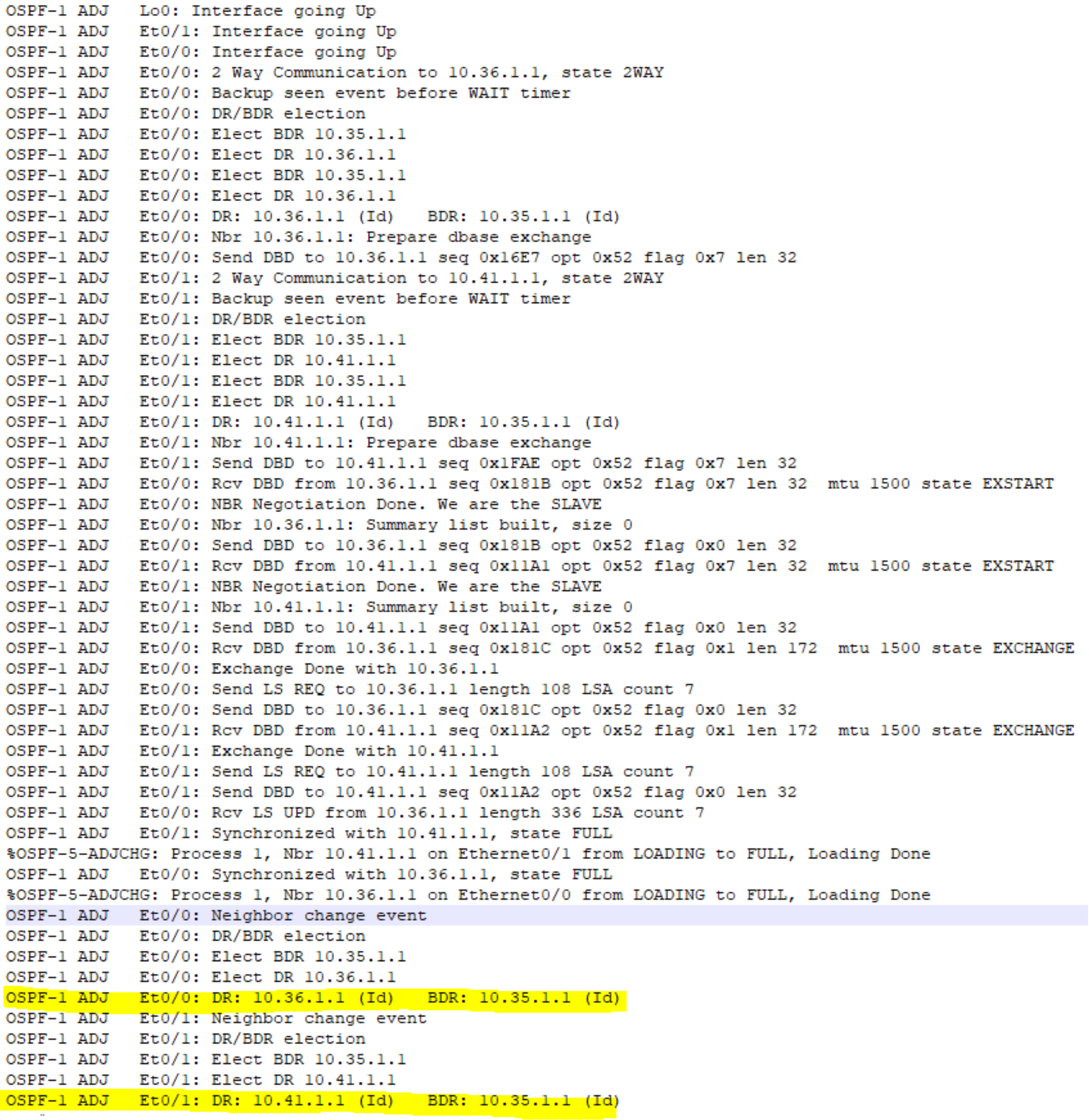
DR / BDR chosen through election process -Election bases on two fields in the Hello packet.

Routers priority,
from 0 – 255
Higher is better
0 = never. (Set ip ospf priority to 0 at the link level)
If there is a tie – then use

Router-ID
Highest loopback / Interface IP
Can be statically set (best practice to statically set)
Higher is better.
Now we configure R35 R36 R41 and R42 and verify the DR BDR

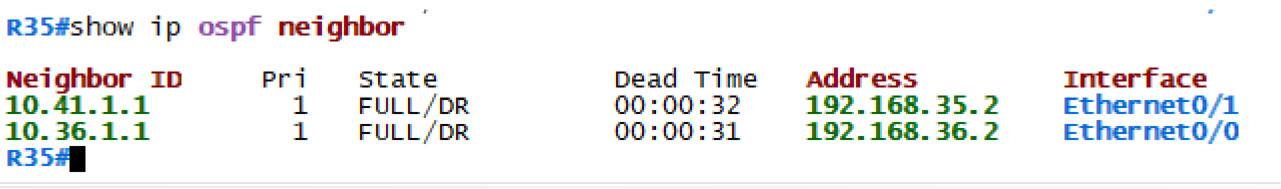


```
R35#show run | s r o
router ospf 1
 router-id 10.35.1.1
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 network 10.35.1.1 0.0.0.0 area 0
 network 192.168.35.0 0.0.0.255 area 0
 network 192.168.36.0 0.0.0.255 area 0
```



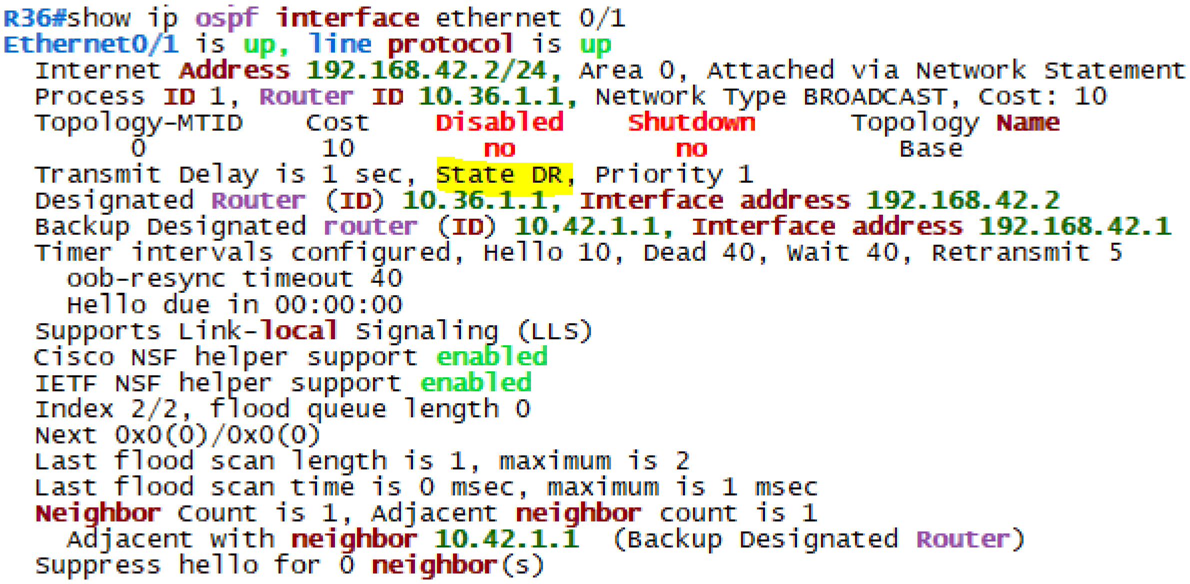
```
OSPF-1 ADJ   Lo0: Interface going Up
OSPF-1 ADJ   Et0/1: Interface going Up
OSPF-1 ADJ   Et0/0: Interface going Up
OSPF-1 ADJ   Et0/0: 2 Way Communication to 10.36.1.1, state 2WAY
OSPF-1 ADJ   Et0/0: Backup seen event before WAIT timer
OSPF-1 ADJ   Et0/0: DR/BDR election
OSPF-1 ADJ   Et0/0: Elect BDR 10.35.1.1
OSPF-1 ADJ   Et0/0: Elect DR 10.36.1.1
OSPF-1 ADJ   Et0/0: Elect BDR 10.35.1.1
OSPF-1 ADJ   Et0/0: Elect DR 10.36.1.1
OSPF-1 ADJ   Et0/0: DR: 10.36.1.1 (Id)   BDR: 10.35.1.1 (Id)
OSPF-1 ADJ   Et0/0: Nbr 10.36.1.1: Prepare dbase exchange
OSPF-1 ADJ   Et0/0: Send DBD to 10.36.1.1 seq 0x16E7 opt 0x52 flag 0x7 len 32
OSPF-1 ADJ   Et0/1: 2 Way Communication to 10.41.1.1, state 2WAY
OSPF-1 ADJ   Et0/1: Backup seen event before WAIT timer
OSPF-1 ADJ   Et0/1: DR/BDR election
OSPF-1 ADJ   Et0/1: Elect BDR 10.35.1.1
OSPF-1 ADJ   Et0/1: Elect DR 10.41.1.1
OSPF-1 ADJ   Et0/1: Elect BDR 10.35.1.1
OSPF-1 ADJ   Et0/1: Elect DR 10.41.1.1
OSPF-1 ADJ   Et0/1: DR: 10.41.1.1 (Id)   BDR: 10.35.1.1 (Id)
OSPF-1 ADJ   Et0/1: Nbr 10.41.1.1: Prepare dbase exchange
OSPF-1 ADJ   Et0/1: Send DBD to 10.41.1.1 seq 0x1FAE opt 0x52 flag 0x7 len 32
OSPF-1 ADJ   Et0/0: Rcv DBD from 10.36.1.1 seq 0x181B opt 0x52 flag 0x7 len 32  mtu 1500 state EXSTART
OSPF-1 ADJ   Et0/0: NBR Negotiation Done. We are the SLAVE
OSPF-1 ADJ   Et0/0: Nbr 10.36.1.1: Summary list built, size 0
OSPF-1 ADJ   Et0/0: Send DBD to 10.36.1.1 seq 0x181B opt 0x52 flag 0x0 len 32
OSPF-1 ADJ   Et0/1: Rcv DBD from 10.41.1.1 seq 0x11A1 opt 0x52 flag 0x7 len 32  mtu 1500 state EXSTART
OSPF-1 ADJ   Et0/1: NBR Negotiation Done. We are the SLAVE
OSPF-1 ADJ   Et0/1: Nbr 10.41.1.1: Summary list built, size 0
OSPF-1 ADJ   Et0/1: Send DBD to 10.41.1.1 seq 0x11A1 opt 0x52 flag 0x0 len 32
OSPF-1 ADJ   Et0/0: Rcv DBD from 10.36.1.1 seq 0x181C opt 0x52 flag 0x1 len 172  mtu 1500 state EXCHANGE
OSPF-1 ADJ   Et0/0: Exchange Done with 10.36.1.1
OSPF-1 ADJ   Et0/0: Send LS REQ to 10.36.1.1 length 108 LSA count 7
OSPF-1 ADJ   Et0/0: Send DBD to 10.36.1.1 seq 0x181C opt 0x52 flag 0x0 len 32
OSPF-1 ADJ   Et0/1: Rcv DBD from 10.41.1.1 seq 0x11A2 opt 0x52 flag 0x1 len 172  mtu 1500 state EXCHANGE
OSPF-1 ADJ   Et0/1: Exchange Done with 10.41.1.1
OSPF-1 ADJ   Et0/1: Send LS REQ to 10.41.1.1 length 108 LSA count 7
OSPF-1 ADJ   Et0/1: Send DBD to 10.41.1.1 seq 0x11A2 opt 0x52 flag 0x0 len 32
OSPF-1 ADJ   Et0/0: Rcv LS UPD from 10.36.1.1 length 336 LSA count 7
OSPF-1 ADJ   Et0/1: Synchronized with 10.41.1.1, state FULL
%OSPF-5-ADJCHG: Process 1, Nbr 10.41.1.1 on Ethernet0/1 from LOADING to FULL, Loading Done
OSPF-1 ADJ   Et0/0: Synchronized with 10.36.1.1, state FULL
%OSPF-5-ADJCHG: Process 1, Nbr 10.36.1.1 on Ethernet0/0 from LOADING to FULL, Loading Done
OSPF-1 ADJ   Et0/0: Neighbor change event
OSPF-1 ADJ   Et0/0: DR/BDR election
OSPF-1 ADJ   Et0/0: Elect BDR 10.35.1.1
OSPF-1 ADJ   Et0/0: Elect DR 10.36.1.1
OSPF-1 ADJ   Et0/0: DR: 10.36.1.1 (Id)   BDR: 10.35.1.1 (Id)
OSPF-1 ADJ   Et0/1: Neighbor change event
OSPF-1 ADJ   Et0/1: DR/BDR election
OSPF-1 ADJ   Et0/1: Elect BDR 10.35.1.1
OSPF-1 ADJ   Et0/1: Elect DR 10.41.1.1
OSPF-1 ADJ   Et0/1: DR: 10.41.1.1 (Id)   BDR: 10.35.1.1 (Id)
```

We can see R35 become BDR for R41 and R36



```
R35#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.41.1.1         1   FULL/DR         00:00:32    192.168.35.2    Ethernet0/1
10.36.1.1         1   FULL/DR         00:00:31    192.168.36.2    Ethernet0/0
R35#
```

Confirm from R36 as DR



```
R36#show ip ospf interface ethernet 0/1
Ethernet0/1 is up, line protocol is up
  Internet Address 192.168.42.2/24, Area 0, Attached via Network Statement
  Process ID 1, Router ID 10.36.1.1, Network Type BROADCAST, Cost: 10
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           10        no          no            Base
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 10.36.1.1, Interface address 192.168.42.2
  Backup Designated router (ID) 10.42.1.1, Interface address 192.168.42.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:00
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 2
  Last flood scan time is 0 msec, maximum is 1 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.42.1.1  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
```

NON-BROADCAST:

Default on multipoint and NBMA medias.
--Frame Relay & ATM – tyoe non-broadcast.

Sends hellos as unicast
-manually defines addresses with neighbour command
-performs DR and BDR elections

This means that we need to make sure the hubs are the DR's and we need to configure under the process the manually defined address with the "neighbour" command.

To specify the DR election we basically just tell the spokes not to become the DR or BDR. By setting the priority to 0.
Again then who is elected the DR is who's process loads first.

Ex:
R(config)#interface serial 0/0
R(config-if)#ip ospf priority 0 ?

We would then need to manually configure the neighbours under the process
Note: the Neighbour commands are only required on the DR and BDR. Once the spokes hello's come in, they will respond to originated them.

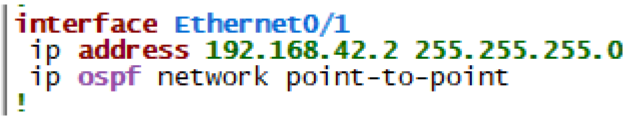
R(config)#router ospf 1
R(config-router)#neighbor x.x.x.x

Note: The default behaviour of the network type broadcast or non-broadcast – the DR is not updating the next-hop value for any link state updates coming from the DROthers.

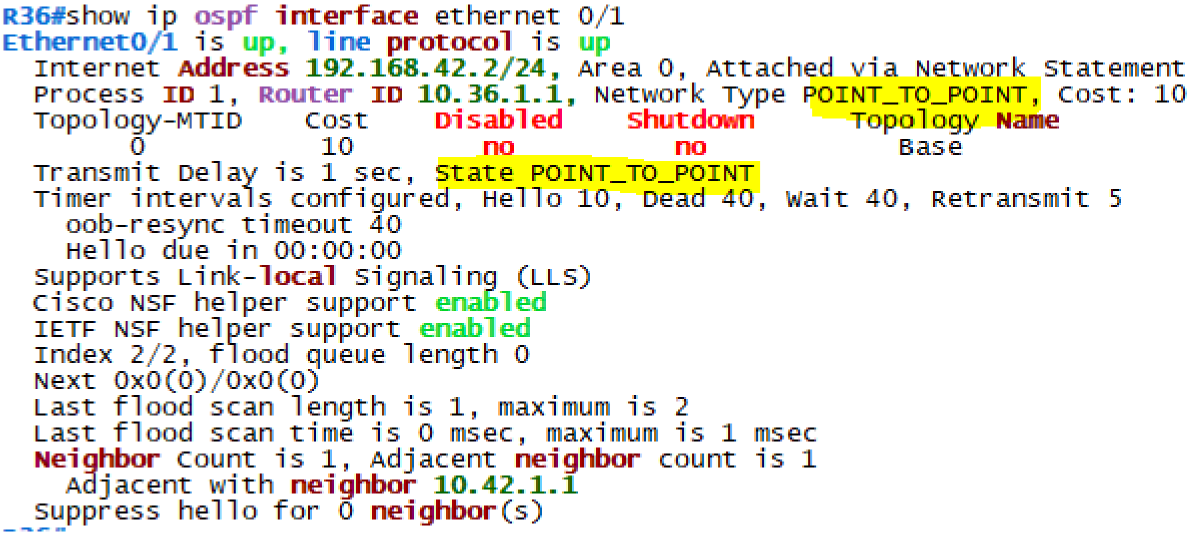
OSPF point-to-point:

- Automatic neighbor discovery so no need to configure OSPF neighbors yourself.
- No DR/BDR election since OSPF sees the network as a collection of point-to-point links.
- Normally uses for point-to-point sub-interfaces with an IP subnet per link.
- Can also be used with multiple PVCs using only one subnet.

Now we configure R36 and R41 Point to Point interface. And verify the output.

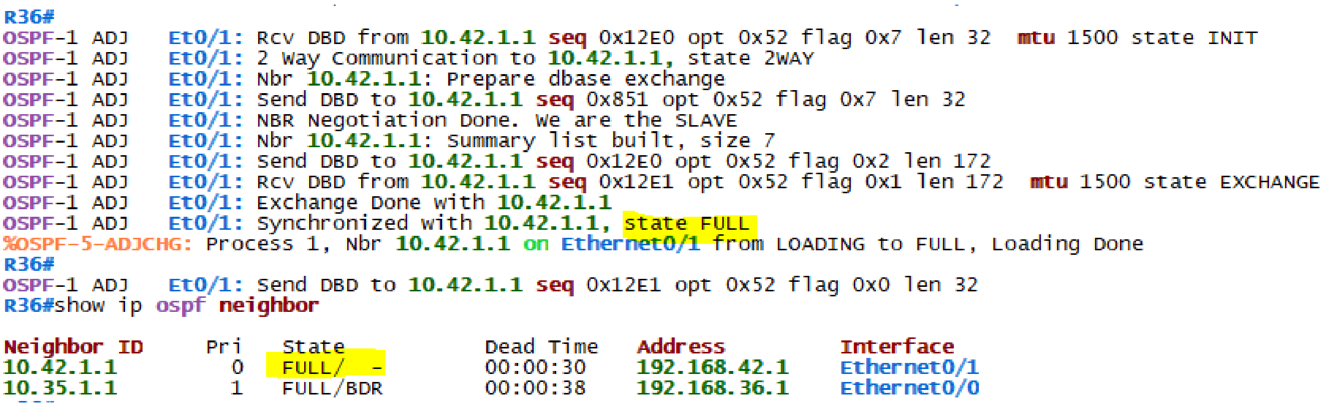


```
interface Ethernet0/1
 ip address 192.168.42.2 255.255.255.0
 ip ospf network point-to-point
!
```



```
R36#show ip ospf interface ethernet 0/1
Ethernet0/1 is up, line protocol is up
  Internet Address 192.168.42.2/24, Area 0, Attached via Network Statement
  Process ID 1, Router ID 10.36.1.1, Network Type POINT_TO_POINT, Cost: 10
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           10        no          no            Base
  Transmit Delay is 1 sec, State POINT_TO_POINT
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:00
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 2
  Last flood scan time is 0 msec, maximum is 1 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.42.1.1
  Suppress hello for 0 neighbor(s)
```

Here is the debug of election process and neighours.

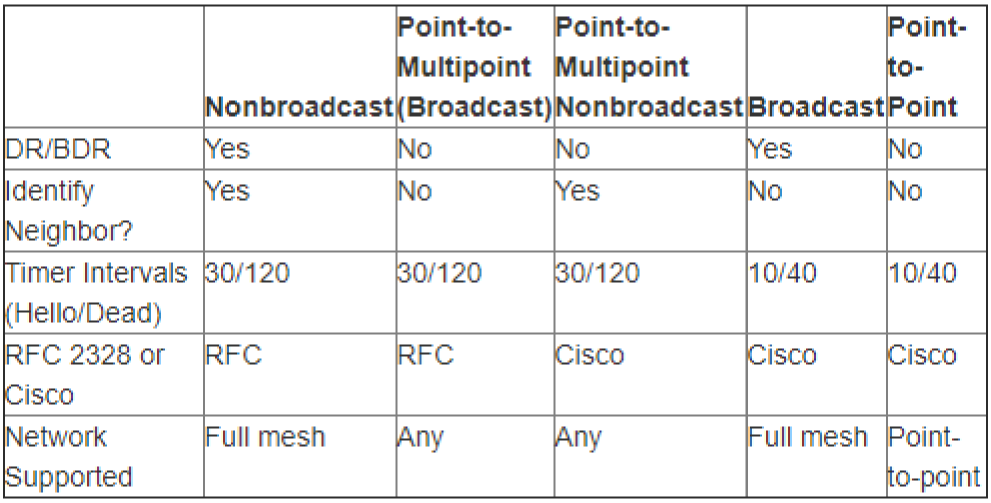


```
R36#
OSPF-1 ADJ   Et0/1: Rcv DBD from 10.42.1.1 seq 0x12E0 opt 0x52 flag 0x7 len 32  mtu 1500 state INIT
OSPF-1 ADJ   Et0/1: 2 Way Communication to 10.42.1.1, state 2WAY
OSPF-1 ADJ   Et0/1: Nbr 10.42.1.1: Prepare dbase exchange
OSPF-1 ADJ   Et0/1: Send DBD to 10.42.1.1 seq 0x851 opt 0x52 flag 0x7 len 32
OSPF-1 ADJ   Et0/1: NBR Negotiation Done. We are the SLAVE
OSPF-1 ADJ   Et0/1: Nbr 10.42.1.1: Summary list built, size 7
OSPF-1 ADJ   Et0/1: Send DBD to 10.42.1.1 seq 0x12E0 opt 0x52 flag 0x2 len 172
OSPF-1 ADJ   Et0/1: Rcv DBD from 10.42.1.1 seq 0x12E1 opt 0x52 flag 0x1 len 172  mtu 1500 state EXCHANGE
OSPF-1 ADJ   Et0/1: Exchange Done with 10.42.1.1
OSPF-1 ADJ   Et0/1: Synchronized with 10.42.1.1, state FULL
%OSPF-5-ADJCHG: Process 1, Nbr 10.42.1.1 on Ethernet0/1 from LOADING to FULL, Loading Done
R36#
OSPF-1 ADJ   Et0/1: Send DBD to 10.42.1.1 seq 0x12E1 opt 0x52 flag 0x0 len 32
R36#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.42.1.1         0   FULL/  -        00:00:30    192.168.42.1    Ethernet0/1
10.35.1.1         1   FULL/BDR        00:00:38    192.168.36.1    Ethernet0/0
```

OSPF Network Type Point-to-Multipoint: (this will be done testing DMVPN)

#ip ospf network point-to-multipoint
-treats network as a an underlying collection of point-to-point links
-Sends hellos as multicast 224.0.0.5
- Special next hop processing
- Usually the best design option for partial mesh NBMA Networks

Point-to-Multipoint - You always chose you closest layer 2 neighbour on the circuit for any routing information that is received. The Next Hop values WILL BE CHANGED.



	Nonbroadcast	**Point-to-Multipoint (Broadcast)**	**Point-to-Multipoint Nonbroadcast**	**Broadcast**	**Point-to-Point**
DR/BDR	Yes	No	No	Yes	No
Identify Neighbor?	Yes	No	Yes	No	No
Timer Intervals (Hello/Dead)	30/120	30/120	30/120	10/40	10/40
RFC 2328 or Cisco	RFC	RFC	Cisco	Cisco	Cisco
Network Supported	Full mesh	Any	Any	Full mesh	Point-to-point

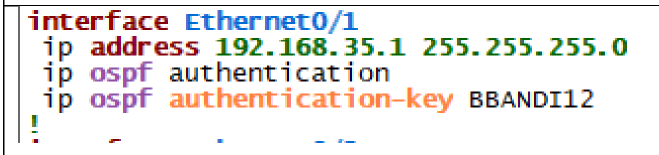
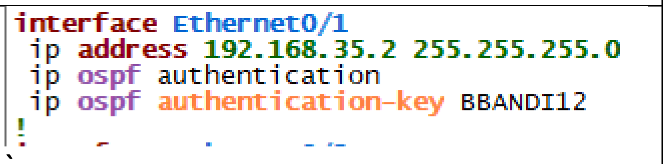
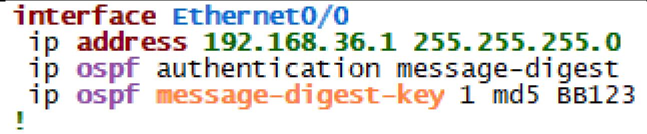
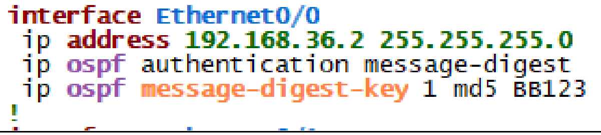
OSPF Authentication:

OSPF supports 3 types of authentication
-0 = Null (Default)
-1 = Clear Text
-2 = MD5

Can be enabled
-On all links in the area
-On a per link basis

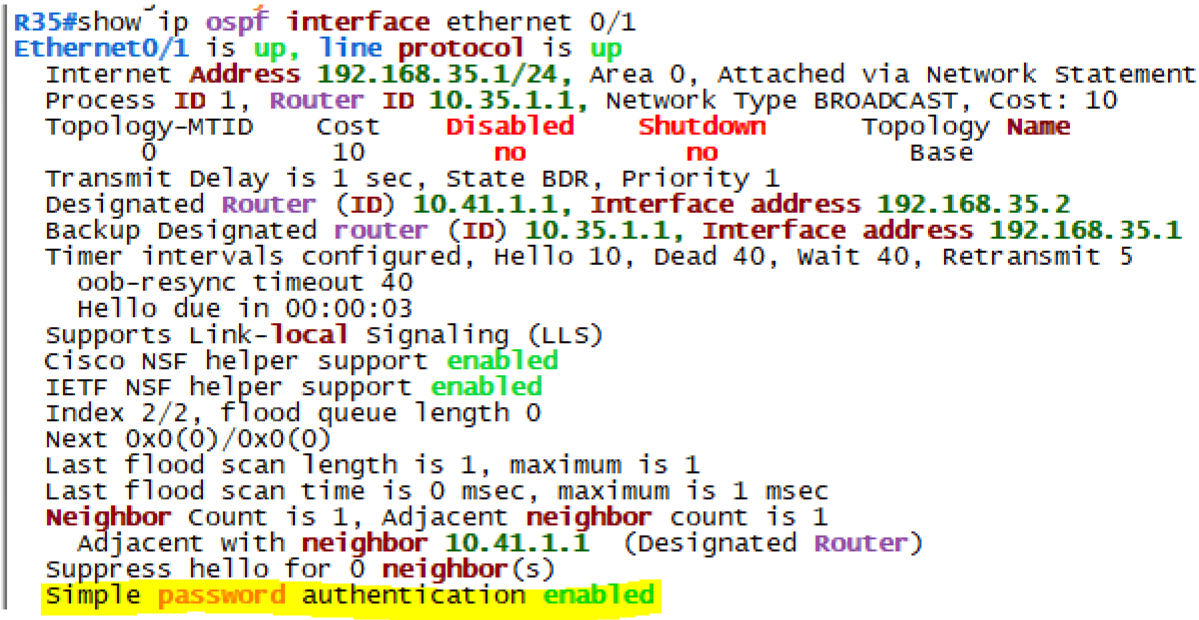
Key – is always applied at the link level
-Virtual-Links are Area0 interfaces

Now we configured R35 and R41 connected interface Clear Text authentication.
R35 and R36 MD5 authentication and verify

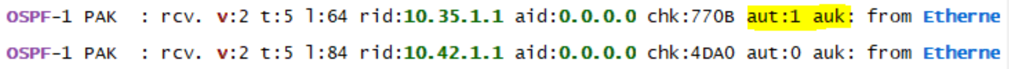
R35	**R41**
 interface Ethernet0/1 ip address 192.168.35.1 255.255.255.0 ip ospf authentication ip ospf authentication-key BBANDI12 !	 interface Ethernet0/1 ip address 192.168.35.2 255.255.255.0 ip ospf authentication ip ospf authentication-key BBANDI12 !
R35	R36
 interface Ethernet0/0 ip address 192.168.36.1 255.255.255.0 ip ospf authentication message-digest ip ospf message-digest-key 1 md5 BB123 !	 interface Ethernet0/0 ip address 192.168.36.2 255.255.255.0 ip ospf authentication message-digest ip ospf message-digest-key 1 md5 BB123 !

Check the Debug, if it shows Auth 1 means clear text, Auth 2 means MD5

Clear Text Authentication debug

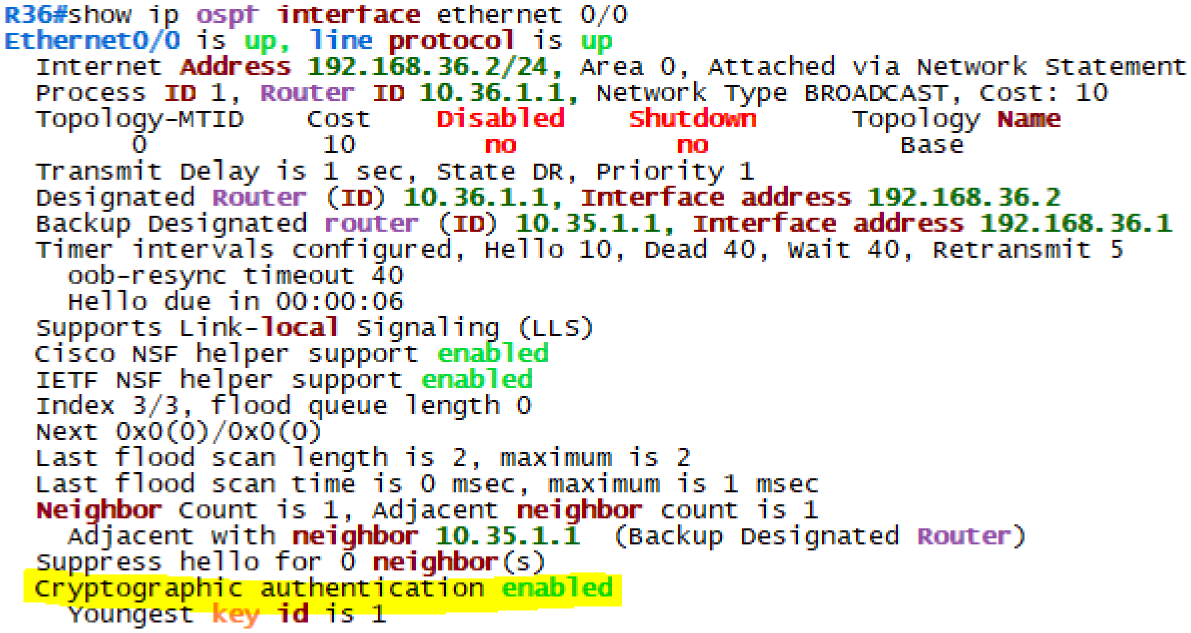


```
R35#show ip ospf interface ethernet 0/1
Ethernet0/1 is up, line protocol is up
  Internet Address 192.168.35.1/24, Area 0, Attached via Network Statement
  Process ID 1, Router ID 10.35.1.1, Network Type BROADCAST, Cost: 10
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           10        no          no            Base
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 10.41.1.1, Interface address 192.168.35.2
  Backup Designated router (ID) 10.35.1.1, Interface address 192.168.35.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:03
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 1 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.41.1.1  (Designated Router)
  Suppress hello for 0 neighbor(s)
  Simple password authentication enabled
```

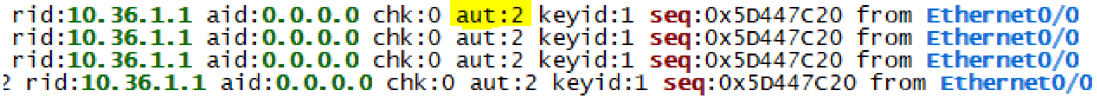


```
OSPF-1 PAK  : rcv. v:2 t:5 l:64 rid:10.35.1.1 aid:0.0.0.0 chk:770B aut:1 auk: from Etherne
OSPF-1 PAK  : rcv. v:2 t:5 l:84 rid:10.42.1.1 aid:0.0.0.0 chk:4DA0 aut:0 auk: from Etherne
```

Md5 Authentication:



```
R36#show ip ospf interface ethernet 0/0
Ethernet0/0 is up, line protocol is up
  Internet Address 192.168.36.2/24, Area 0, Attached via Network Statement
  Process ID 1, Router ID 10.36.1.1, Network Type BROADCAST, Cost: 10
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           10        no          no            Base
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 10.36.1.1, Interface address 192.168.36.2
  Backup Designated router (ID) 10.35.1.1, Interface address 192.168.36.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:06
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 3/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 2, maximum is 2
  Last flood scan time is 0 msec, maximum is 1 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.35.1.1  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
  Cryptographic authentication enabled
    Youngest key id is 1
```

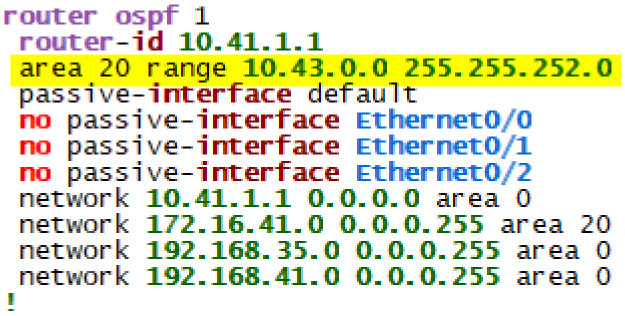


```
 rid:10.36.1.1 aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x5D447C20 from Ethernet0/0
 rid:10.36.1.1 aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x5D447C20 from Ethernet0/0
 rid:10.36.1.1 aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x5D447C20 from Ethernet0/0
? rid:10.36.1.1 aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x5D447C20 from Ethernet0/0
```

Internal Summarisation:

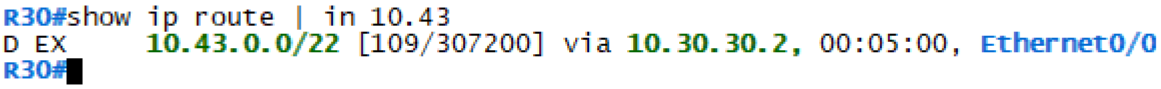
OSPF area require the same copy of the database to compute correct SPF, filtering or summarization of routes in the database can only occur between areas or domains, not within an area. The below configuration illustrates how an Intra-Area Summary Network LSA (LSA 3) can be summarized as it is originated by an ABR.

From R43 area 20 we will summarise to Area 0



```
router ospf 1
 router-id 10.41.1.1
 area 20 range 10.43.0.0 255.255.252.0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/2
 network 10.41.1.1 0.0.0.0 area 0
 network 172.16.41.0 0.0.0.255 area 20
 network 192.168.35.0 0.0.0.255 area 0
 network 192.168.41.0 0.0.0.255 area 0
!
```

Verify on other routers for the same.

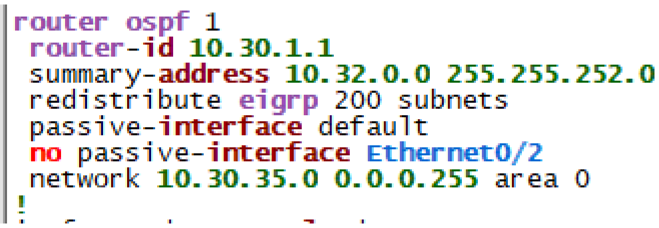


```
R30#show ip route | in 10.43
D EX     10.43.0.0/22 [109/307200] via 10.30.30.2, 00:05:00, Ethernet0/0
R30#
```

OSPF External Summarization:

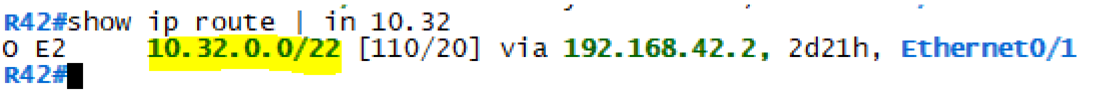
External OSPF summarization is configured at the redistribution point between routing domains with the **summary-address** command. These summaries inherit their attributes from the subnets that make them up. For example, a summary comprised of External Type-1 routes will result in an External Type-1 summary.
The metric-type 1 command is set at the time of redistribution instead of on the summary itself.
External Type-2 OSPF routes, which are the default, do not install the end-to-end metric in the routing table. Instead, only the metric that was reported via the ASBR is installed. The actual routing path is
determined by the addition of the reported metric and the metric toward the ASBR, which is called the forward metric

On our topology R30 and R31 are ASBR we summarise the address and check on R42 for verification.

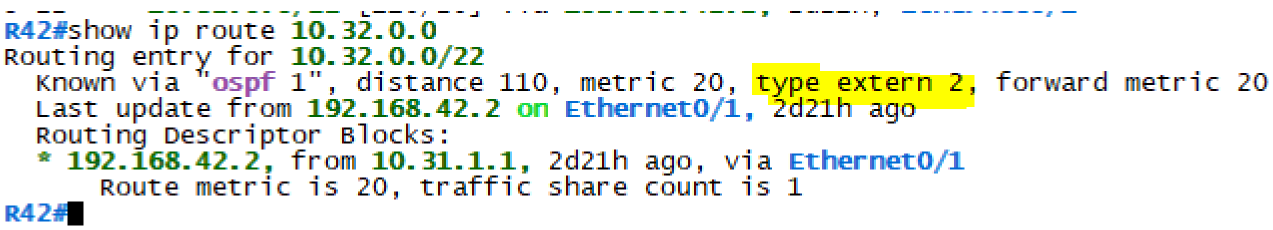


```
router ospf 1
 router-id 10.30.1.1
 summary-address 10.32.0.0 255.255.252.0
 redistribute eigrp 200 subnets
 passive-interface default
 no passive-interface Ethernet0/2
 network 10.30.35.0 0.0.0.255 area 0
!
```

Verification



```
R42#show ip route | in 10.32
O E2     10.32.0.0/22 [110/20] via 192.168.42.2, 2d21h, Ethernet0/1
R42#
```



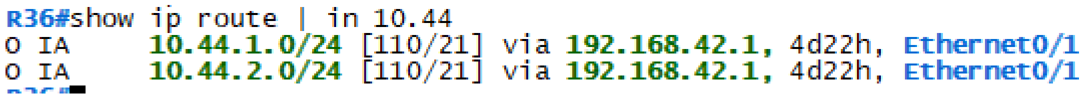
```
R42#show ip route 10.32.0.0
Routing entry for 10.32.0.0/22
  Known via "ospf 1", distance 110, metric 20, type extern 2, forward metric 20
  Last update from 192.168.42.2 on Ethernet0/1, 2d21h ago
  Routing Descriptor Blocks:
  * 192.168.42.2, from 10.31.1.1, 2d21h ago, via Ethernet0/1
      Route metric is 20, traffic share count is 1
R42#
```

Inter-area Filtering:

OSPF uses LSA type 3 for inter-area prefixes and if you want, you can filter these between OSPF areas. Since you can only filter between areas you'll have to configure this on the ABR. Filtering is possible **inbound** or **outbound** an area by using the area **filter-list** command.

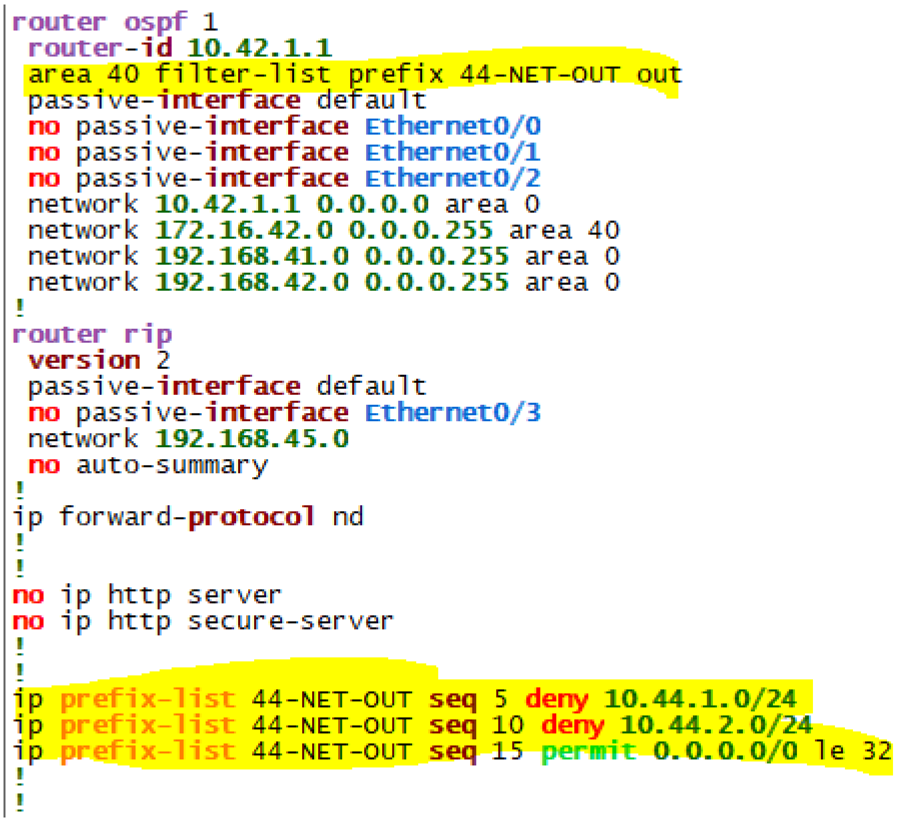
R43 have area 40, so let filter 10.44.0.0/23 network going out to Area 0

Before doing let's check the routing on R36 about 10.40.x.x network



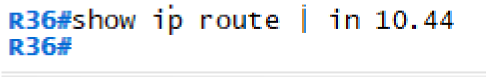
```
R36#show ip route | in 10.44
O IA     10.44.1.0/24 [110/21] via 192.168.42.1, 4d22h, Ethernet0/1
O IA     10.44.2.0/24 [110/21] via 192.168.42.1, 4d22h, Ethernet0/1
```

Now we create a prefix list and apply to ospf.



```
router ospf 1
 router-id 10.42.1.1
 area 40 filter-list prefix 44-NET-OUT out
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/2
 network 10.42.1.1 0.0.0.0 area 0
 network 172.16.42.0 0.0.0.255 area 40
 network 192.168.41.0 0.0.0.255 area 0
 network 192.168.42.0 0.0.0.255 area 0
!
router rip
 version 2
 passive-interface default
 no passive-interface Ethernet0/3
 network 192.168.45.0
 no auto-summary
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
ip prefix-list 44-NET-OUT seq 5 deny 10.44.1.0/24
ip prefix-list 44-NET-OUT seq 10 deny 10.44.2.0/24
ip prefix-list 44-NET-OUT seq 15 permit 0.0.0.0/0 le 32
!
!
```

Check now on R36 (none were found to display)



```
R36#show ip route | in 10.44
R36#
```

2.6.d [ii] LSA types, area type: backbone, normal, transit, stub, NSSA, totally stub

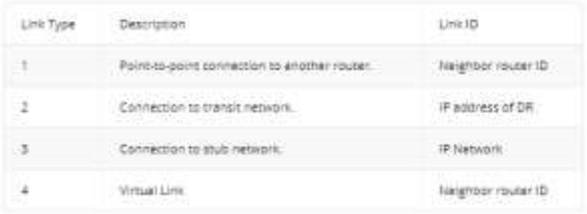
LSA Types: OSPF uses a LSDB (link state database) and fills this with LSAs (link state advertisement). Instead of using 1 LSA packet OSPF has many different types of LSAs:

https://blog.ine.com/2009/08/17/ospf-route-filtering-demystified

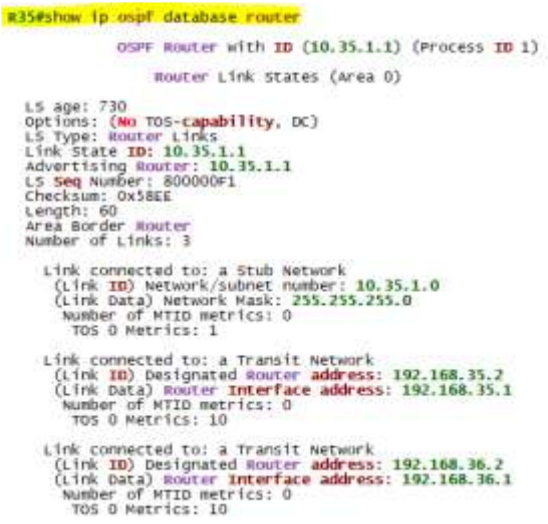
LSA Type 1: Router LSA

Each router within the area will flood a type 1 router LSA within the area. In this LSA you will find a list with all the directly connected links of this router. How do we identify a link?

- The IP prefix on an interface.
- The link type. There are 4 different link types:



Link Type	Description	Link ID
1	Point-to-point connection to another router.	Neighbor router ID
2	Connection to transit network.	IP address of DR
3	Connection to stub network.	IP Network
4	Virtual Link	Neighbor router ID



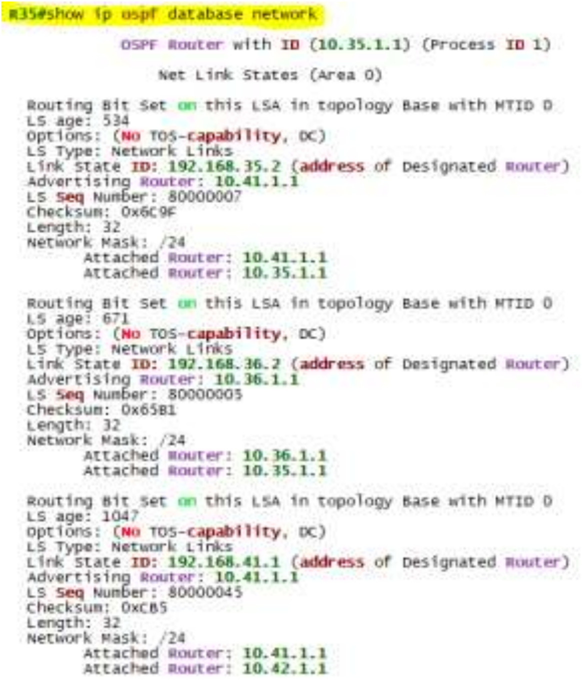
```
R35#show ip ospf database router

            OSPF Router with ID (10.35.1.1) (Process ID 1)

                Router Link States (Area 0)

  LS age: 730
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 10.35.1.1
  Advertising Router: 10.35.1.1
  LS Seq Number: 800000F1
  Checksum: 0x58EE
  Length: 60
  Area Border Router
  Number of Links: 3

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 10.35.1.0
     (Link Data) Network Mask: 255.255.255.0
      Number of MTID metrics: 0
       TOS 0 Metrics: 1

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.168.35.2
     (Link Data) Router Interface address: 192.168.35.1
      Number of MTID metrics: 0
       TOS 0 Metrics: 10

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.168.36.2
     (Link Data) Router Interface address: 192.168.36.1
      Number of MTID metrics: 0
       TOS 0 Metrics: 10
```

LSA Type 2: Network LSA

Validation LSA for the network connected.
In this LSA we will find all the routers that are connected to the multi-access network, the DR and of course the prefix and subnet mask.

For the Point to Point both the neighbour need to validated sending extra LSA Type 1

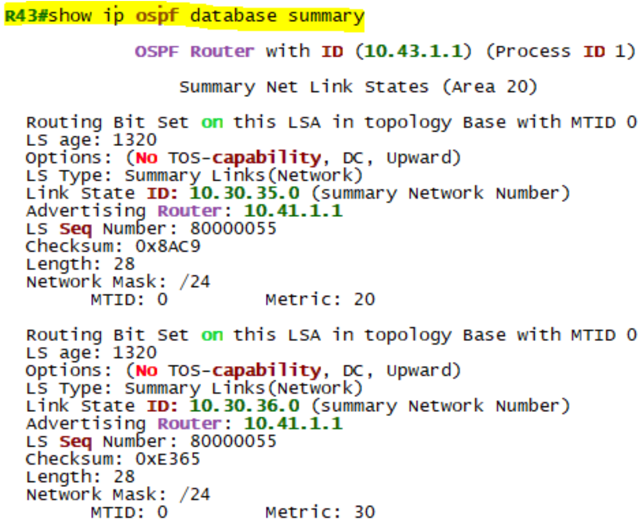
Type1 and Type 2 information only can be seen in the same Area, other area information will not show.



```
R35#show ip ospf database network

            OSPF Router with ID (10.35.1.1) (Process ID 1)

                Net Link States (Area 0)

  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 534
  Options: (No TOS-capability, DC)
  LS Type: Network Links
  Link State ID: 192.168.35.2 (address of Designated Router)
  Advertising Router: 10.41.1.1
  LS Seq Number: 80000007
  Checksum: 0x6C9F
  Length: 32
  Network Mask: /24
        Attached Router: 10.41.1.1
        Attached Router: 10.35.1.1

  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 671
  Options: (No TOS-capability, DC)
  LS Type: Network Links
  Link State ID: 192.168.36.2 (address of Designated Router)
  Advertising Router: 10.36.1.1
  LS Seq Number: 80000005
  Checksum: 0x65B1
  Length: 32
  Network Mask: /24
        Attached Router: 10.36.1.1
        Attached Router: 10.35.1.1

  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 1047
  Options: (No TOS-capability, DC)
  LS Type: Network Links
  Link State ID: 192.168.41.1 (address of Designated Router)
  Advertising Router: 10.41.1.1
  LS Seq Number: 80000045
  Checksum: 0xCB5
  Length: 32
  Network Mask: /24
        Attached Router: 10.41.1.1
        Attached Router: 10.42.1.1
```

LSA Type 3: Summary LSA

Summary LSA done by ABR
The route will be appeared as O IA routes in the routing table.
This is not Summarisation, this LSA Summary.
You cannot do Summarisation area 0 (or with in the area, since OSPF looking to see entire topology).
Summary possible to do on ABR.



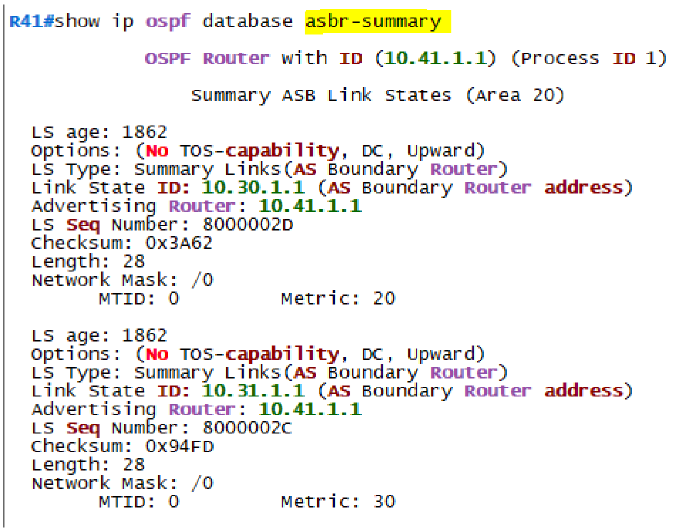
```
R43#show ip ospf database summary

            OSPF Router with ID (10.43.1.1) (Process ID 1)

                Summary Net Link States (Area 20)

  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 1320
  Options: (No TOS-capability, DC, Upward)
  LS Type: Summary Links(Network)
  Link State ID: 10.30.35.0 (summary Network Number)
  Advertising Router: 10.41.1.1
  LS Seq Number: 80000055
  Checksum: 0x8AC9
  Length: 28
  Network Mask: /24
        MTID: 0         Metric: 20

  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 1320
  Options: (No TOS-capability, DC, Upward)
  LS Type: Summary Links(Network)
  Link State ID: 10.30.36.0 (summary Network Number)
  Advertising Router: 10.41.1.1
  LS Seq Number: 80000055
  Checksum: 0xE365
  Length: 28
  Network Mask: /24
        MTID: 0         Metric: 30
```

LSA Type 4: Summary ASBR LSA

It is generated by an ABR. It is flooded from area 0 into a non-transit area and vice versa. It represents the ABR's reachability to ASBRs in other areas where in it includes cost but hides the ABR's actual path to the destination.

This is needed because Type 5 External LSAs are flooded to all areas and the detailed next-hop information may not be available in those other areas. This is solved by an Area Border Router flooding the information for the router (i.e. the Autonomous System Boundary Router) where the type 5 originated. The link-state ID is the router ID of the described ASBR for type 4 LSAs.

Example to see type 4 LSA :

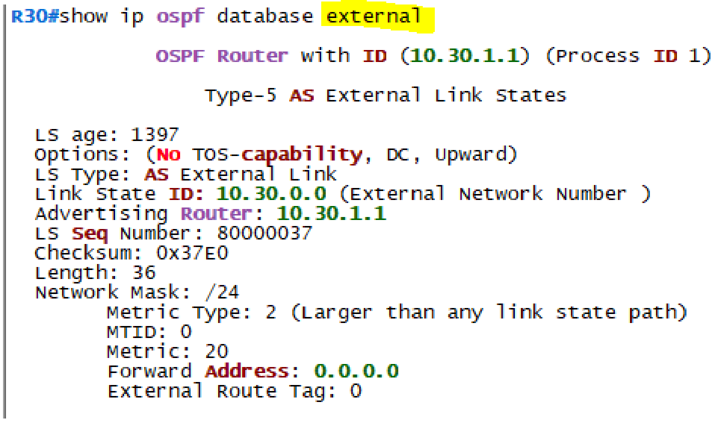


```
R41#show ip ospf database asbr-summary

            OSPF Router with ID (10.41.1.1) (Process ID 1)

                Summary ASB Link States (Area 20)

  LS age: 1862
  Options: (No TOS-capability, DC, Upward)
  LS Type: Summary Links(AS Boundary Router)
  Link State ID: 10.30.1.1 (AS Boundary Router address)
  Advertising Router: 10.41.1.1
  LS Seq Number: 8000002D
  Checksum: 0x3A62
  Length: 28
  Network Mask: /0
        MTID: 0         Metric: 20

  LS age: 1862
  Options: (No TOS-capability, DC, Upward)
  LS Type: Summary Links(AS Boundary Router)
  Link State ID: 10.31.1.1 (AS Boundary Router address)
  Advertising Router: 10.41.1.1
  LS Seq Number: 8000002C
  Checksum: 0x94FD
  Length: 28
  Network Mask: /0
        MTID: 0         Metric: 30
```

LSA Type 5: Autonomous system external LSA

The external LSA defines routes to destinations external to the autonomous system. Domain-wide, the default route can also be injected as an external route. External LSAs are flooded throughout the OSPF domain, except to stubby areas. To install an external LSA in the routing table, two essential things must take place:

- The calculating router must see the ASBR through the intra-area or inter area route. This means that it should have either a router LSA for the ASBR or a Type 4 LSA for the ASBR, in case of multiple areas.
- The forwarding address must be known through an intra- or inter area route.



```
R30#show ip ospf database external

            OSPF Router with ID (10.30.1.1) (Process ID 1)

                Type-5 AS External Link States

  LS age: 1397
  Options: (No TOS-capability, DC, Upward)
  LS Type: AS External Link
  Link State ID: 10.30.0.0 (External Network Number )
  Advertising Router: 10.30.1.1
  LS Seq Number: 80000037
  Checksum: 0x37E0
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        MTID: 0
        Metric: 20
        Forward Address: 0.0.0.0
        External Route Tag: 0
```

LSA Type 6: Multicast OSPF LSA

LSA Type 7: Not-so-stubby area LSA

LSA Type 8: External attribute LSA for BGP

Area type: backbone, normal, transit, stub, NSSA, totally stub

BACKBONE:

OSPF has special restrictions when multiple areas are involved. If more than one area is configured, one of these areas has be to be area 0. This is called the backbone.

Stub, NSSA, totally stub:

OSPF has special area types called **stub areas**. special area types:

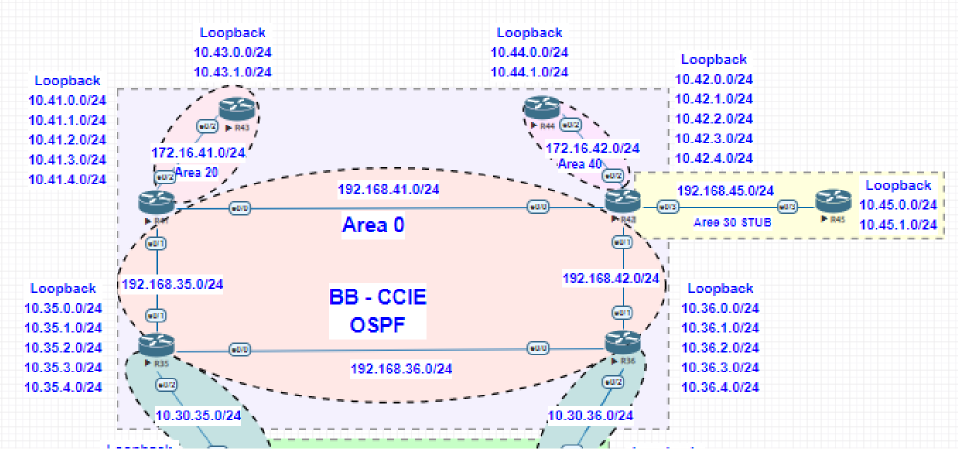
- **Stub area**
  A stub area is an area in which you do not allow advertisements of external routes, which thus reduces the size of the database even more. Instead, a default summary route (0.0.0.0) is inserted into the stub area in order to reach these external routes. If you have no external routes in your network, then you have no need to define stub areas.

- **Totally stub area**
  Stub areas are shielded from external routes but receive information about networks that belong to other areas of the same OSPF domain. You can define totally stubby areas. Routers in totally stubby areas keep their LSDB-only information about routing within their area, plus the default route.

- **NSSA (not so stubby area)**
  Not-so-stubby areas (NSSAs) are an extension of OSPF stub areas. Like stub areas, they prevent the flooding of AS-external link-state advertisements (LSAs) into NSSAs and instead rely on default routing to external destinations. As a result, NSSAs (like stub areas) must be placed at the edge of an OSPF routing domain. NSSAs are more flexible than stub areas in that an NSSA can import external routes into the OSPF routing domain and thereby provide transit service to small routing domains that are not part of the OSPF routing domain.
- **Totally NSSA (totally not so stubby area)**
  implement stub or totally stubby functionality yet contain an ASBR. Type 7 LSAs generated by the ASBR are converted to type 5 by ABRs to be flooded to the rest of the OSPF domain.

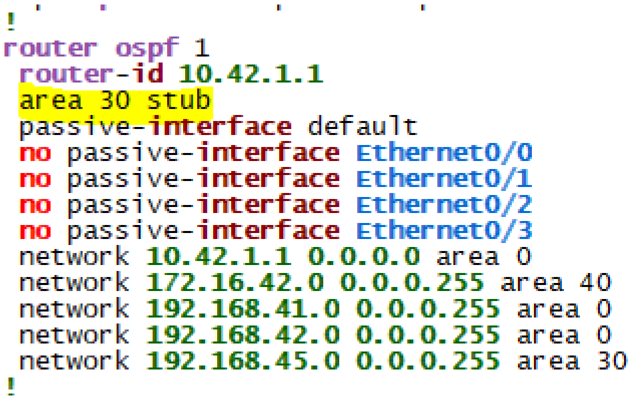
Standard areas	can contain LSAs of type 1, 2, 3, 4, and 5, and may contain an ASBR. The backbone is considered a standard area.
Stub areas	can contain type 1, 2, and 3 LSAs. A default route is substituted for external routes.
Totally stubby areas	can only contain type 1 and 2 LSAs, and a single type 3 LSA. The type 3 LSA describes a default route, substituted for all external and inter-area routes.
Not-so-stubby areas	implement stub or totally stubby functionality yet contain an ASBR. Type 7 LSAs generated by the ASBR are converted to type 5 by ABRs to be flooded to the rest of the OSPF domain

Stub Area:

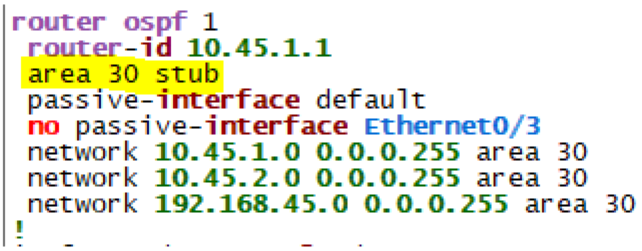


In the network topology we use R45 as Stub area of 30

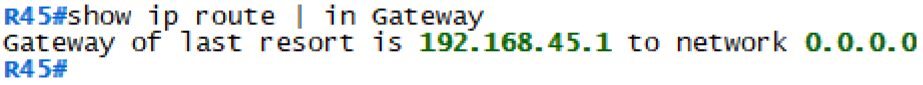
Here is the config on R42 and R45 and check the verification for external routes



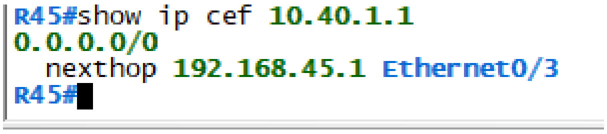
```
!
router ospf 1
 router-id 10.42.1.1
 area 30 stub
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/2
 no passive-interface Ethernet0/3
 network 10.42.1.1 0.0.0.0 area 0
 network 172.16.42.0 0.0.0.255 area 40
 network 192.168.41.0 0.0.0.255 area 0
 network 192.168.42.0 0.0.0.255 area 0
 network 192.168.45.0 0.0.0.255 area 30
!
```



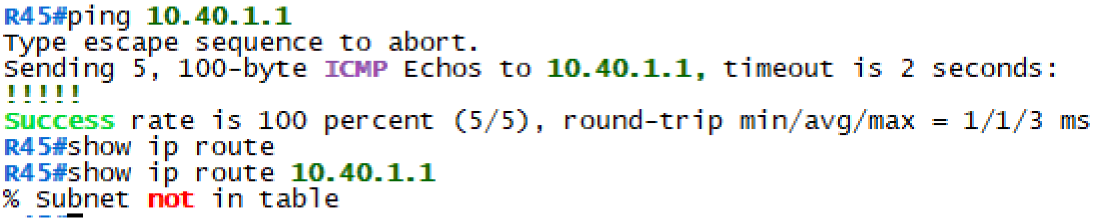
```
router ospf 1
 router-id 10.45.1.1
 area 30 stub
 passive-interface default
 no passive-interface Ethernet0/3
 network 10.45.1.0 0.0.0.255 area 30
 network 10.45.2.0 0.0.0.255 area 30
 network 192.168.45.0 0.0.0.255 area 30
!
```



```
R45#show ip route | in Gateway
Gateway of last resort is 192.168.45.1 to network 0.0.0.0
R45#
```



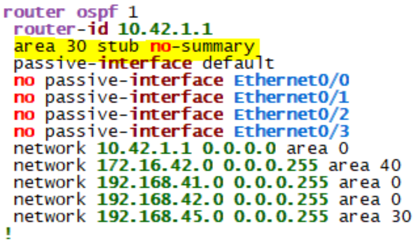
```
R45#show ip cef 10.40.1.1
0.0.0.0/0
  nexthop 192.168.45.1 Ethernet0/3
R45#
```



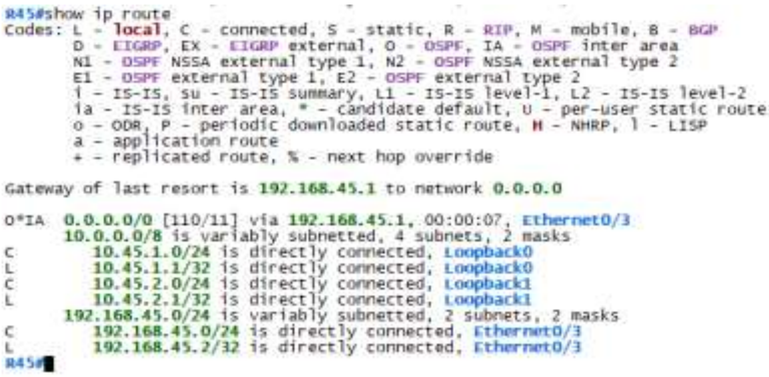
```
R45#ping 10.40.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.40.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/3 ms
R45#show ip route
R45#show ip route 10.40.1.1
% Subnet not in table
```

Totally Stubby Area:

-Removes External routes (LSA 5)
-Removes ASBR advertisements (LSA 4)
-Removes Inter-area default route (LSA 3)



```
router ospf 1
 router-id 10.42.1.1
 area 30 stub no-summary
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/2
 no passive-interface Ethernet0/3
 network 10.42.1.1 0.0.0.0 area 0
 network 172.16.42.0 0.0.0.255 area 40
 network 192.168.41.0 0.0.0.255 area 0
 network 192.168.42.0 0.0.0.255 area 0
 network 192.168.45.0 0.0.0.255 area 30
!
```

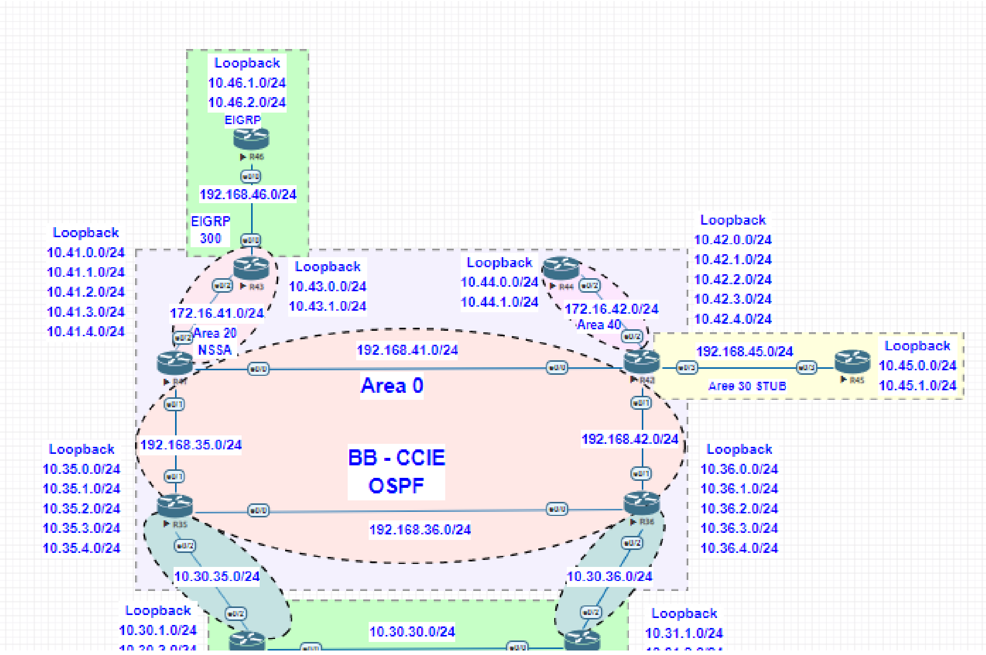


```
R45#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.45.1 to network 0.0.0.0

O*IA  0.0.0.0/0 [110/11] via 192.168.45.1, 00:00:07, Ethernet0/3
      10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C        10.45.1.0/24 is directly connected, Loopback0
L        10.45.1.1/32 is directly connected, Loopback0
C        10.45.2.0/24 is directly connected, Loopback1
L        10.45.2.1/32 is directly connected, Loopback1
      192.168.45.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.45.0/24 is directly connected, Ethernet0/3
L        192.168.45.2/32 is directly connected, Ethernet0/3
R45#
```

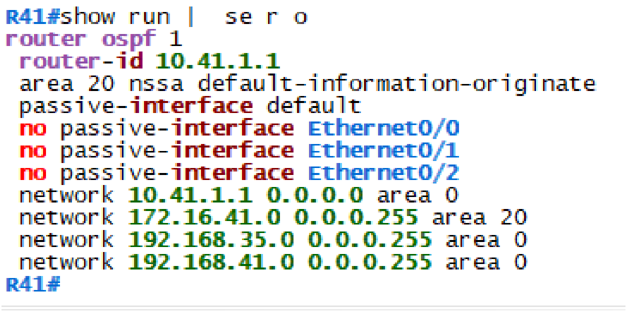
Not-so-stubby area: (NSSA)

- Allows NSSA external generation (LSA 7)
- Removes External routes (LSA 5)
- Removes ASBR Advertisements (LSA 4)
- All routers must agree on the NSSA

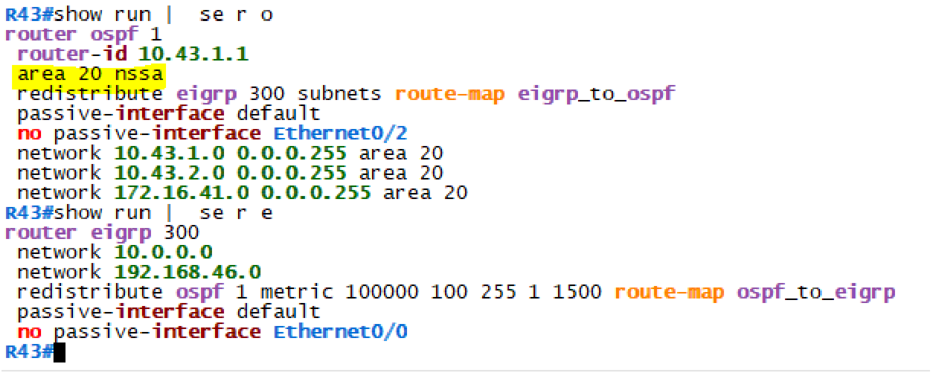
I have extended the network on R43 so that I can see the difference on NSSA



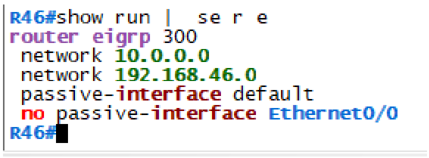
In the above topology, I am running OSPF and EIGRP on R43, so we can see the different of N Routes how populated and couple of issue we see when redistribute done between different IGP processes.



```
R41#show run |  se r o
router ospf 1
 router-id 10.41.1.1
 area 20 nssa default-information-originate
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/2
 network 10.41.1.1 0.0.0.0 area 0
 network 172.16.41.0 0.0.0.255 area 20
 network 192.168.35.0 0.0.0.255 area 0
 network 192.168.41.0 0.0.0.255 area 0
R41#
```

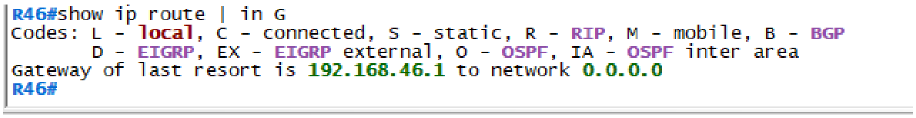


```
R43#show run |  se r o
router ospf 1
 router-id 10.43.1.1
 area 20 nssa
 redistribute eigrp 300 subnets route-map eigrp_to_ospf
 passive-interface default
 no passive-interface Ethernet0/2
 network 10.43.1.0 0.0.0.255 area 20
 network 10.43.2.0 0.0.0.255 area 20
 network 172.16.41.0 0.0.0.255 area 20
R43#show run |  se r e
router eigrp 300
 network 10.0.0.0
 network 192.168.46.0
 redistribute ospf 1 metric 100000 100 255 1 1500 route-map ospf_to_eigrp
 passive-interface default
 no passive-interface Ethernet0/0
R43#
```

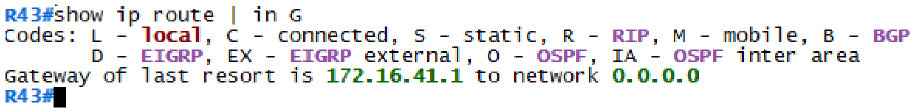


```
R46#show run |  se r e
router eigrp 300
 network 10.0.0.0
 network 192.168.46.0
 passive-interface default
 no passive-interface Ethernet0/0
R46#
```

Now we verify, R46. R43 Default Route injected from R41 (default-information-originate)

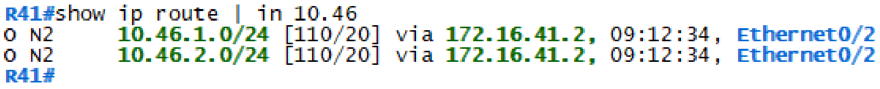


```
R46#show ip route | in G
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
Gateway of last resort is 192.168.46.1 to network 0.0.0.0
R46#
```

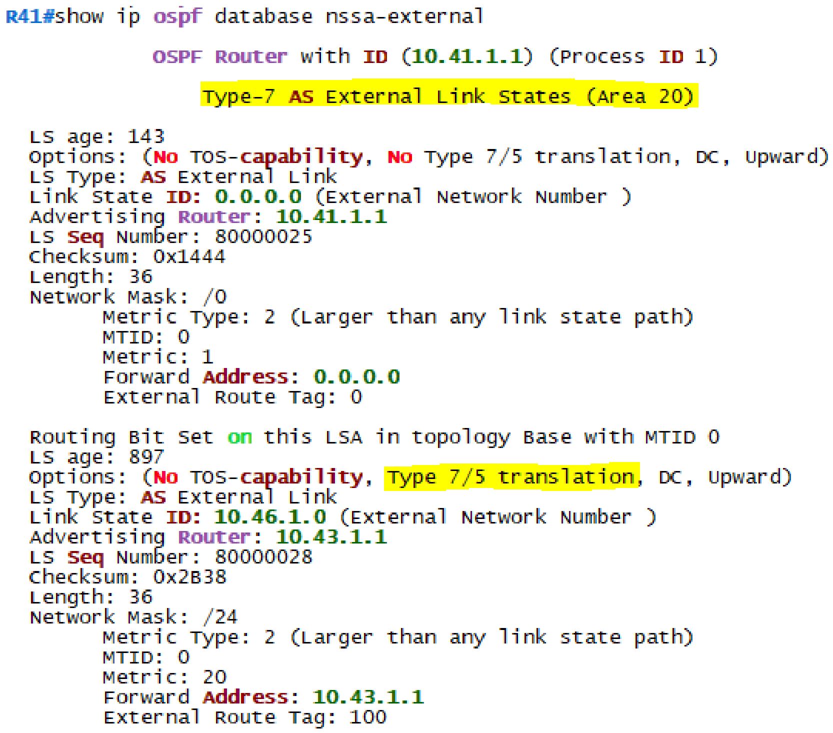


```
R43#show ip route | in G
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
Gateway of last resort is 172.16.41.1 to network 0.0.0.0
R43#
```

Verify the routes in R41

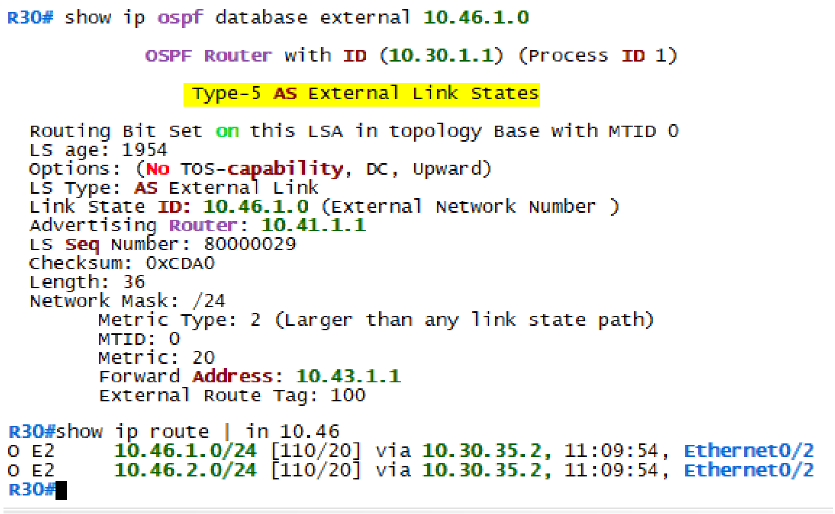


```
R41#show ip route | in 10.46
O N2     10.46.1.0/24 [110/20] via 172.16.41.2, 09:12:34, Ethernet0/2
O N2     10.46.2.0/24 [110/20] via 172.16.41.2, 09:12:34, Ethernet0/2
R41#
```



```
R41#show ip ospf database nssa-external

            OSPF Router with ID (10.41.1.1) (Process ID 1)

                Type-7 AS External Link States (Area 20)

  LS age: 143
  Options: (No TOS-capability, No Type 7/5 translation, DC, Upward)
  LS Type: AS External Link
  Link State ID: 0.0.0.0 (External Network Number )
  Advertising Router: 10.41.1.1
  LS Seq Number: 80000025
  Checksum: 0x1444
  Length: 36
  Network Mask: /0
        Metric Type: 2 (Larger than any link state path)
        MTID: 0
        Metric: 1
        Forward Address: 0.0.0.0
        External Route Tag: 0

  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 897
  Options: (No TOS-capability, Type 7/5 translation, DC, Upward)
  LS Type: AS External Link
  Link State ID: 10.46.1.0 (External Network Number )
  Advertising Router: 10.43.1.1
  LS Seq Number: 80000028
  Checksum: 0x2B38
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        MTID: 0
        Metric: 20
        Forward Address: 10.43.1.1
        External Route Tag: 100
```

Area 0 you see type 5 LSA



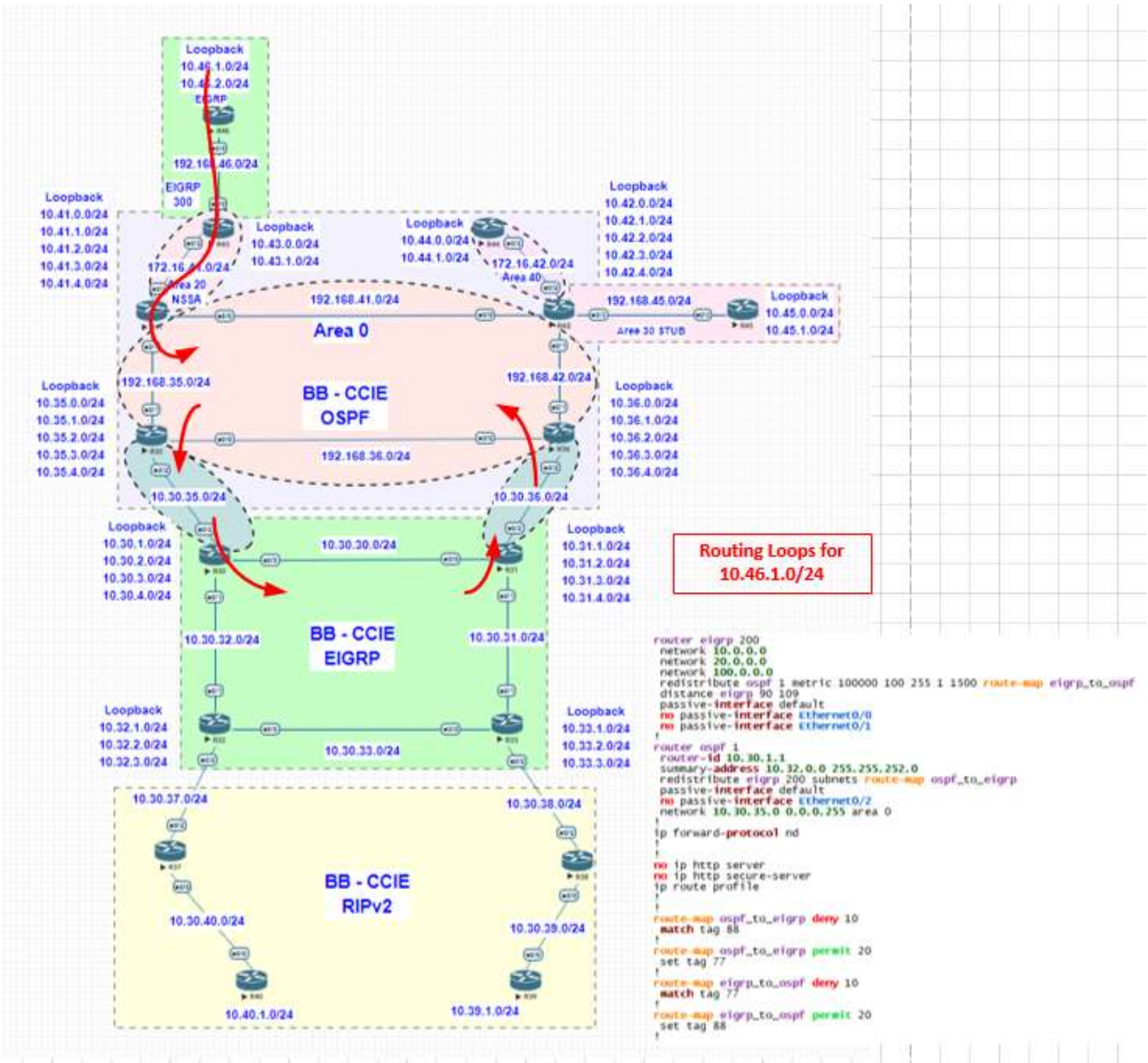
```
R30# show ip ospf database external 10.46.1.0

            OSPF Router with ID (10.30.1.1) (Process ID 1)

                Type-5 AS External Link States

  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 1954
  Options: (No TOS-capability, DC, Upward)
  LS Type: AS External Link
  Link State ID: 10.46.1.0 (External Network Number )
  Advertising Router: 10.41.1.1
  LS Seq Number: 80000029
  Checksum: 0xCDA0
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        MTID: 0
        Metric: 20
        Forward Address: 10.43.1.1
        External Route Tag: 100

R30#show ip route | in 10.46
O E2     10.46.1.0/24 [110/20] via 10.30.35.2, 11:09:54, Ethernet0/2
O E2     10.46.2.0/24 [110/20] via 10.30.35.2, 11:09:54, Ethernet0/2
R30#
```

How to Prevent Routing Loops when we doing Multiple Redistribution Points Issues

Solution with: **metrics**, **administrative distance**, and **route tags**

On the Bigger picture, of below topology, when 2 process of IGP redistributing, on the ASBR you need to have Route TAG to deny the routes with multi exit point network, if not you have routing loops, since same route will be advertised in OSPF routing table (this is draw back)

Fix as below: You need to have R30 and R31 route-map TAG (for solution)

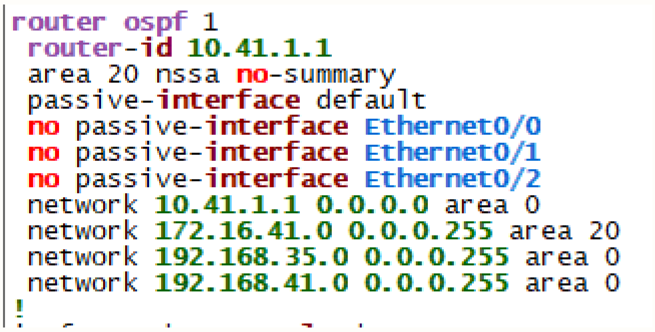


```
router eigrp 200
 network 10.0.0.0
 network 20.0.0.0
 network 100.0.0.0
 redistribute ospf 1 metric 100000 100 255 1 1500 route-map eigrp_to_ospf
 distance eigrp 90 109
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
!
router ospf 1
 router-id 10.30.1.1
 summary-address 10.32.0.0 255.255.252.0
 redistribute eigrp 200 subnets route-map ospf_to_eigrp
 passive-interface default
 no passive-interface Ethernet0/2
 network 10.30.35.0 0.0.0.255 area 0
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
ip route profile
!
!
route-map ospf_to_eigrp deny 10
 match tag 88
!
route-map ospf_to_eigrp permit 20
 set tag 77
!
route-map eigrp_to_ospf deny 10
 match tag 77
!
route-map eigrp_to_ospf permit 20
 set tag 88
!
```

NSSA – Totally Stub
- Allows NSSA external generation (LSA 7)
- Removes External routes (LSA 5)
- Removes ASBR Advertisements (LSA 4)
- All routers must agree on the NSSA

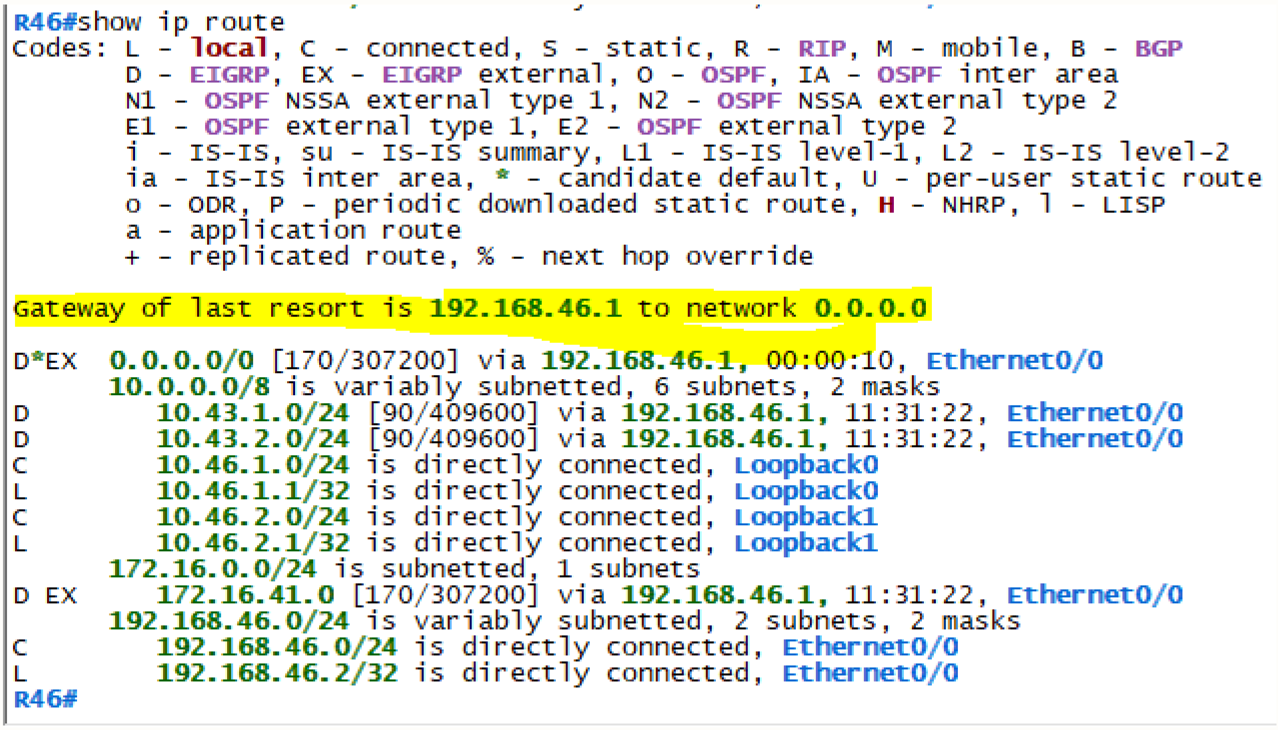
- Block (LSA 3)
- Injects Default Gateway

R41 we make config change to check this.



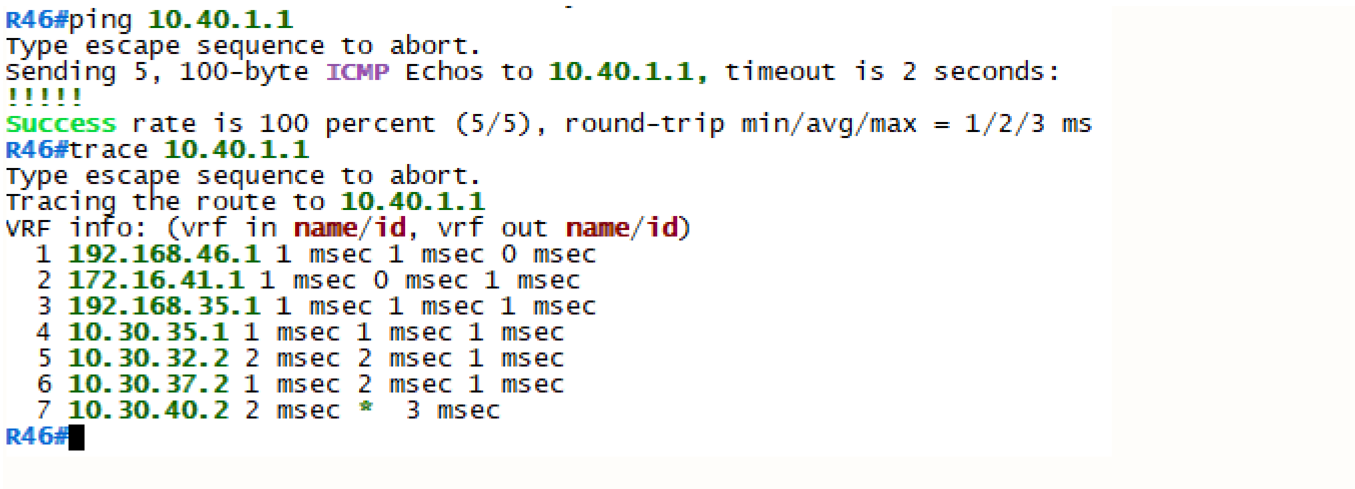
```
router ospf 1
 router-id 10.41.1.1
 area 20 nssa no-summary
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/2
 network 10.41.1.1 0.0.0.0 area 0
 network 172.16.41.0 0.0.0.255 area 20
 network 192.168.35.0 0.0.0.255 area 0
 network 192.168.41.0 0.0.0.255 area 0
!
```

R46 we see all routes were disappeared and have only routes locally available and have default gateway.



```
R46#show ip route
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.46.1 to network 0.0.0.0

D*EX  0.0.0.0/0 [170/307200] via 192.168.46.1, 00:00:10, Ethernet0/0
      10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
D        10.43.1.0/24 [90/409600] via 192.168.46.1, 11:31:22, Ethernet0/0
D        10.43.2.0/24 [90/409600] via 192.168.46.1, 11:31:22, Ethernet0/0
C        10.46.1.0/24 is directly connected, Loopback0
L        10.46.1.1/32 is directly connected, Loopback0
C        10.46.2.0/24 is directly connected, Loopback1
L        10.46.2.1/32 is directly connected, Loopback1
      172.16.0.0/24 is subnetted, 1 subnets
D EX     172.16.41.0 [170/307200] via 192.168.46.1, 11:31:22, Ethernet0/0
      192.168.46.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.46.0/24 is directly connected, Ethernet0/0
L        192.168.46.2/32 is directly connected, Ethernet0/0
R46#
```

R46 still have reachability to R40



```
R46#ping 10.40.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.40.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/3 ms
R46#trace 10.40.1.1
Type escape sequence to abort.
Tracing the route to 10.40.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.46.1 1 msec 1 msec 0 msec
  2 172.16.41.1 1 msec 0 msec 1 msec
  3 192.168.35.1 1 msec 1 msec 1 msec
  4 10.30.35.1 1 msec 1 msec 1 msec
  5 10.30.32.2 2 msec 2 msec 1 msec
  6 10.30.37.2 1 msec 2 msec 1 msec
  7 10.30.40.2 2 msec *  3 msec
R46#
```

2.6.d [iii] Internal router, ABR, ASBR

There are four types of OSPF routers which are determined by a router's function and/or location within an OSPF area:
Internal (IR) – all OSPF interfaces must belong to the same OSPF area.
Backbone – at least one OSPF interface must belong to area 0 (backbone area)
Area Border Router (ABR) – at least one OSPF interface must belong to area 0 (backbone area) and at least one OSPF interface must belong to a non-backbone (area 0) area.
Autonomous System Boundry Router (ASBR) – an OSPF router that performs route injection (redistribution) from another route source (RIP, EIGRP, IS-IS, BGP, another OSPF process, etc.).

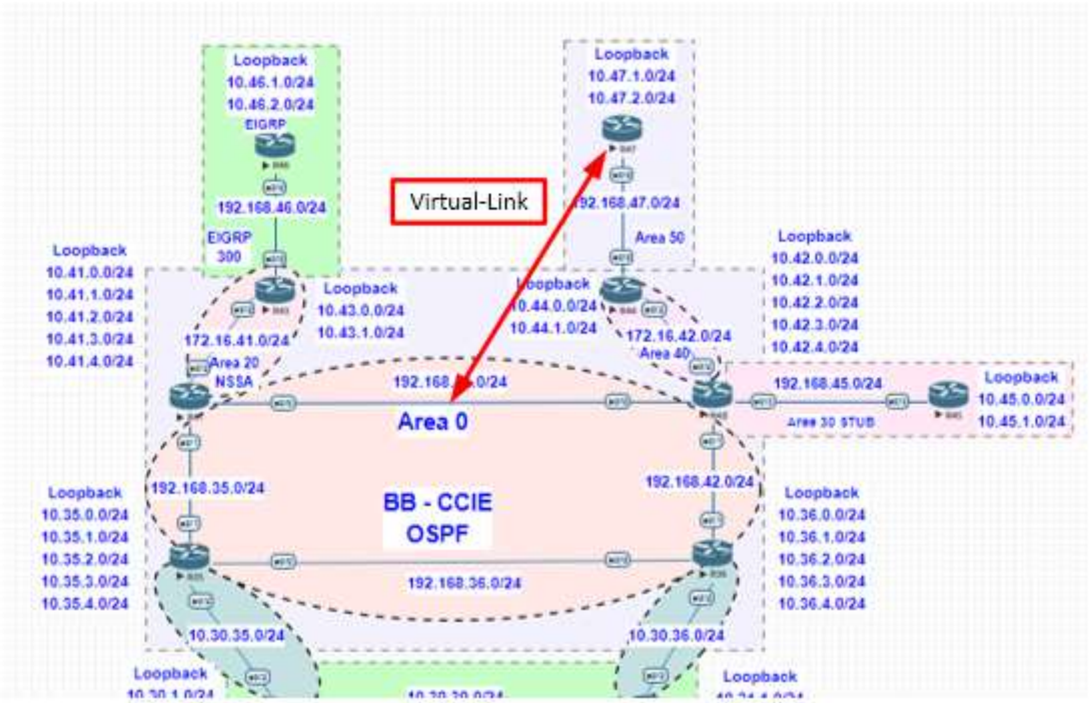
2.6.d [iv] Virtual link

Virtual links are used for two purposes:

- Linking an area that does not have a physical connection to the backbone.
- Patching the backbone in case discontinuity of area 0 occurs.

Areas Not Physically Connected to Area 0

Area 0 has to be at the center of all other areas. In some rare case where it is impossible to have an area physically connected to the backbone, a virtual link is used. The virtual link will provide the disconnected area a logical path to the backbone. The virtual link has to be established between two ABRs that have a common area, with one ABR connected to the backbone.

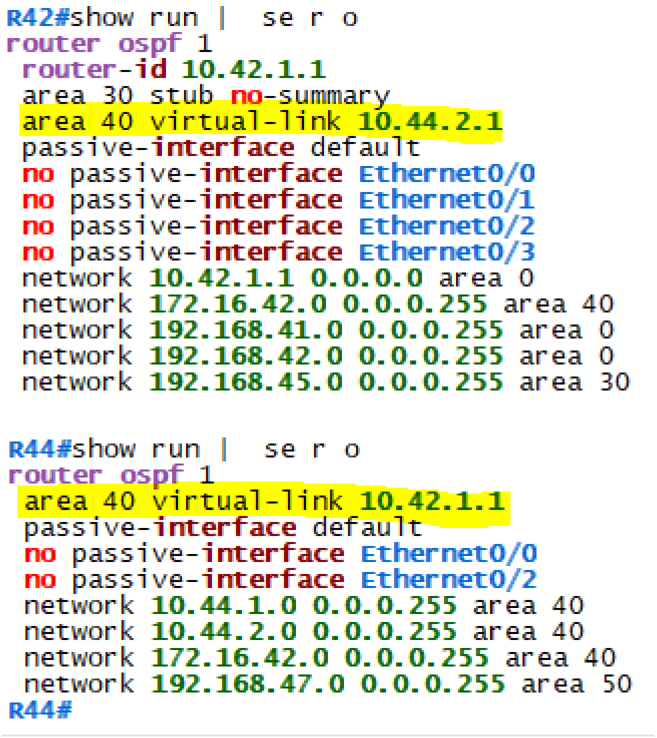


Setup virtual link between R44 and R42 So R47 can able to get all the routes from Area 0

R47 do not see any routes without an Virtual link.

```
R47#show ip route
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C        10.47.1.0/24 is directly connected, Loopback0
L        10.47.1.1/32 is directly connected, Loopback0
C        10.47.2.0/24 is directly connected, Loopback1
L        10.47.2.1/32 is directly connected, Loopback1
      192.168.47.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.47.0/24 is directly connected, Ethernet0/0
L        192.168.47.2/32 is directly connected, Ethernet0/0
R47#
R47#
R47#
```

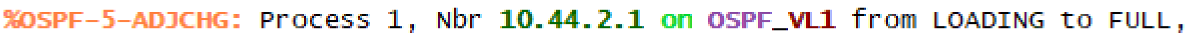
Now configure virtual link on R42 and R44



```
R42#show run |  se r o
router ospf 1
 router-id 10.42.1.1
 area 30 stub no-summary
 area 40 virtual-link 10.44.2.1
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/2
 no passive-interface Ethernet0/3
 network 10.42.1.1 0.0.0.0 area 0
 network 172.16.42.0 0.0.0.255 area 40
 network 192.168.41.0 0.0.0.255 area 0
 network 192.168.42.0 0.0.0.255 area 0
 network 192.168.45.0 0.0.0.255 area 30

R44#show run |  se r o
router ospf 1
 area 40 virtual-link 10.42.1.1
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/2
 network 10.44.1.0 0.0.0.255 area 40
 network 10.44.2.0 0.0.0.255 area 40
 network 172.16.42.0 0.0.0.255 area 40
 network 192.168.47.0 0.0.0.255 area 50
R44#
```

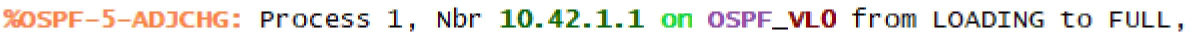
Now we can see the logs OSPF forms the neighbourship

R42

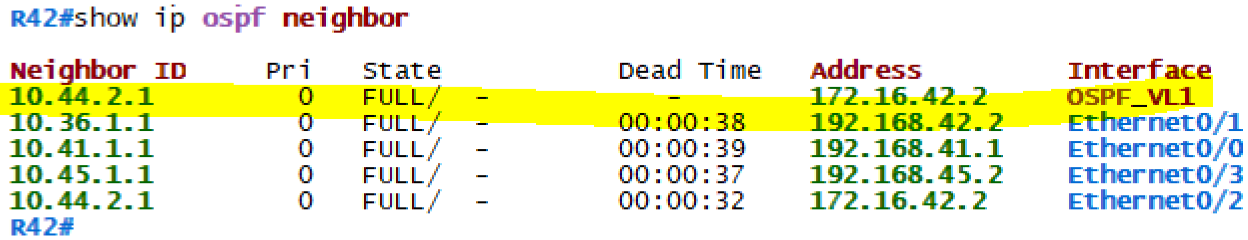


```
%OSPF-5-ADJCHG: Process 1, Nbr 10.44.2.1 on OSPF_VL1 from LOADING to FULL,
```

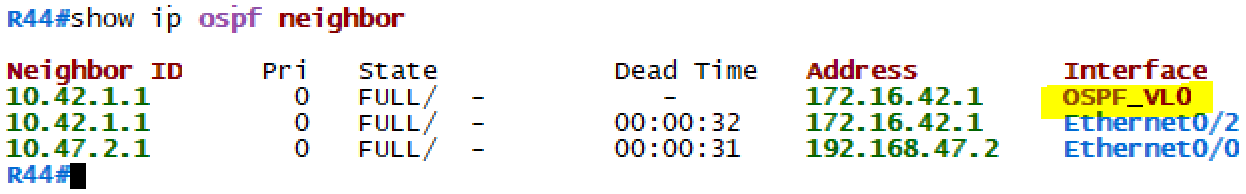
R44



```
%OSPF-5-ADJCHG: Process 1, Nbr 10.42.1.1 on OSPF_VL0 from LOADING to FULL,
```

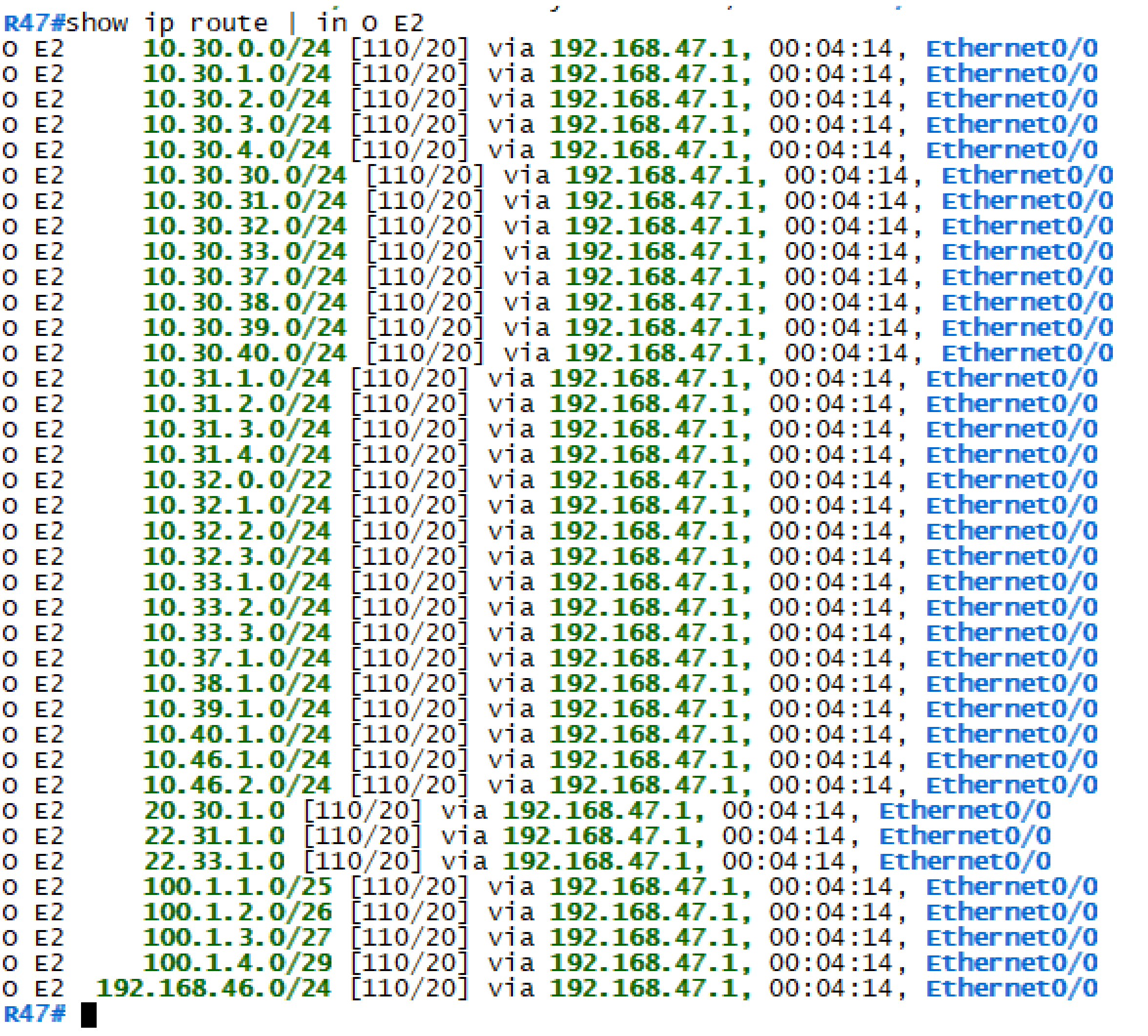


```
R42#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.44.2.1         0   FULL/  -            -       172.16.42.2     OSPF_VL1
10.36.1.1         0   FULL/  -        00:00:38    192.168.42.2    Ethernet0/1
10.41.1.1         0   FULL/  -        00:00:39    192.168.41.1    Ethernet0/0
10.45.1.1         0   FULL/  -        00:00:37    192.168.45.2    Ethernet0/3
10.44.2.1         0   FULL/  -        00:00:32    172.16.42.2     Ethernet0/2
R42#
```

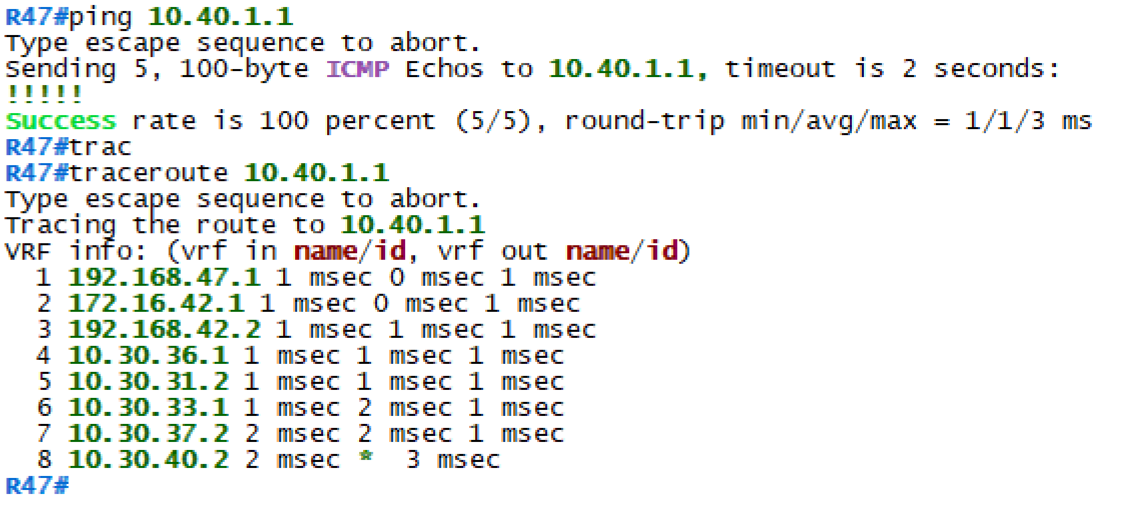


```
R44#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.42.1.1         0   FULL/  -            -       172.16.42.1     OSPF_VL0
10.42.1.1         0   FULL/  -        00:00:32    172.16.42.1     Ethernet0/2
10.47.2.1         0   FULL/  -        00:00:31    192.168.47.2    Ethernet0/0
R44#
```

Lets check the routes now in R47



```
R47#show ip route | in O E2
O E2     10.30.0.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.2.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.3.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.4.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.30.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.31.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.32.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.33.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.37.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.38.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.39.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.30.40.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.31.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.31.2.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.31.3.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.31.4.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.32.0.0/22 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.32.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.32.2.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.32.3.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.33.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.33.2.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.33.3.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.37.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.38.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.39.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.40.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.46.1.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     10.46.2.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     20.30.1.0 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     22.31.1.0 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     22.33.1.0 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     100.1.1.0/25 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     100.1.2.0/26 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     100.1.3.0/27 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2     100.1.4.0/29 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
O E2  192.168.46.0/24 [110/20] via 192.168.47.1, 00:04:14, Ethernet0/0
R47#
```



```
R47#ping 10.40.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.40.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/3 ms
R47#trac
R47#traceroute 10.40.1.1
Type escape sequence to abort.
Tracing the route to 10.40.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.47.1 1 msec 0 msec 1 msec
  2 172.16.42.1 1 msec 0 msec 1 msec
  3 192.168.42.2 1 msec 1 msec 1 msec
  4 10.30.36.1 1 msec 1 msec 1 msec
  5 10.30.31.2 1 msec 1 msec 1 msec
  6 10.30.33.1 1 msec 2 msec 1 msec
  7 10.30.37.2 2 msec 2 msec 1 msec
  8 10.30.40.2 2 msec *  3 msec
R47#
```

2.6.e Implement and troubleshoot path preference

2.6.f Implement and troubleshoot operations

2.6.f [i] General operations

2.6.f [ii] Graceful shutdown

2.6.f [iii] GTSM [generic TTL security mechanism]

2.6.g Implement, troubleshoot and optimize OSPF convergence and scalability

2.6.g [i] Metrics

We can manipulate OSPF route cost in two ways.

- By changing bandwidth of interface
- By changing reference bandwidth value

Example :

```
interface eth 0/0
bandwidth xxx

router ospf 1
auto-cost reference-bandwidth 1000
```

2.6.g [ii] LSA throttling, SPF tuning, fast hello

2.6.g [iii] LSA propagation control [area types, ISPF]

2.6.g [iv] IP FR/fast reroute [single hop]

LFA Repair Paths

The figure below shows how the OSPFv2 Loop-Free Alternate Fast Reroute feature reroutes traffic if a link fails. A protecting router precomputes per-prefix repair paths and installs them in the global Routing Information Base (RIB). When the protected primary path fails, the protecting router diverts live traffic from the primary path to the stored repair path, without other routers' having to recomputed network topology or even be aware that the network topology has changed.

2.6.g [v] LFA/loop-free alternative [multi hop]

2.6.g [vi] OSPFv3 prefix suppression.

Suppress IPv4 and IPv6 Prefix Advertisements on a Per-Interface Basis

You can explicitly configure an OSPFv3 interface not to advertise its IP network to its neighbors by using the ipv6 ospf prefix-suppression command or the ospfv3 prefix-suppression command in interface configuration mode.

IPv6

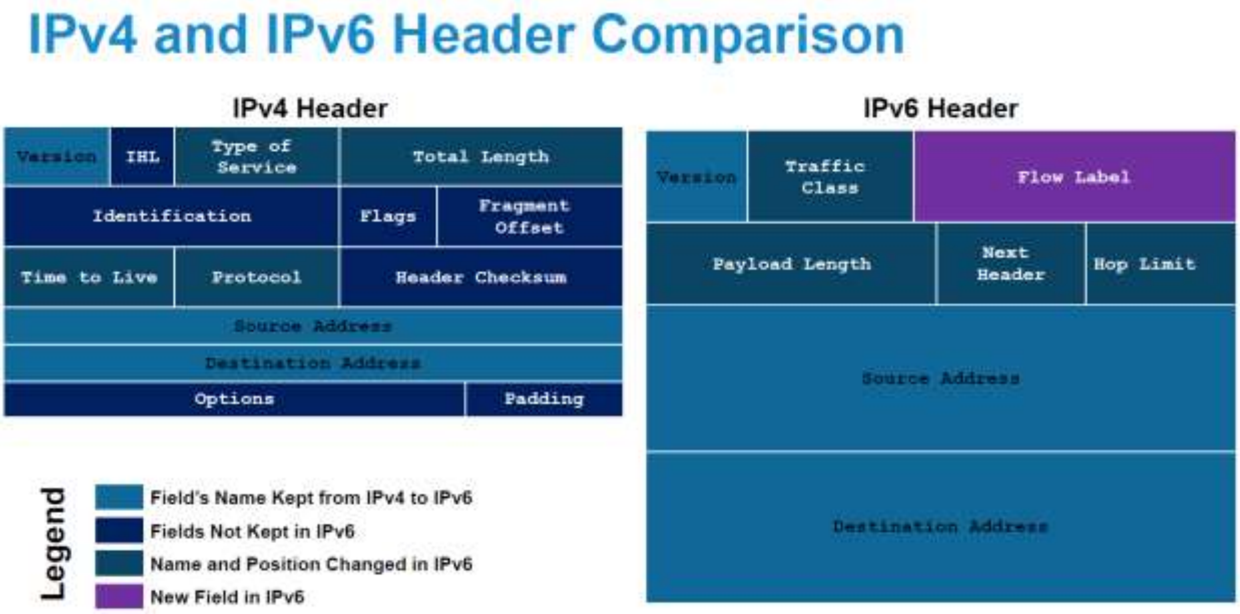
IPv6 has 128 bit addresses and has a much larger address space than 32-bit IPv4 which offered us a bit more than 4 billion addresses. Keep in mind every additional bit doubles the number of IP addresses…so we go from 4 billion to 8 billion, 16,32,64, etc. Keep doubling until you reach 128 bit.

Highlights:

- Addresses are 128 bits long
- Separated with colons every 16 bits
- Address separated in prefix and interface id, most common is /64
- Leading zeroes can be omitted from address and double colon may be used to represent Successive zeroes, may only be used once
- Unicast, multicast and anycast, doesn't use broadcast

The main reason to start using IPv6 is that we need more addresses but it also offers some new features:

- **No Broadcast traffic**: that's right, we don't use broadcasts anymore. We use multicast instead. This means some protocols like ARP are replaced with other solutions.
- **Stateless Autoconfiguration**: this is like a "mini DHCP server". Routers running IPv6 are able to advertise the IPv6 prefix and gateway address to hosts so that they can automatically configure themselves and get access outside of their own network.
- **Address Renumbering**: renumbering static IPv4 addresses on your network is a pain. If you use stateless autoconfiguration for IPv6 then you can easily swap the current prefix with another one.
- **Mobility**: IPv6 has built-in support for mobile devices. Hosts will be able to move from one network to another and keep their current IPv6 address.
- **No NAT / PAT**: we have so much IPv6 addresses that we don't need NAT or PAT anymore, every device in your network can have a public IPv6 address.
- **IPsec**: IPv6 has native support for IPsec, you don't have to use it but it's built-in the protocol.
- **Improved header**: the IPv6 header is simpler and doesn't require checksums. It also has a flow label that is used to quickly see if certain packets belong to the same flow or not.
- **Migration Tools**: IPv4 and IPv6 are **not compatible** so we need migration tools. There are multiple tunneling techniques that we can use to transport IPv6 over IPv4 networks (or the other way around). Running IPv4 and IPv6 simultaneously is called "dual stack".



IPv4 - Decimal
IPv6 - Hexa Decimal [0-9, A-F]

IPv6 - 8 Block of 4 hexa = 32bit Hex

Network / Host Portions of address

- IPv4
CLASS A = 8 bits for network 24 bits for host
CLASS B = 16 bits for network and 16 bits for host
CLASS C = 24 bits for nework and 8 bits for Host

- IPv6 - No Classes, it uses Flat Model

- First 64 bits NETWORK - Next 64bits HOST

Example :

2000:0000:0000:0000:0BED:DEDD:CEFD:1000

2000:0000:0000:0000 - NETWORK
0BED:DEDD:CEFD:1000 - HOST

- IPv6 complex so it rely on DHCP / DNS

- Subnetmask for IPv6 /64

Type of Address :

Public Address / Global Address -

First 3 bits of IPv6 public adress are 0 0 1
8 4 2 1
0 0 1 _ (0 or 1) that will be 2 or 3

The First hex digit Global Address either 2 or 3 start with example :

2000:0000:0000:0000:0BED:DEDD:CEFD:1000
3000:0000:0000:0000:0BED:DEDD:CEFD:1000

Private Address / Unique Local Address -

First 7 bits of IPv6 private adress are 1 1 1 1 1 1 0 _
F C or D
1 1 1 1 1 1 0 _

Example :

FC or FD

FC00:0000:0000:0000:0BED:DEDD:CEFD:1000
FD00:0000:0000:0000:0BED:DEDD:CEFD:1000

Lets write Class C network.
Lets write example with 192.168.1.1 to IPV6
FC00:192:168:1::1/64
FC00:192:168:1::2/64

FD00:192:168:1::1/64
FD00:192:168:1::2/64

Lets write Class A network.

10.10.10.1

FC00:10:10:10::1/64
FC00:10:10:10::2/64

Lets write Class B network.
FC00:172:168:1::1/64
FC00:172:168:1::2/64

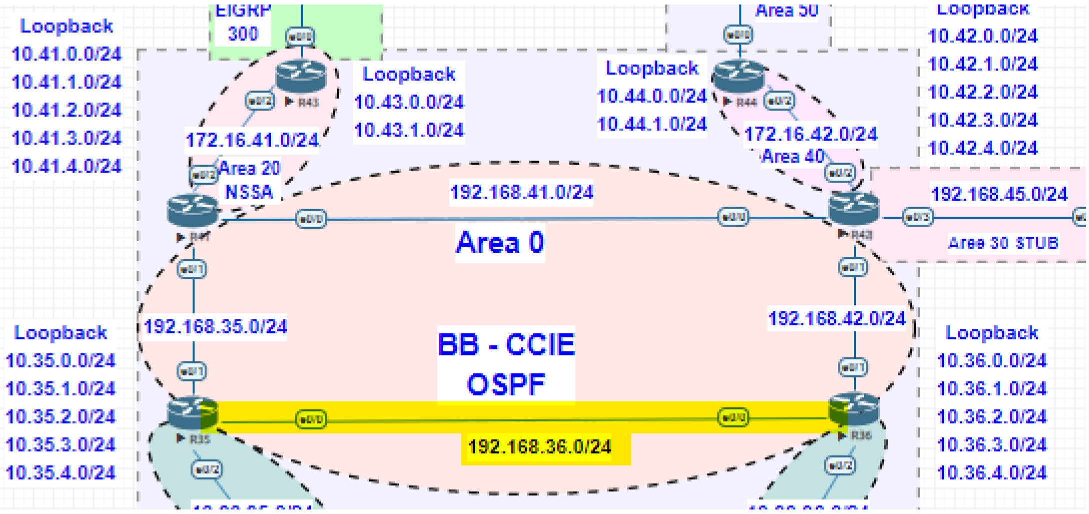
High Level :

```
#Multicast
        #ff00::/8
        #ff01:0:0:0:0:0:2
        #224.0.0.0/4
#All
        # ::/128
        # 0.0.0.0
#LoopBack
        # ::1/128
        # 127.0.0.1
#Link Local Address(non routable)
        # fe80::/10, fe90::/10, feA0::/10, feB0::/10
        # fe80::200:5aee:feaa:22a2
        # 169.254.0.0/16
        # Not Routable between interfaces
        # Used for : SLAAC, Neighbor Discovery, Router Discovery

#Unique Local Addresses(ULAs)
        # fc00::1/7
        # fdf8:f53b:82e4::53
        # 10.0.0.0/8 - 172.16.0.0/12 - 192.168.0.0/16
        # Private ipv6 address/not routable via global BGP/for internal use only

#Global Unicast
        #2000::/3 (public IP)
```

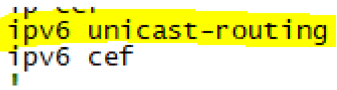
Multicast table to understand

Routing Protocol	IPv4	IPv6
RIP	224.0.0.9	FF02::9
EIGRP	224.0.0.10	FF02::A
OSPF	224.0.0.5	FF02::5
	224.0.0.6	FF02::6

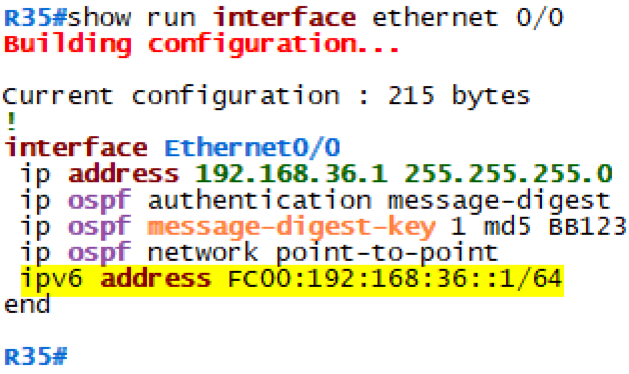
Now we configure simple configuration on our exiting topology
R35 to R36 using Interface Eth 0/0



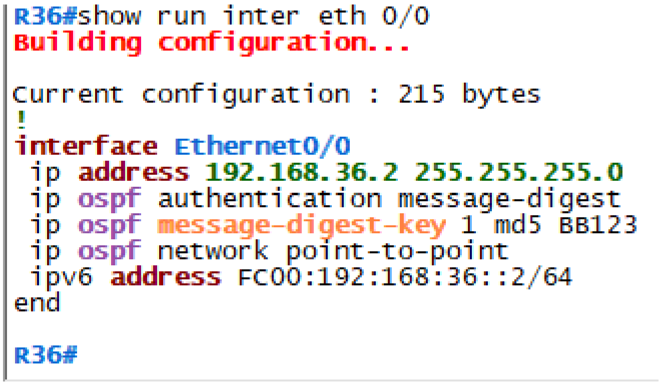
You need to enable IPv6 Globally to work in IPv6



```
ip cef
ipv6 unicast-routing
ipv6 cef
!
```

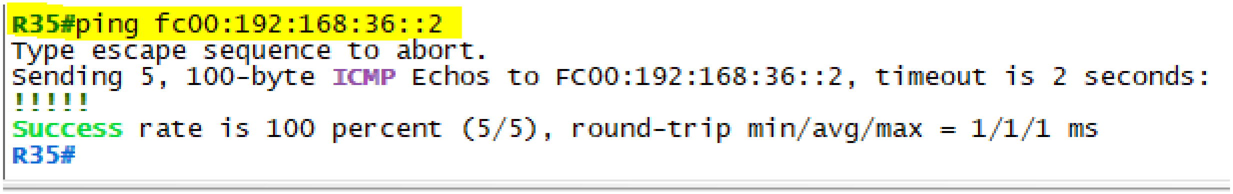


```
R35#show run interface ethernet 0/0
Building configuration...

Current configuration : 215 bytes
!
interface Ethernet0/0
 ip address 192.168.36.1 255.255.255.0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 BB123
 ip ospf network point-to-point
 ipv6 address FC00:192:168:36::1/64
end

R35#
```

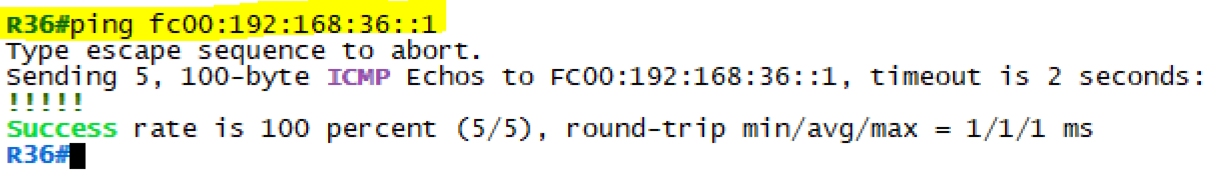


```
R36#show run inter eth 0/0
Building configuration...

Current configuration : 215 bytes
!
interface Ethernet0/0
 ip address 192.168.36.2 255.255.255.0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 BB123
 ip ospf network point-to-point
 ipv6 address FC00:192:168:36::2/64
end

R36#
```

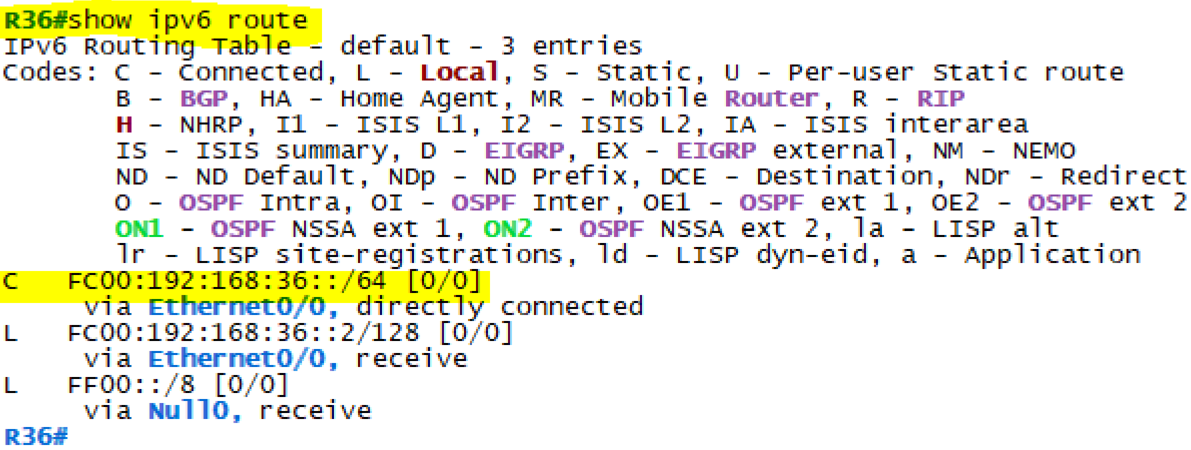
Now we ping from 1 to 2 and vice versa



```
R35#ping fc00:192:168:36::2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:192:168:36::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R35#
```

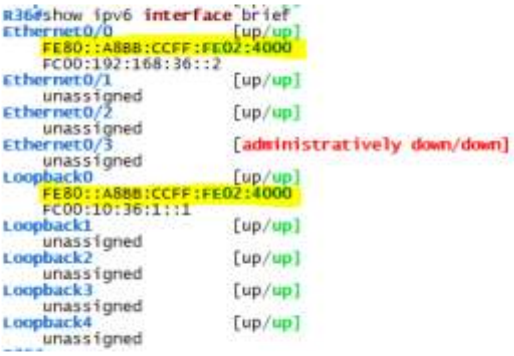


```
R36#ping fc00:192:168:36::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:192:168:36::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R36#
```



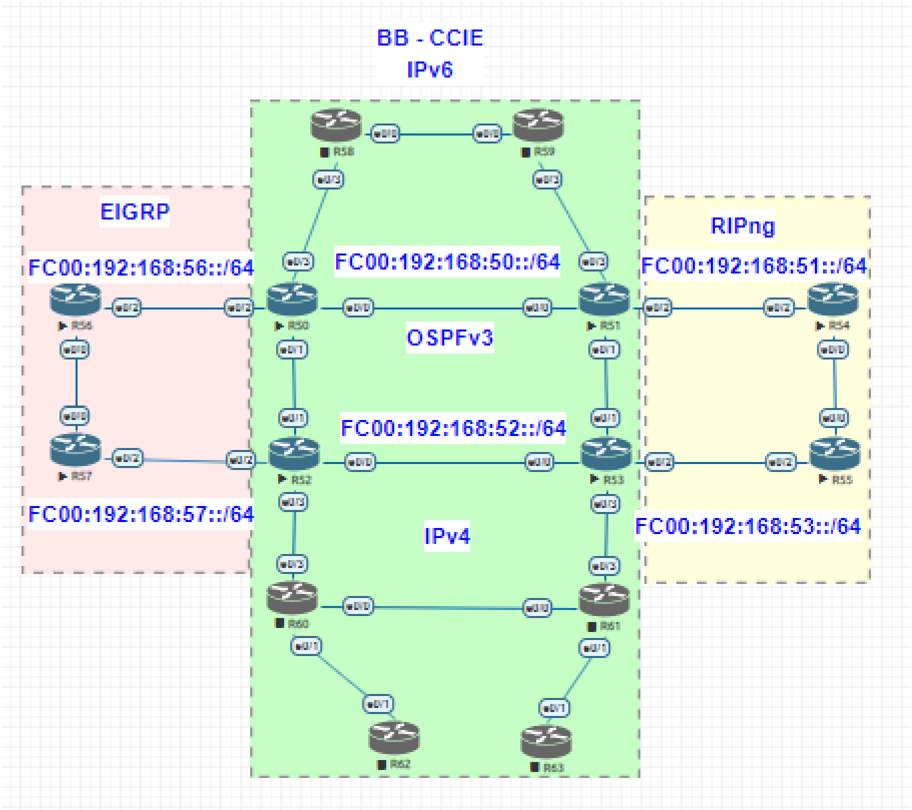
```
R36#show ipv6 route
IPv6 Routing Table - default - 3 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
C   FC00:192:168:36::/64 [0/0]
     via Ethernet0/0, directly connected
L   FC00:192:168:36::2/128 [0/0]
     via Ethernet0/0, receive
L   FF00::/8 [0/0]
     via Null0, receive
R36#
```

Link Local address: uses MAC address of interface and add in the middle FF:FE as example below :

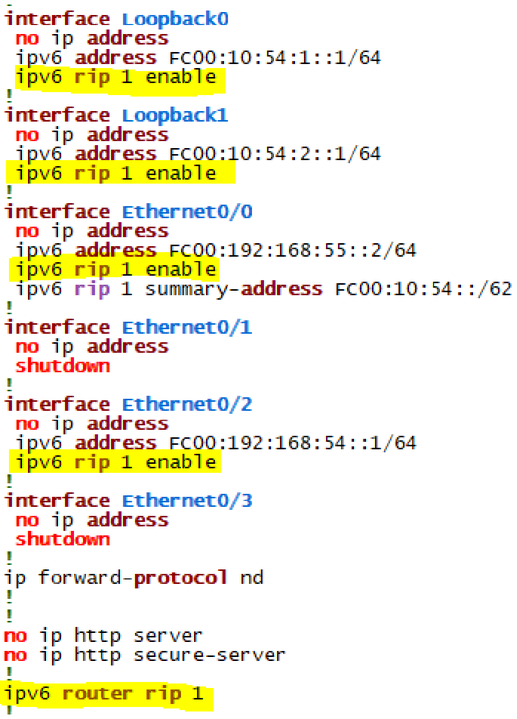


```
R36#show ipv6 interface brief
Ethernet0/0            [up/up]
    FE80::A8BB:CCFF:FE02:4000
    FC00:192:168:36::2
Ethernet0/1            [up/up]
    unassigned
Ethernet0/2            [up/up]
    unassigned
Ethernet0/3            [administratively down/down]
    unassigned
Loopback0              [up/up]
    FE80::A8BB:CCFF:FE02:4000
    FC00:10:36:1::1
Loopback1              [up/up]
    unassigned
Loopback2              [up/up]
    unassigned
Loopback3              [up/up]
    unassigned
Loopback4              [up/up]
    unassigned
```

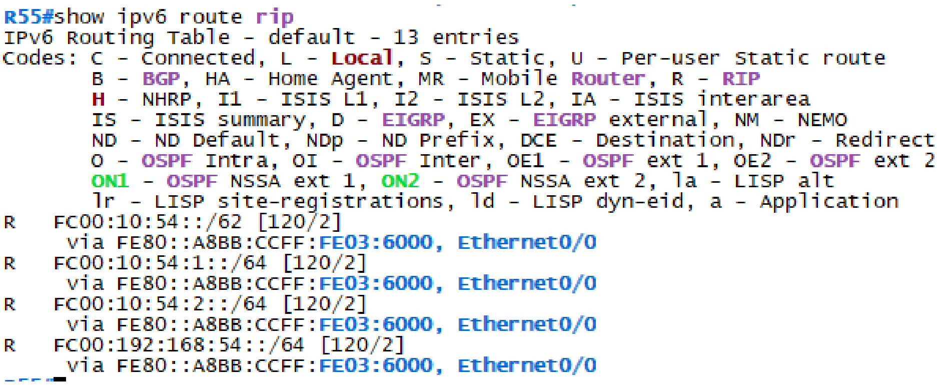
Create a simple LAB for IPv6 to test, which has OSPF / EIGRP / RIP



Lets configure RIPNG on R51, R53, R54, R55 check the router

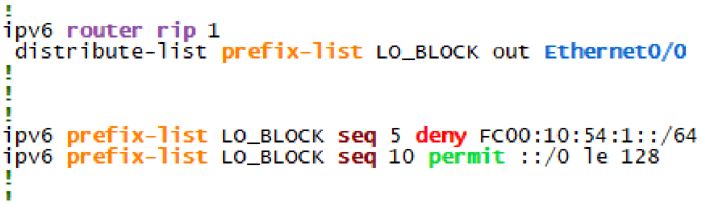

```
!
interface Loopback0
 no ip address
 ipv6 address FC00:10:54:1::1/64
 ipv6 rip 1 enable
!
interface Loopback1
 no ip address
 ipv6 address FC00:10:54:2::1/64
 ipv6 rip 1 enable
!
interface Ethernet0/0
 no ip address
 ipv6 address FC00:192:168:55::2/64
 ipv6 rip 1 enable
 ipv6 rip 1 summary-address FC00:10:54::/62
!
interface Ethernet0/1
 no ip address
 shutdown
!
interface Ethernet0/2
 no ip address
 ipv6 address FC00:192:168:54::1/64
 ipv6 rip 1 enable
!
interface Ethernet0/3
 no ip address
 shutdown
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
ipv6 router rip 1
!
```

Check RIP Route available on R55 we can see the routes of RIP and loopback ipv6 address from R54



```
R55#show ipv6 route rip
IPv6 Routing Table - default - 13 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
R   FC00:10:54::/62 [120/2]
     via FE80::A8BB:CCFF:FE03:6000, Ethernet0/0
R   FC00:10:54:1::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:6000, Ethernet0/0
R   FC00:10:54:2::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:6000, Ethernet0/0
R   FC00:192:168:54::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:6000, Ethernet0/0
```

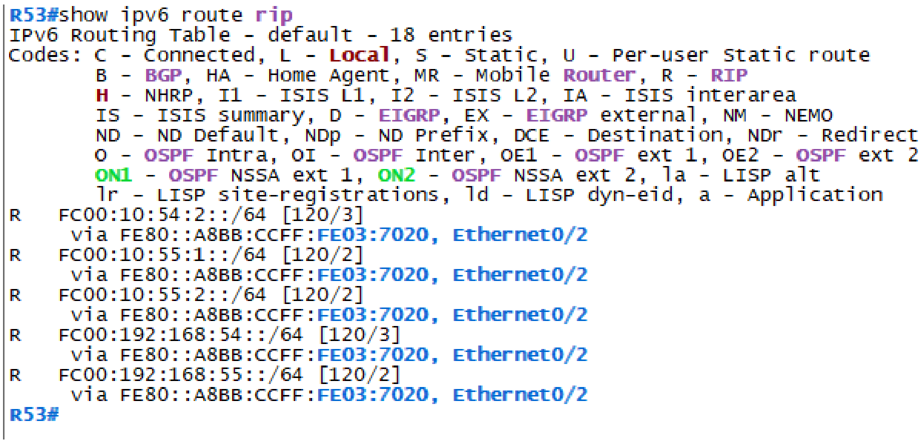
Now we block sending IPv6 loopback address from R54



```
!
ipv6 router rip 1
 distribute-list prefix-list LO_BLOCK out Ethernet0/0
!
!
!
ipv6 prefix-list LO_BLOCK seq 5 deny FC00:10:54:1::/64
ipv6 prefix-list LO_BLOCK seq 10 permit ::/0 le 128
!
!
```

Now we verify on R53

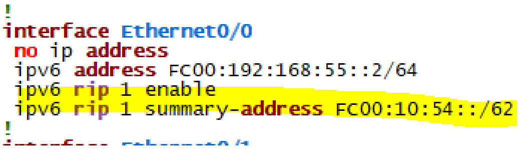
No route



```
R53#show ipv6 route rip
IPv6 Routing Table - default - 18 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
R   FC00:10:54:2::/64 [120/3]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R   FC00:10:55:1::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R   FC00:10:55:2::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R   FC00:192:168:54::/64 [120/3]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R   FC00:192:168:55::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R53#
```

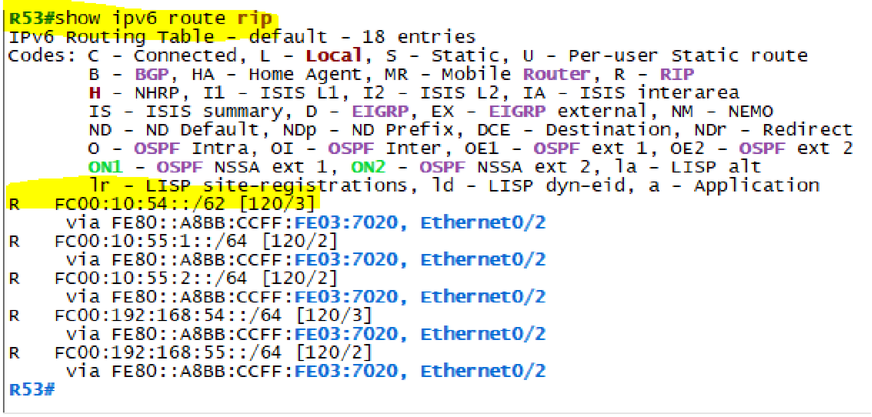
Summary

Now we summary R54 Loopback address



```
!
interface Ethernet0/0
 no ip address
 ipv6 address FC00:192:168:55::2/64
 ipv6 rip 1 enable
 ipv6 rip 1 summary-address FC00:10:54::/62
!
interface Ethernet0/1
```

Verify on R53 for this summarisation:

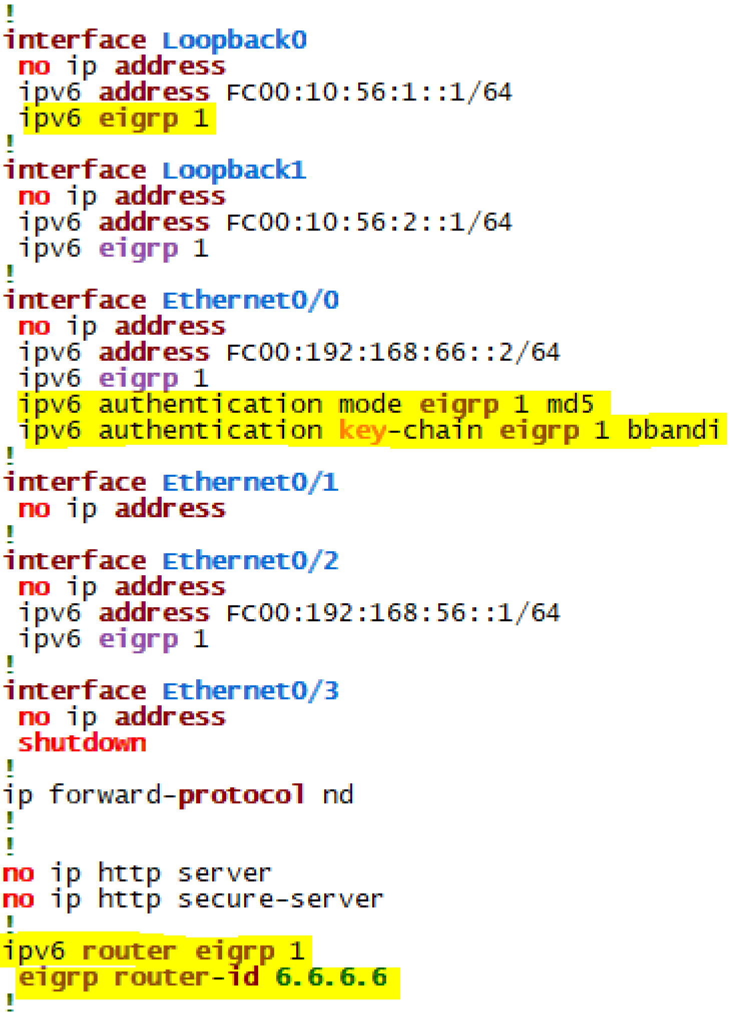


```
R53#show ipv6 route rip
IPv6 Routing Table - default - 18 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
R   FC00:10:54::/62 [120/3]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R   FC00:10:55:1::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R   FC00:10:55:2::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R   FC00:192:168:54::/64 [120/3]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R   FC00:192:168:55::/64 [120/2]
     via FE80::A8BB:CCFF:FE03:7020, Ethernet0/2
R53#
```

IPv6 EIGRP

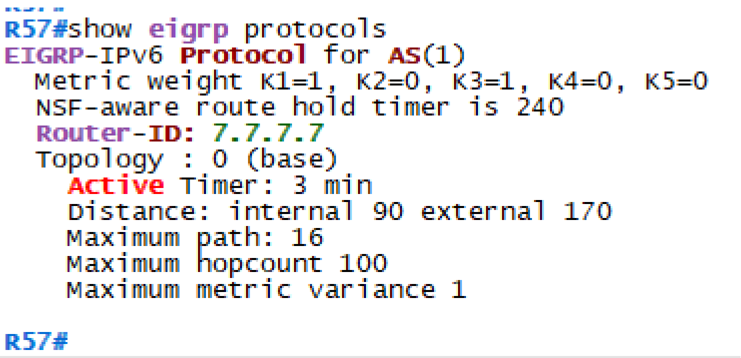
In our topology R56 / R57 / R50 /R52 will be part of EIGRP domain.

We can configure basic config as below which contains authentication and enable EIGRP on the interface.

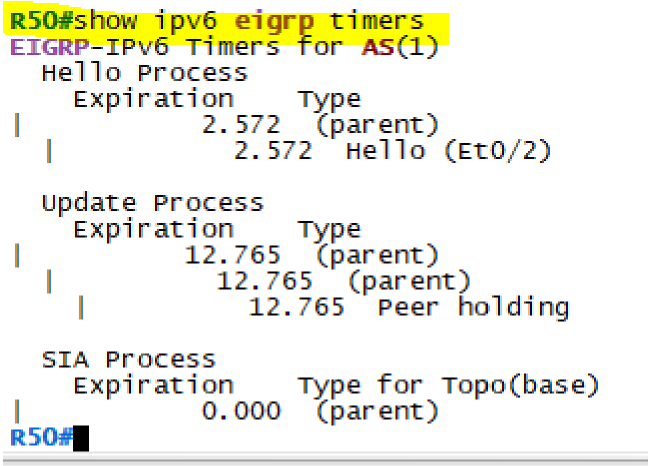


```
!
interface Loopback0
 no ip address
 ipv6 address FC00:10:56:1::1/64
 ipv6 eigrp 1
!
interface Loopback1
 no ip address
 ipv6 address FC00:10:56:2::1/64
 ipv6 eigrp 1
!
interface Ethernet0/0
 no ip address
 ipv6 address FC00:192:168:66::2/64
 ipv6 eigrp 1
 ipv6 authentication mode eigrp 1 md5
 ipv6 authentication key-chain eigrp 1 bbandi
!
interface Ethernet0/1
 no ip address
!
interface Ethernet0/2
 no ip address
 ipv6 address FC00:192:168:56::1/64
 ipv6 eigrp 1
!
interface Ethernet0/3
 no ip address
 shutdown
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
ipv6 router eigrp 1
 eigrp router-id 6.6.6.6
!
```

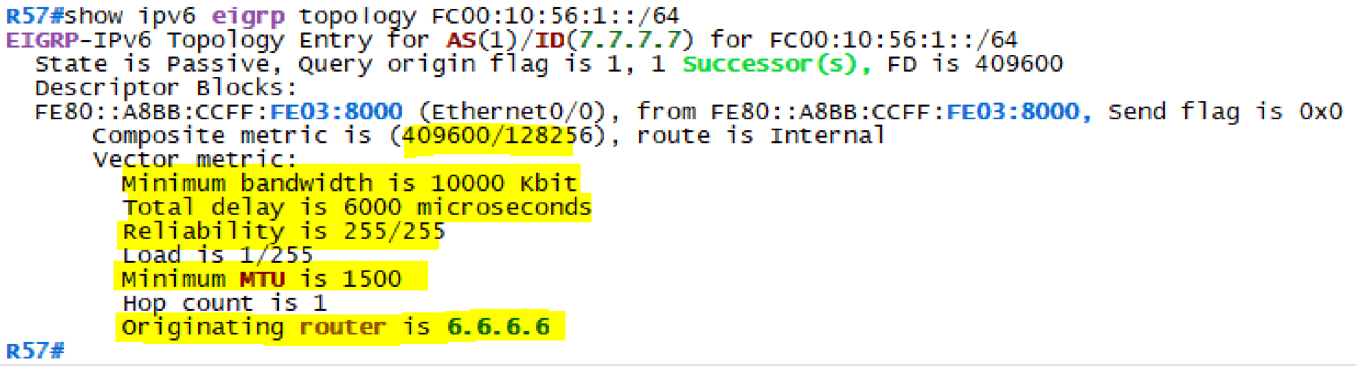
You can view default timers, maximum path, variance, distance internal and external, K values



```
R57#show eigrp protocols
EIGRP-IPv6 Protocol for AS(1)
  Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  NSF-aware route hold timer is 240
  Router-ID: 7.7.7.7
  Topology : 0 (base)
    Active Timer: 3 min
    Distance: internal 90 external 170
    Maximum path: 16
    Maximum hopcount 100
    Maximum metric variance 1

R57#
```

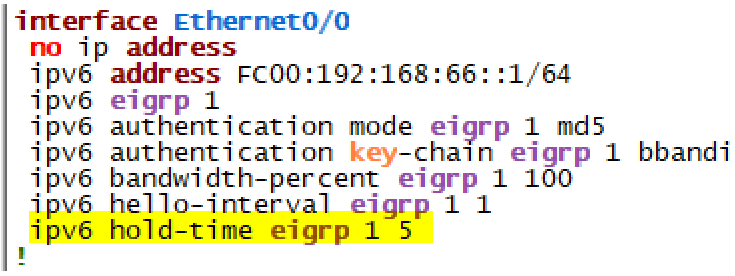


```
R50#show ipv6 eigrp timers
EIGRP-IPv6 Timers for AS(1)
  Hello Process
    Expiration    Type
|           2.572  (parent)
  |           2.572  Hello (Et0/2)

  Update Process
    Expiration    Type
|          12.765  (parent)
  |          12.765  (parent)
    |          12.765  Peer holding

  SIA Process
    Expiration    Type for Topo(base)
|           0.000  (parent)
R50#
```



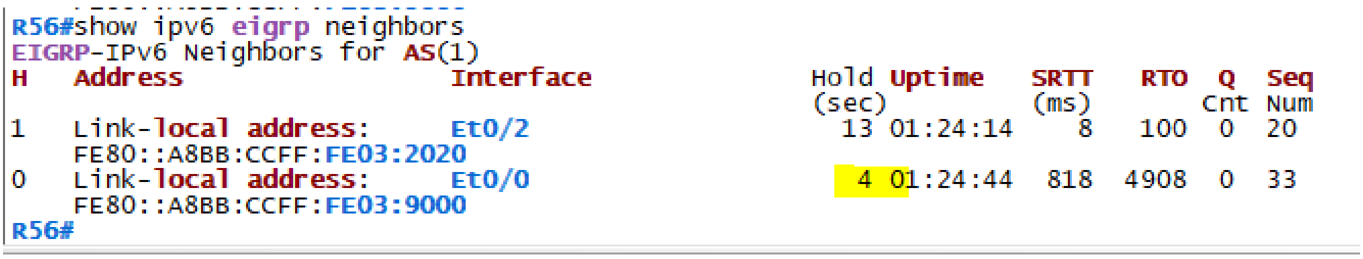
```
R57#show ipv6 eigrp topology FC00:10:56:1::/64
EIGRP-IPv6 Topology Entry for AS(1)/ID(7.7.7.7) for FC00:10:56:1::/64
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 409600
  Descriptor Blocks:
  FE80::A8BB:CCFF:FE03:8000 (Ethernet0/0), from FE80::A8BB:CCFF:FE03:8000, Send flag is 0x0
      Composite metric is (409600/128256), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 6000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
        Originating router is 6.6.6.6
R57#
```

Now have changed hold timer to 5 seconds.



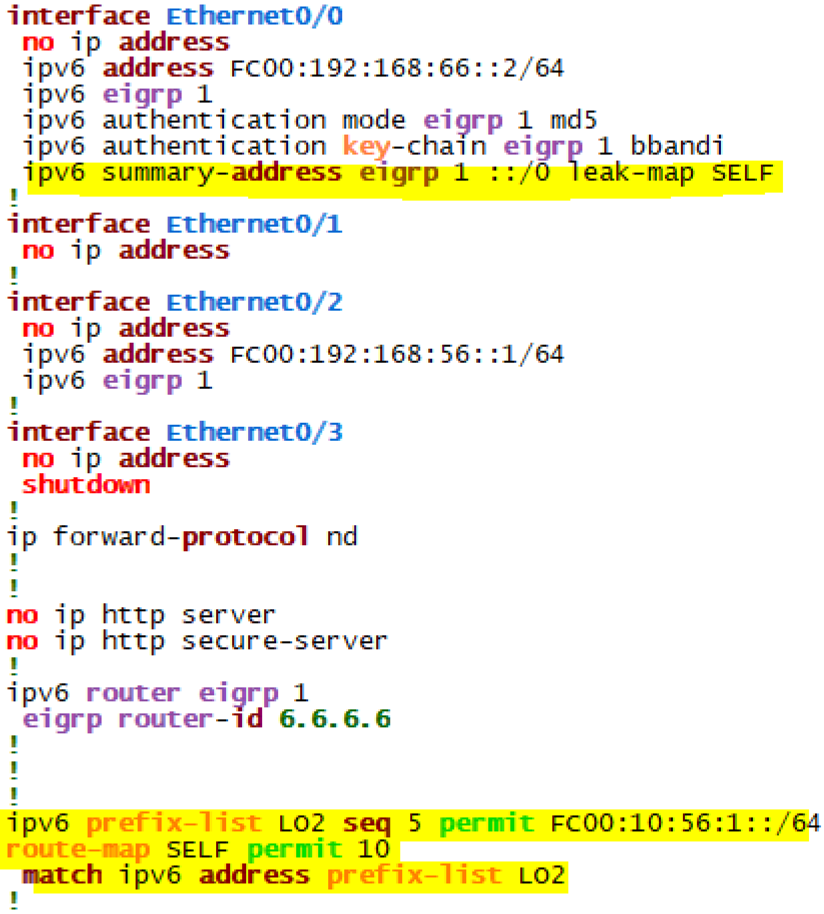
```
interface Ethernet0/0
 no ip address
 ipv6 address FC00:192:168:66::1/64
 ipv6 eigrp 1
 ipv6 authentication mode eigrp 1 md5
 ipv6 authentication key-chain eigrp 1 bbandi
 ipv6 bandwidth-percent eigrp 1 100
 ipv6 hello-interval eigrp 1 1
 ipv6 hold-time eigrp 1 5
!
```

Verify on R56



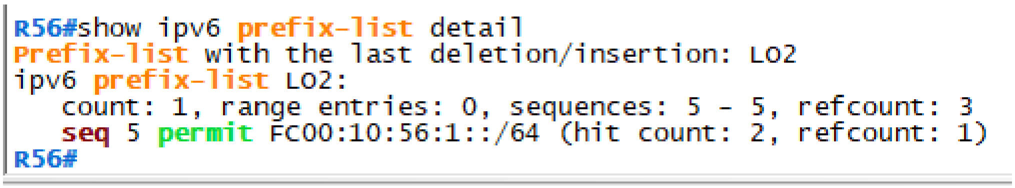
```
R56#show ipv6 eigrp neighbors
EIGRP-IPv6 Neighbors for AS(1)
H   Address                    Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                      (sec)         (ms)       Cnt Num
1   Link-local address:        Et0/2                    13 01:24:14    8   100  0  20
    FE80::A8BB:CCFF:FE03:2020
0   Link-local address:        Et0/0                     4 01:24:44  818  4908  0  33
    FE80::A8BB:CCFF:FE03:9000
R56#
```

Now we configure on R56 Default route and with Loopback0 leakmap



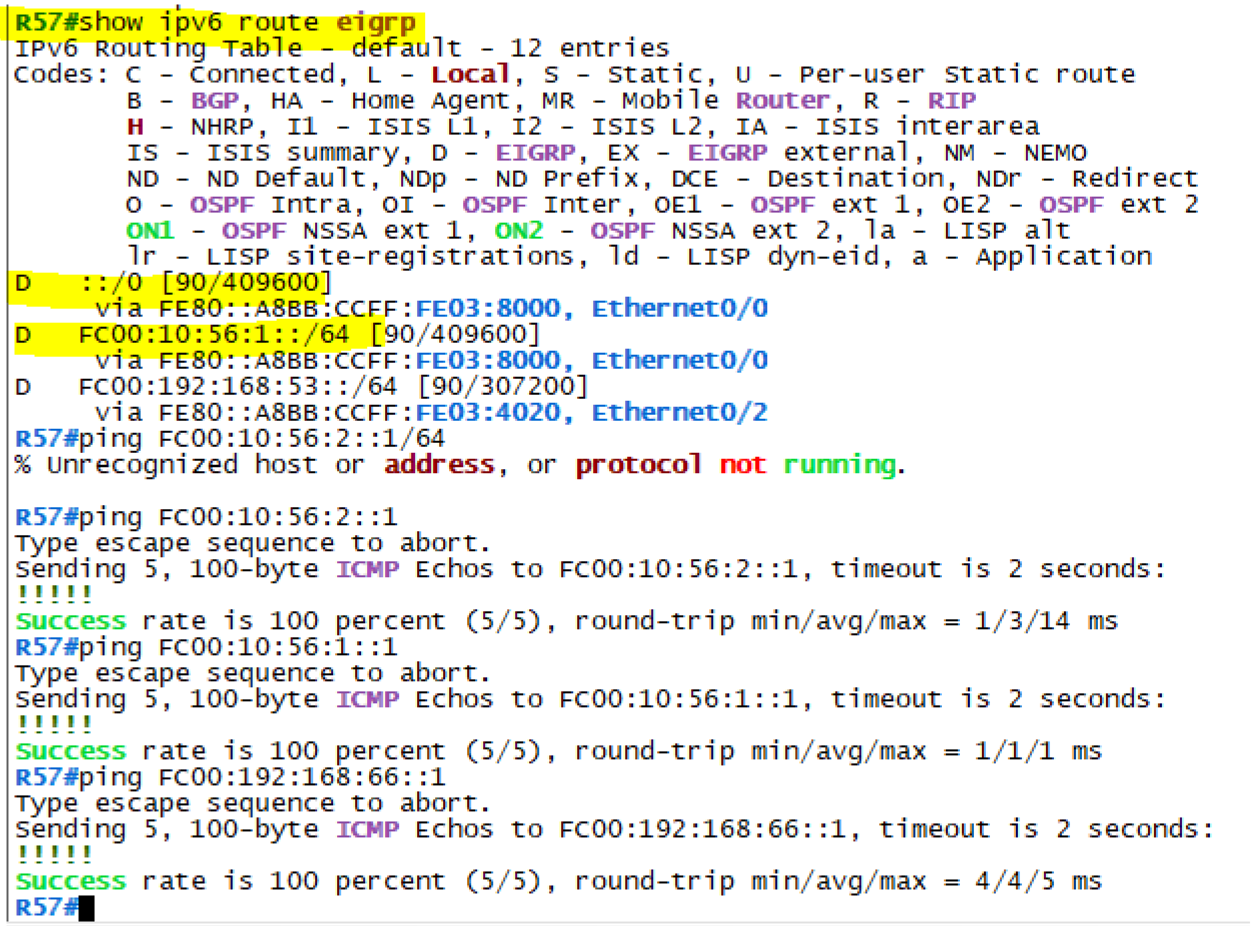
```
interface Ethernet0/0
 no ip address
 ipv6 address FC00:192:168:66::2/64
 ipv6 eigrp 1
 ipv6 authentication mode eigrp 1 md5
 ipv6 authentication key-chain eigrp 1 bbandi
 ipv6 summary-address eigrp 1 ::/0 leak-map SELF
!
interface Ethernet0/1
 no ip address
!
interface Ethernet0/2
 no ip address
 ipv6 address FC00:192:168:56::1/64
 ipv6 eigrp 1
!
interface Ethernet0/3
 no ip address
 shutdown
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
ipv6 router eigrp 1
 eigrp router-id 6.6.6.6
!
!
!
ipv6 prefix-list LO2 seq 5 permit FC00:10:56:1::/64
route-map SELF permit 10
 match ipv6 address prefix-list LO2
!
```

Verify the prefix-list hits



```
R56#show ipv6 prefix-list detail
Prefix-list with the last deletion/insertion: LO2
ipv6 prefix-list LO2:
   count: 1, range entries: 0, sequences: 5 - 5, refcount: 3
   seq 5 permit FC00:10:56:1::/64 (hit count: 2, refcount: 1)
R56#
```

R57 lets verify default route, Loopback 0 IP of R56 in leak map and verify pings

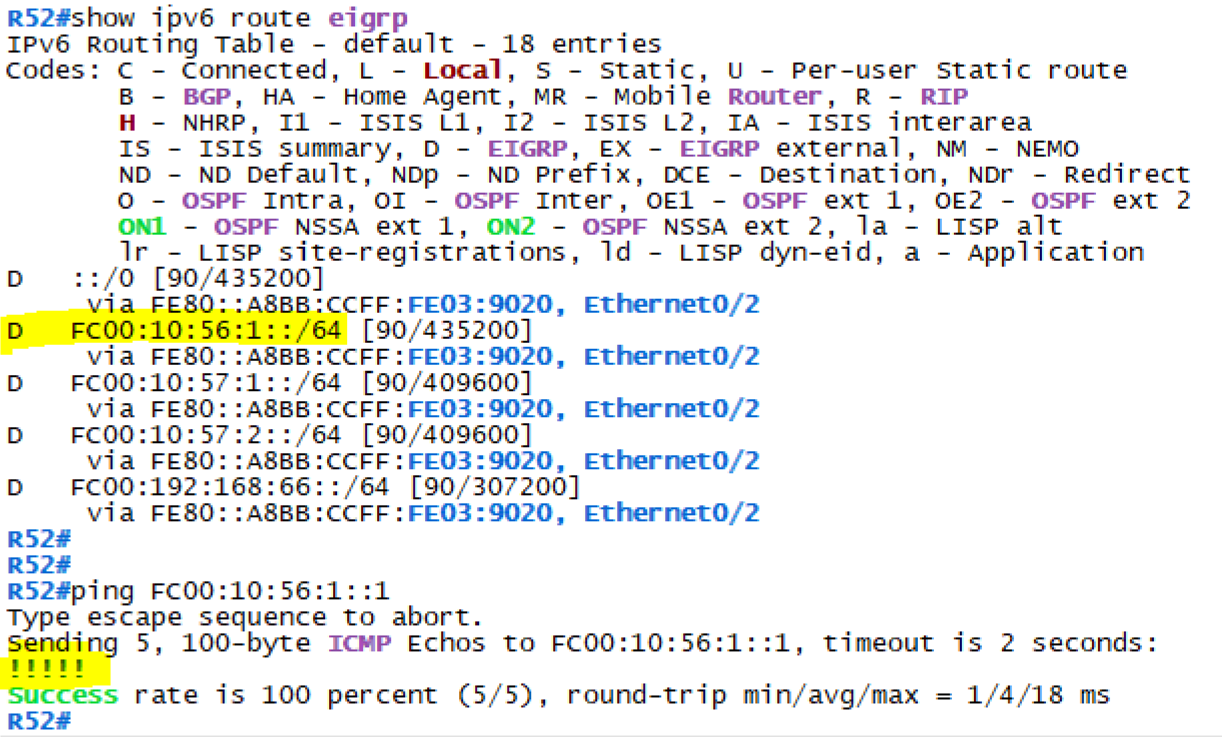


```
R57#show ipv6 route eigrp
IPv6 Routing Table - default - 12 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
D   ::/0 [90/409600]
     via FE80::A8BB:CCFF:FE03:8000, Ethernet0/0
D   FC00:10:56:1::/64 [90/409600]
     via FE80::A8BB:CCFF:FE03:8000, Ethernet0/0
D   FC00:192:168:53::/64 [90/307200]
     via FE80::A8BB:CCFF:FE03:4020, Ethernet0/2
R57#ping FC00:10:56:2::1/64
% Unrecognized host or address, or protocol not running.

R57#ping FC00:10:56:2::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:10:56:2::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/14 ms
R57#ping FC00:10:56:1::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:10:56:1::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R57#ping FC00:192:168:66::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:192:168:66::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/5 ms
R57#
```

Route Filter

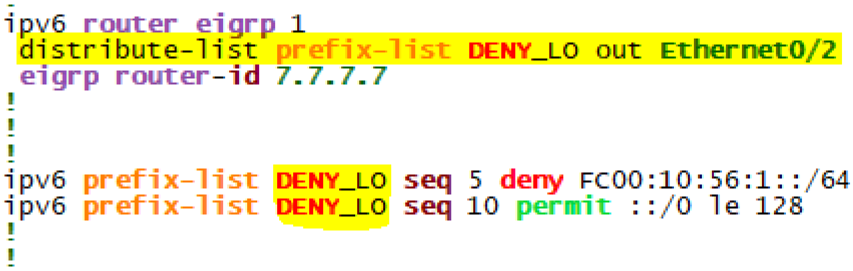
We configure on R57 so R52 no longer receives R56 Loopback0 address

Before configure lets verify on R52

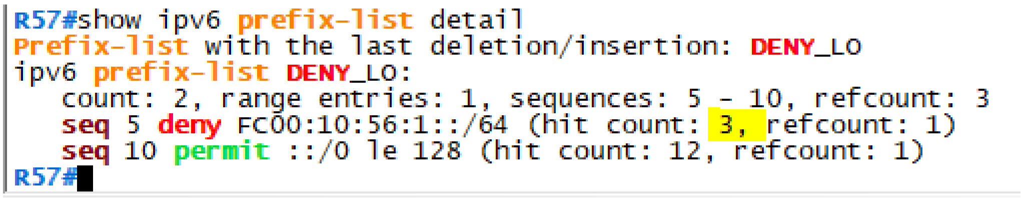


```
R52#show ipv6 route eigrp
IPv6 Routing Table - default - 18 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
D   ::/0 [90/435200]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
D   FC00:10:56:1::/64 [90/435200]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
D   FC00:10:57:1::/64 [90/409600]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
D   FC00:10:57:2::/64 [90/409600]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
D   FC00:192:168:66::/64 [90/307200]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
R52#
R52#
R52#ping FC00:10:56:1::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:10:56:1::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/18 ms
R52#
```

Now we configure prefix like on R57

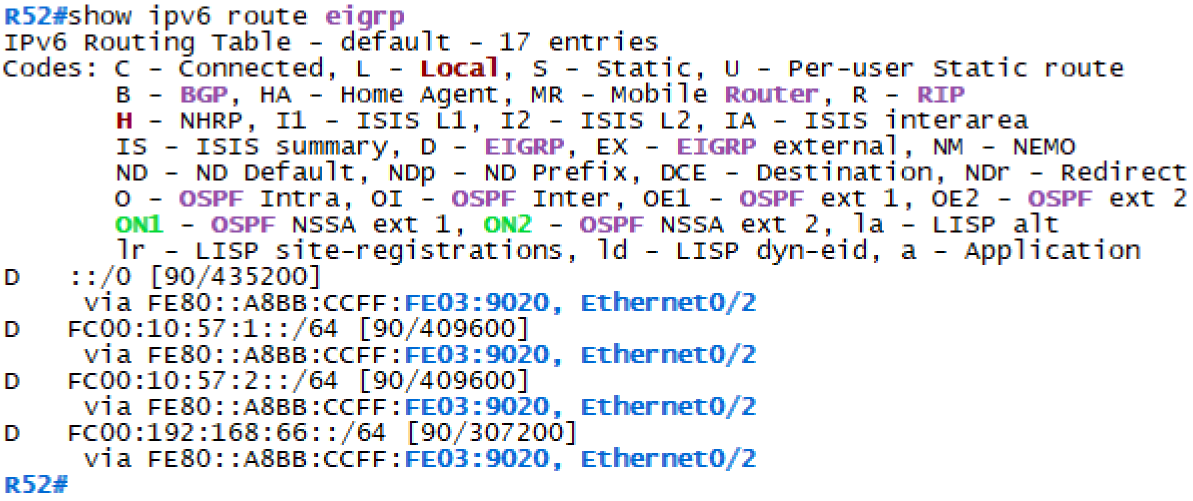


```
!
ipv6 router eigrp 1
 distribute-list prefix-list DENY_LO out Ethernet0/2
 eigrp router-id 7.7.7.7
!
!
!
ipv6 prefix-list DENY_LO seq 5 deny FC00:10:56:1::/64
ipv6 prefix-list DENY_LO seq 10 permit ::/0 le 128
!
!
```



```
R57#show ipv6 prefix-list detail
Prefix-list with the last deletion/insertion: DENY_LO
ipv6 prefix-list DENY_LO:
   count: 2, range entries: 1, sequences: 5 - 10, refcount: 3
   seq 5 deny FC00:10:56:1::/64 (hit count: 3, refcount: 1)
   seq 10 permit ::/0 le 128 (hit count: 12, refcount: 1)
R57#
```

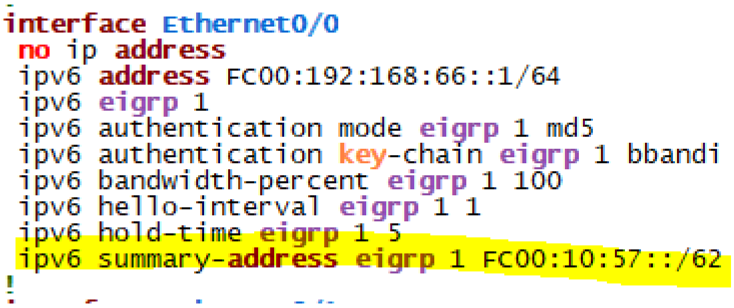
Lets verify the route in R52



```
R52#show ipv6 route eigrp
IPv6 Routing Table - default - 17 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
D   ::/0 [90/435200]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
D   FC00:10:57:1::/64 [90/409600]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
D   FC00:10:57:2::/64 [90/409600]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
D   FC00:192:168:66::/64 [90/307200]
     via FE80::A8BB:CCFF:FE03:9020, Ethernet0/2
R52#
```

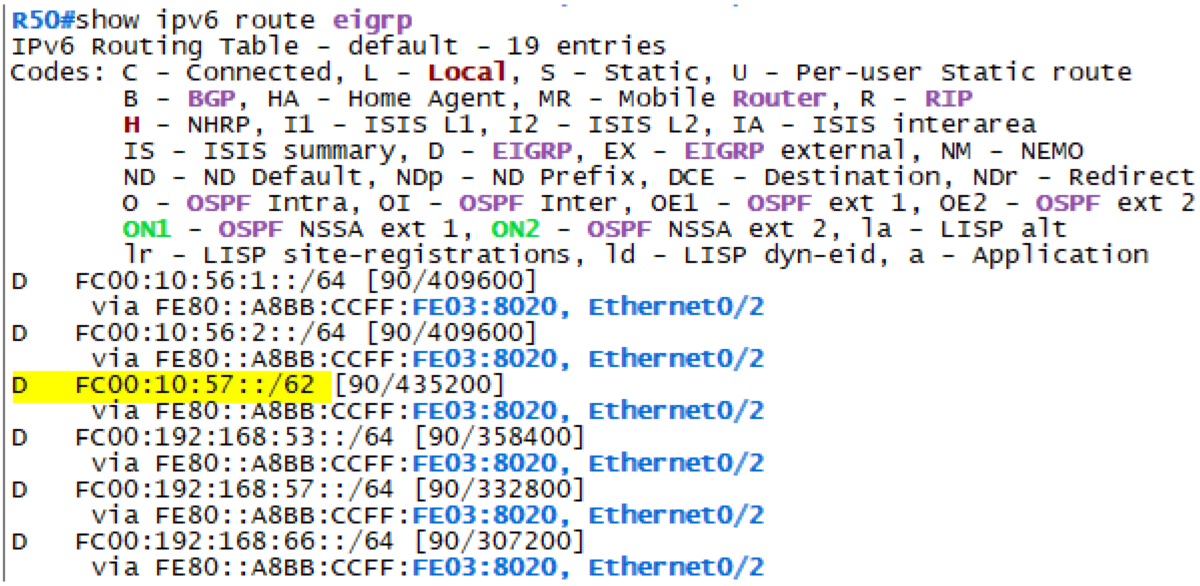
Summary

We will do summarisation for the R57 Loopback address and verify on R50



```
interface Ethernet0/0
 no ip address
 ipv6 address FC00:192:168:66::1/64
 ipv6 eigrp 1
 ipv6 authentication mode eigrp 1 md5
 ipv6 authentication key-chain eigrp 1 bbandi
 ipv6 bandwidth-percent eigrp 1 100
 ipv6 hello-interval eigrp 1 1
 ipv6 hold-time eigrp 1 5
 ipv6 summary-address eigrp 1 FC00:10:57::/62
!
```

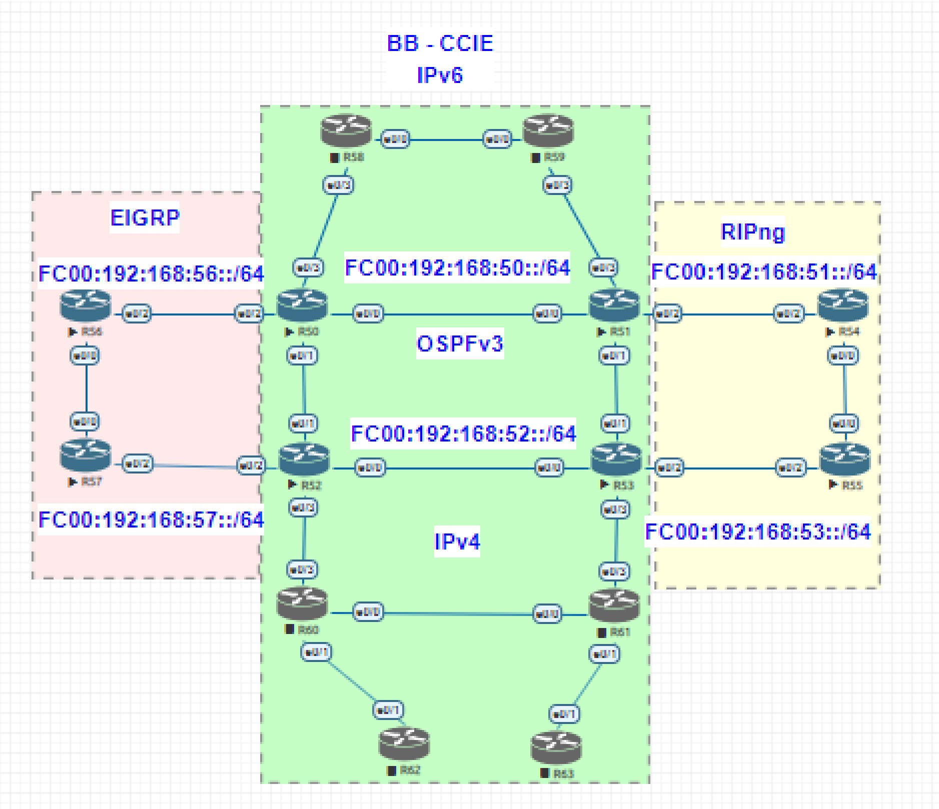
R50 Verification:



```
R50#show ipv6 route eigrp
IPv6 Routing Table - default - 19 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
D   FC00:10:56:1::/64 [90/409600]
     via FE80::A8BB:CCFF:FE03:8020, Ethernet0/2
D   FC00:10:56:2::/64 [90/409600]
     via FE80::A8BB:CCFF:FE03:8020, Ethernet0/2
D   FC00:10:57::/62 [90/435200]
     via FE80::A8BB:CCFF:FE03:8020, Ethernet0/2
D   FC00:192:168:53::/64 [90/358400]
     via FE80::A8BB:CCFF:FE03:8020, Ethernet0/2
D   FC00:192:168:57::/64 [90/332800]
     via FE80::A8BB:CCFF:FE03:8020, Ethernet0/2
D   FC00:192:168:66::/64 [90/307200]
     via FE80::A8BB:CCFF:FE03:8020, Ethernet0/2
```

IPv6 OSPFv3

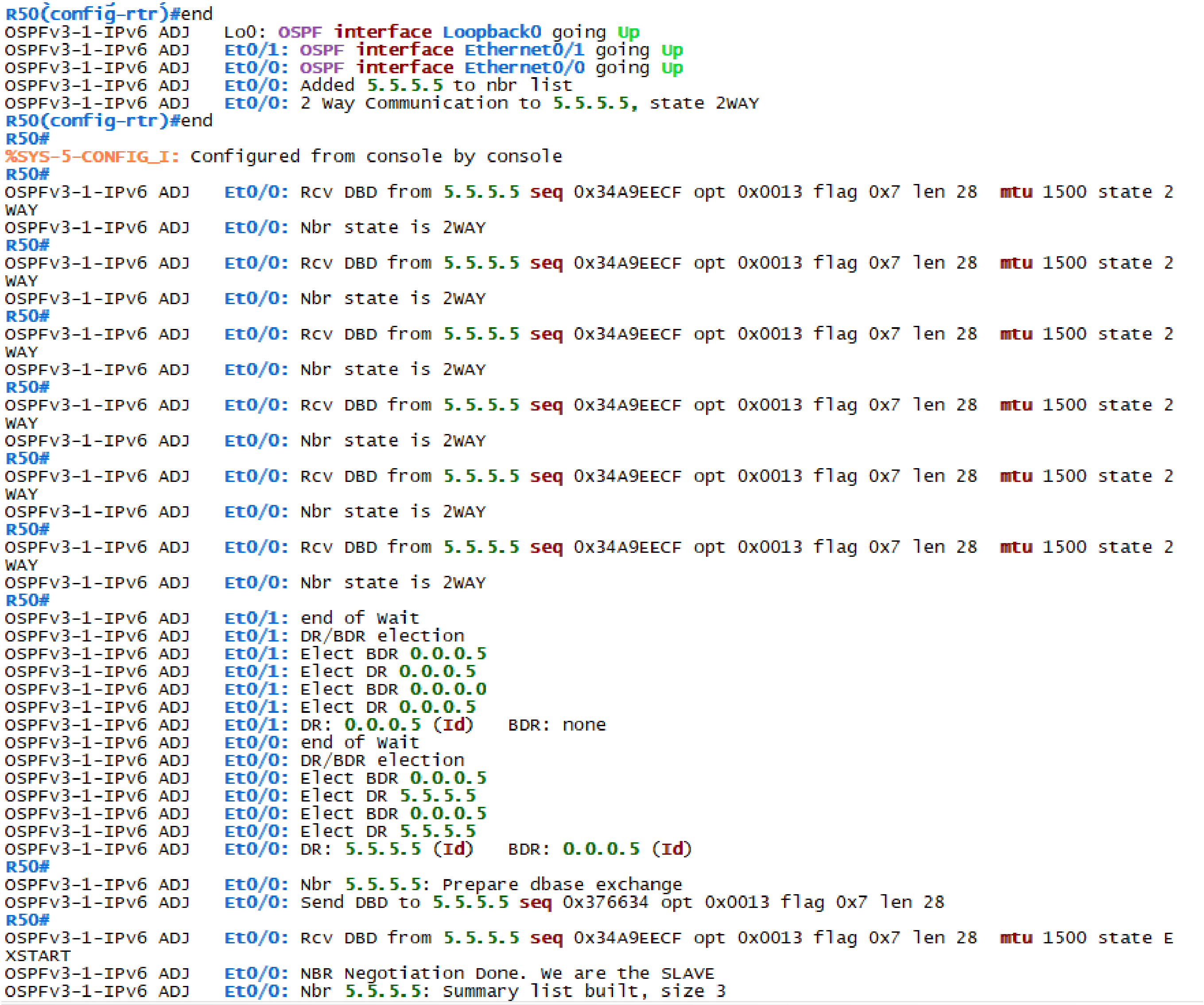
Basic Config enable OSPFv3 with the below topology.



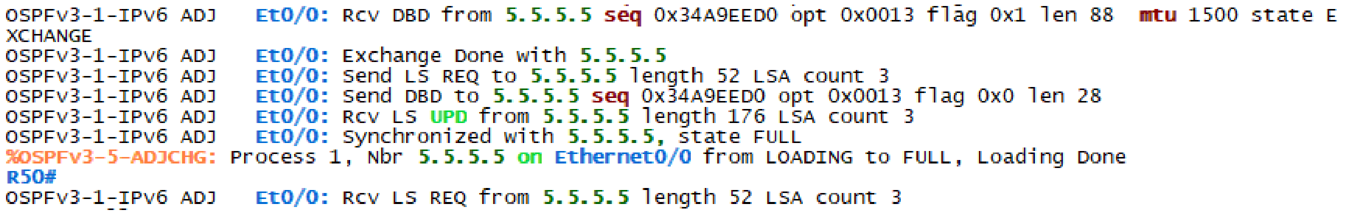
OSPF Basic config to enable IPv6

```
interface Ethernet0/1
 no ip address
 ipv6 address FC00:192:168:52::2/64
 ipv6 ospf 1 area 0
!
interface Ethernet0/2
 no ip address
 ipv6 address FC00:192:168:56::2/64
 ipv6 eigrp 1
!
interface Ethernet0/3
 no ip address
 ipv6 address FC00:192:168:58::2/64
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
ipv6 router eigrp 1
 passive-interface default
 no passive-interface Ethernet0/2
 eigrp router-id 0.0.0.1
!
ipv6 router ospf 1
 router-id 0.0.0.5
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/3
 no passive-interface Loopback0
 no passive-interface Loopback1
!
```

Below debug to form the OSPF neighbour

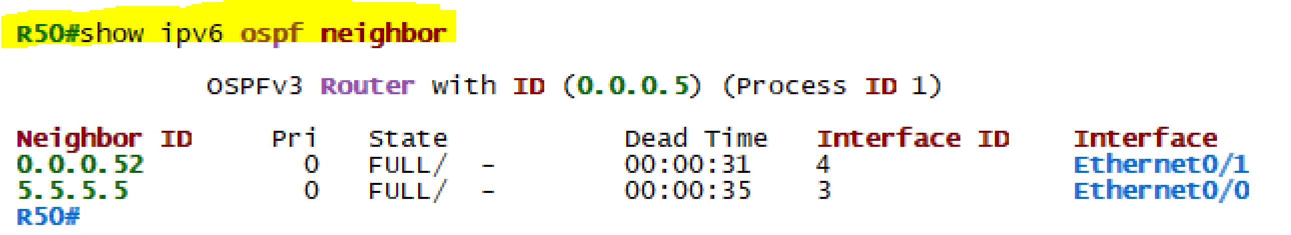


```
R50(config-rtr)#end
OSPFv3-1-IPv6 ADJ   Lo0: OSPF interface Loopback0 going Up
OSPFv3-1-IPv6 ADJ   Et0/1: OSPF interface Ethernet0/1 going Up
OSPFv3-1-IPv6 ADJ   Et0/0: OSPF interface Ethernet0/0 going Up
OSPFv3-1-IPv6 ADJ   Et0/0: Added 5.5.5.5 to nbr list
OSPFv3-1-IPv6 ADJ   Et0/0: 2 Way Communication to 5.5.5.5, state 2WAY
R50(config-rtr)#end
R50#
%SYS-5-CONFIG_I: Configured from console by console
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv DBD from 5.5.5.5 seq 0x34A9EECF opt 0x0013 flag 0x7 len 28  mtu 1500 state 2
WAY
OSPFv3-1-IPv6 ADJ   Et0/0: Nbr state is 2WAY
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv DBD from 5.5.5.5 seq 0x34A9EECF opt 0x0013 flag 0x7 len 28  mtu 1500 state 2
WAY
OSPFv3-1-IPv6 ADJ   Et0/0: Nbr state is 2WAY
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv DBD from 5.5.5.5 seq 0x34A9EECF opt 0x0013 flag 0x7 len 28  mtu 1500 state 2
WAY
OSPFv3-1-IPv6 ADJ   Et0/0: Nbr state is 2WAY
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv DBD from 5.5.5.5 seq 0x34A9EECF opt 0x0013 flag 0x7 len 28  mtu 1500 state 2
WAY
OSPFv3-1-IPv6 ADJ   Et0/0: Nbr state is 2WAY
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv DBD from 5.5.5.5 seq 0x34A9EECF opt 0x0013 flag 0x7 len 28  mtu 1500 state 2
WAY
OSPFv3-1-IPv6 ADJ   Et0/0: Nbr state is 2WAY
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv DBD from 5.5.5.5 seq 0x34A9EECF opt 0x0013 flag 0x7 len 28  mtu 1500 state 2
WAY
OSPFv3-1-IPv6 ADJ   Et0/0: Nbr state is 2WAY
R50#
OSPFv3-1-IPv6 ADJ   Et0/1: end of Wait
OSPFv3-1-IPv6 ADJ   Et0/1: DR/BDR election
OSPFv3-1-IPv6 ADJ   Et0/1: Elect BDR 0.0.0.5
OSPFv3-1-IPv6 ADJ   Et0/1: Elect DR 0.0.0.5
OSPFv3-1-IPv6 ADJ   Et0/1: Elect BDR 0.0.0.0
OSPFv3-1-IPv6 ADJ   Et0/1: Elect DR 0.0.0.5
OSPFv3-1-IPv6 ADJ   Et0/1: DR: 0.0.0.5 (Id)   BDR: none
OSPFv3-1-IPv6 ADJ   Et0/0: end of Wait
OSPFv3-1-IPv6 ADJ   Et0/0: DR/BDR election
OSPFv3-1-IPv6 ADJ   Et0/0: Elect BDR 0.0.0.5
OSPFv3-1-IPv6 ADJ   Et0/0: Elect DR 5.5.5.5
OSPFv3-1-IPv6 ADJ   Et0/0: Elect BDR 0.0.0.5
OSPFv3-1-IPv6 ADJ   Et0/0: Elect DR 5.5.5.5
OSPFv3-1-IPv6 ADJ   Et0/0: DR: 5.5.5.5 (Id)   BDR: 0.0.0.5 (Id)
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Nbr 5.5.5.5: Prepare dbase exchange
OSPFv3-1-IPv6 ADJ   Et0/0: Send DBD to 5.5.5.5 seq 0x376634 opt 0x0013 flag 0x7 len 28
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv DBD from 5.5.5.5 seq 0x34A9EECF opt 0x0013 flag 0x7 len 28  mtu 1500 state E
XSTART
OSPFv3-1-IPv6 ADJ   Et0/0: NBR Negotiation Done. We are the SLAVE
OSPFv3-1-IPv6 ADJ   Et0/0: Nbr 5.5.5.5: Summary list built, size 3
```



```
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv DBD from 5.5.5.5 seq 0x34A9EED0 opt 0x0013 flag 0x1 len 88  mtu 1500 state E
XCHANGE
OSPFv3-1-IPv6 ADJ   Et0/0: Exchange Done with 5.5.5.5
OSPFv3-1-IPv6 ADJ   Et0/0: Send LS REQ to 5.5.5.5 length 52 LSA count 3
OSPFv3-1-IPv6 ADJ   Et0/0: Send DBD to 5.5.5.5 seq 0x34A9EED0 opt 0x0013 flag 0x0 len 28
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv LS UPD from 5.5.5.5 length 176 LSA count 3
OSPFv3-1-IPv6 ADJ   Et0/0: Synchronized with 5.5.5.5, state FULL
%OSPFv3-5-ADJCHG: Process 1, Nbr 5.5.5.5 on Ethernet0/0 from LOADING to FULL, Loading Done
R50#
OSPFv3-1-IPv6 ADJ   Et0/0: Rcv LS REQ from 5.5.5.5 length 52 LSA count 3
```

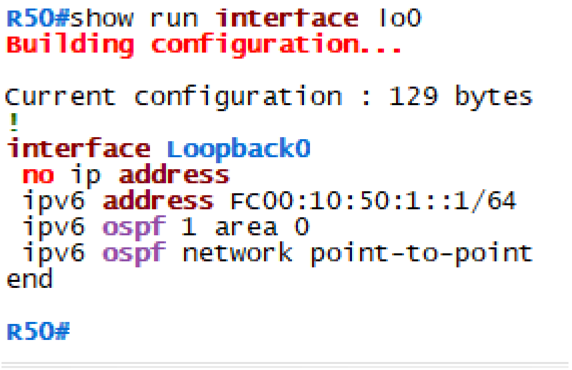
Verification for the OSPF neighbour.



```
R50#show ipv6 ospf neighbor

            OSPFv3 Router with ID (0.0.0.5) (Process ID 1)

Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
0.0.0.52          0   FULL/  -        00:00:31    4               Ethernet0/1
5.5.5.5           0   FULL/  -        00:00:35    3               Ethernet0/0
R50#
```

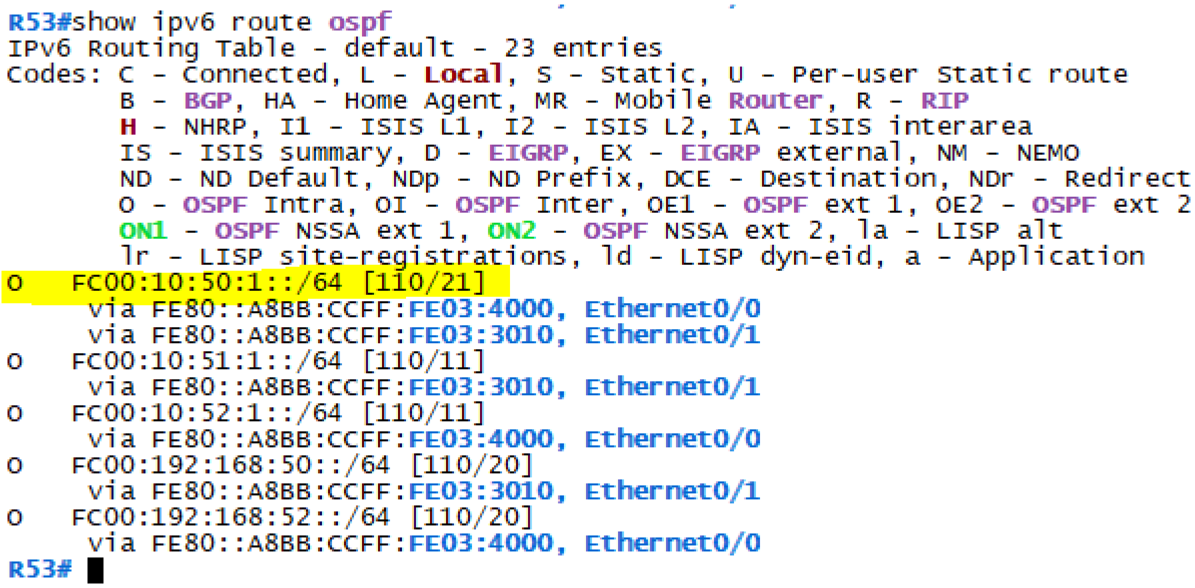
Route Filter

R53 we block the R50 Loopback address and verify



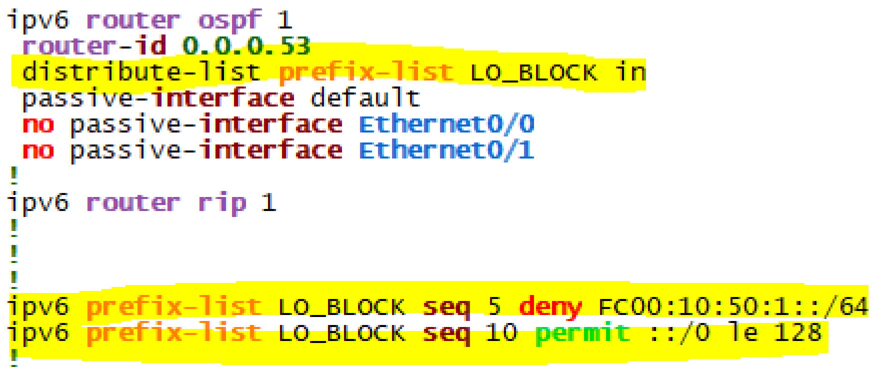
```
R50#show run interface lo0
Building configuration...

Current configuration : 129 bytes
!
interface Loopback0
 no ip address
 ipv6 address FC00:10:50:1::1/64
 ipv6 ospf 1 area 0
 ipv6 ospf network point-to-point
end

R50#
```

We verify the route before we apply route filter



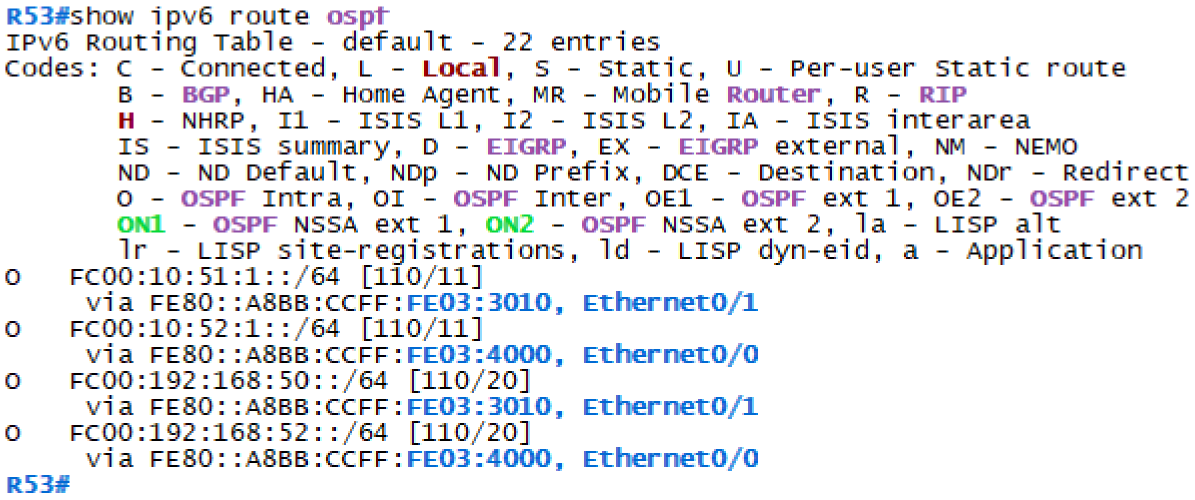
```
R53#show ipv6 route ospf
IPv6 Routing Table - default - 23 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
O   FC00:10:50:1::/64 [110/21]
     via FE80::A8BB:CCFF:FE03:4000, Ethernet0/0
     via FE80::A8BB:CCFF:FE03:3010, Ethernet0/1
O   FC00:10:51:1::/64 [110/11]
     via FE80::A8BB:CCFF:FE03:3010, Ethernet0/1
O   FC00:10:52:1::/64 [110/11]
     via FE80::A8BB:CCFF:FE03:4000, Ethernet0/0
O   FC00:192:168:50::/64 [110/20]
     via FE80::A8BB:CCFF:FE03:3010, Ethernet0/1
O   FC00:192:168:52::/64 [110/20]
     via FE80::A8BB:CCFF:FE03:4000, Ethernet0/0
R53#
```

Route filter config on R53

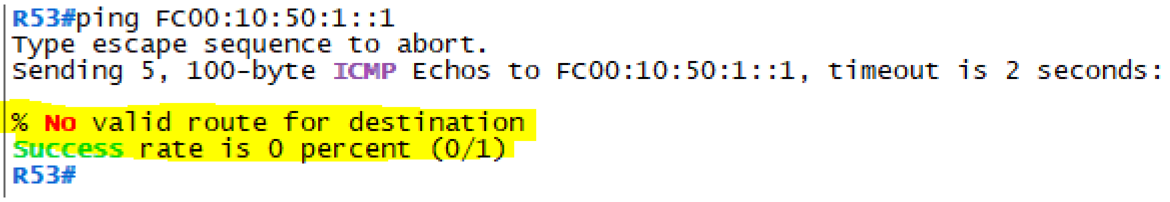


```
ipv6 router ospf 1
 router-id 0.0.0.53
 distribute-list prefix-list LO_BLOCK in
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
!
ipv6 router rip 1
!
!
!
ipv6 prefix-list LO_BLOCK seq 5 deny FC00:10:50:1::/64
ipv6 prefix-list LO_BLOCK seq 10 permit ::/0 le 128
!
```

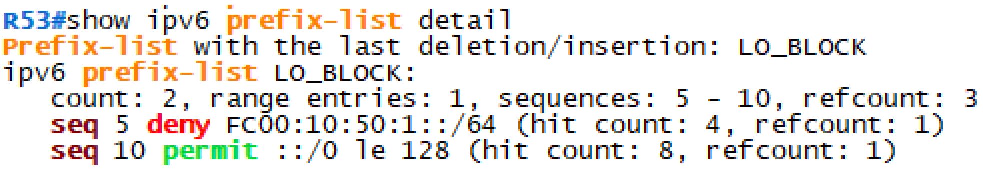
Lets verify now : (there is no route entry of R50 L0 IP address)



```
R53#show ipv6 route ospf
IPv6 Routing Table - default - 22 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
O   FC00:10:51:1::/64 [110/11]
     via FE80::A8BB:CCFF:FE03:3010, Ethernet0/1
O   FC00:10:52:1::/64 [110/11]
     via FE80::A8BB:CCFF:FE03:4000, Ethernet0/0
O   FC00:192:168:50::/64 [110/20]
     via FE80::A8BB:CCFF:FE03:3010, Ethernet0/1
O   FC00:192:168:52::/64 [110/20]
     via FE80::A8BB:CCFF:FE03:4000, Ethernet0/0
R53#
```



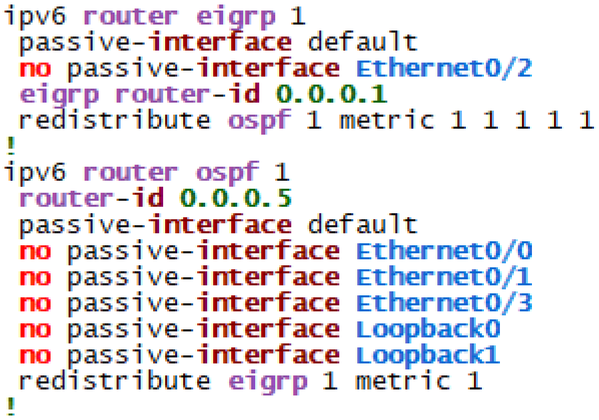
```
R53#ping FC00:10:50:1::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:10:50:1::1, timeout is 2 seconds:

% No valid route for destination
Success rate is 0 percent (0/1)
R53#
```



```
R53#show ipv6 prefix-list detail
Prefix-list with the last deletion/insertion: LO_BLOCK
ipv6 prefix-list LO_BLOCK:
   count: 2, range entries: 1, sequences: 5 - 10, refcount: 3
   seq 5 deny FC00:10:50:1::/64 (hit count: 4, refcount: 1)
   seq 10 permit ::/0 le 128 (hit count: 8, refcount: 1)
```

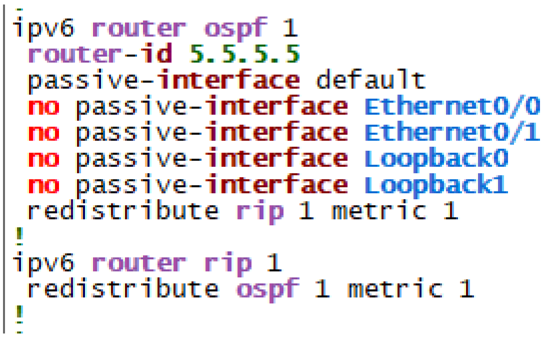
Redistribution: EIGRP to OSPF and OSPF to RIP on R50, R52 and R51,R53 verify the same.

R50



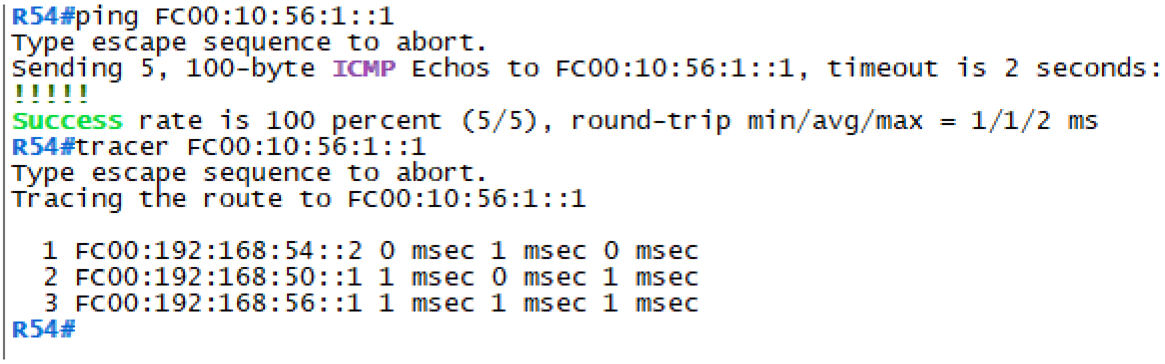
```
ipv6 router eigrp 1
 passive-interface default
 no passive-interface Ethernet0/2
 eigrp router-id 0.0.0.1
 redistribute ospf 1 metric 1 1 1 1 1
!
ipv6 router ospf 1
 router-id 0.0.0.5
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Ethernet0/3
 no passive-interface Loopback0
 no passive-interface Loopback1
 redistribute eigrp 1 metric 1
!
```

R51

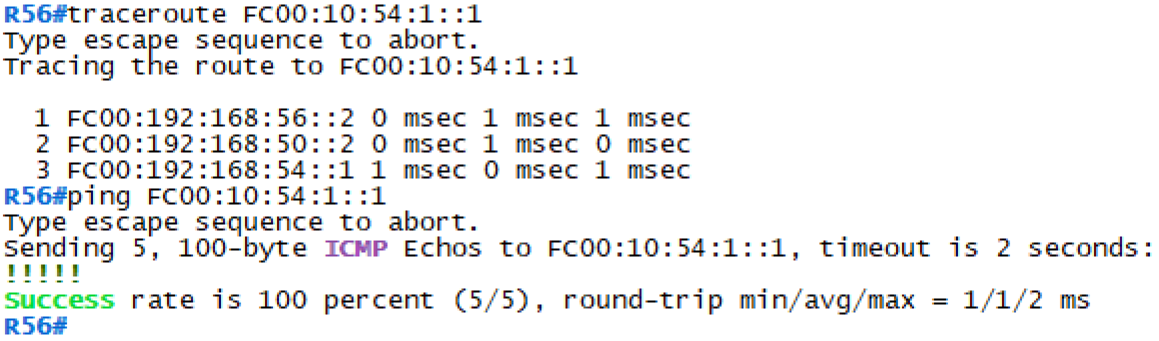


```
ipv6 router ospf 1
 router-id 5.5.5.5
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 no passive-interface Loopback0
 no passive-interface Loopback1
 redistribute rip 1 metric 1
!
ipv6 router rip 1
 redistribute ospf 1 metric 1
!
```

Check the router and reachability from R56 and R54

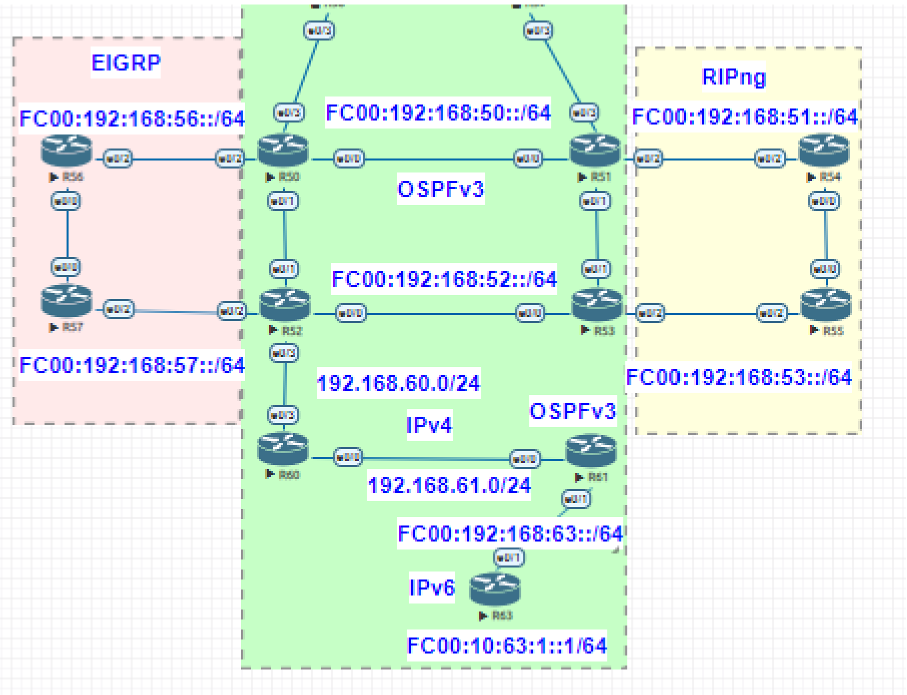


```
R54#ping FC00:10:56:1::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:10:56:1::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
R54#tracer FC00:10:56:1::1
Type escape sequence to abort.
Tracing the route to FC00:10:56:1::1

  1 FC00:192:168:54::2 0 msec 1 msec 0 msec
  2 FC00:192:168:50::1 1 msec 0 msec 1 msec
  3 FC00:192:168:56::1 1 msec 1 msec 1 msec
R54#
```

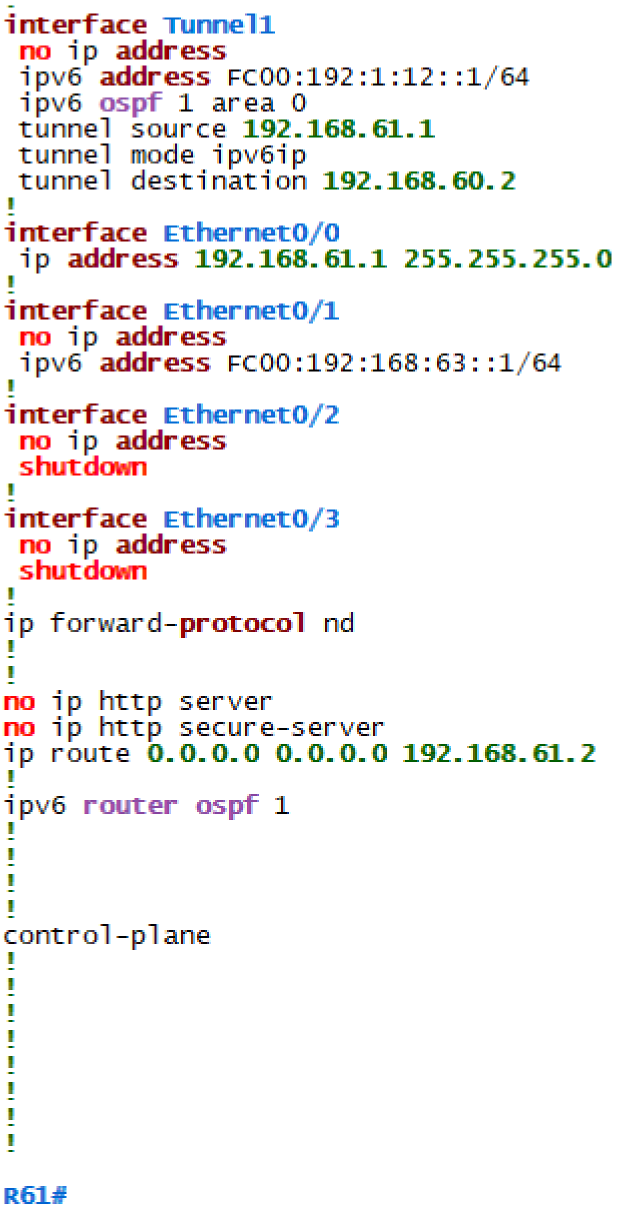


```
R56#traceroute FC00:10:54:1::1
Type escape sequence to abort.
Tracing the route to FC00:10:54:1::1

  1 FC00:192:168:56::2 0 msec 1 msec 1 msec
  2 FC00:192:168:50::2 0 msec 1 msec 0 msec
  3 FC00:192:168:54::1 1 msec 0 msec 1 msec
R56#ping FC00:10:54:1::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:10:54:1::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
R56#
```

IPv6 tunnel over IPv4 network

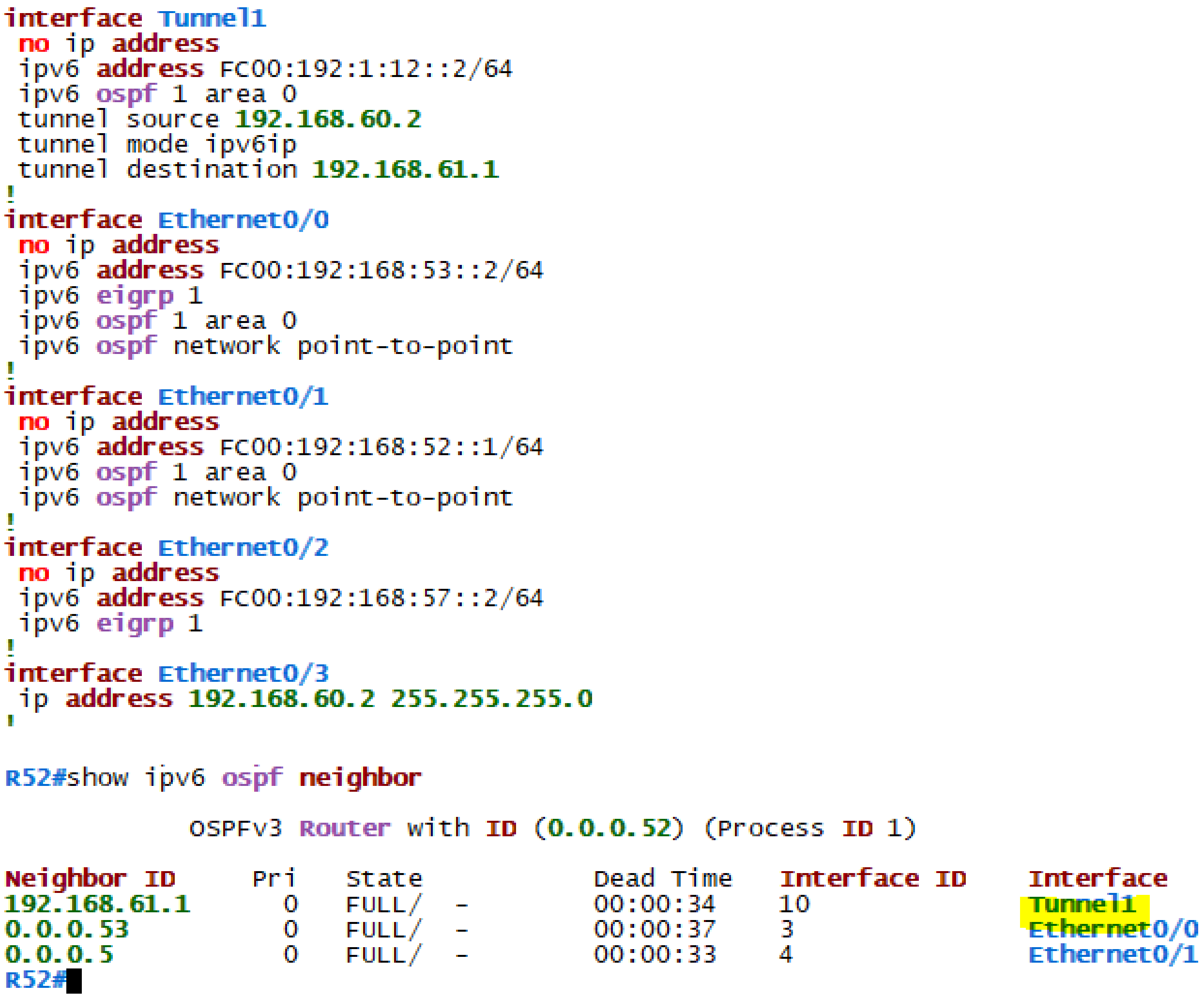


R52, R60, R61 is IPv4 network, we make tunnel between R61 and R52 , and verify routes



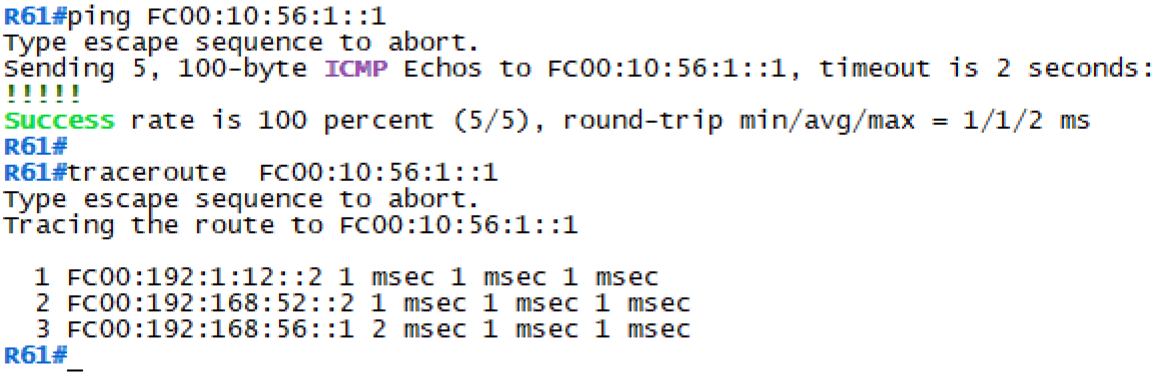
```
!
interface Tunnel1
 no ip address
 ipv6 address FC00:192:1:12::1/64
 ipv6 ospf 1 area 0
 tunnel source 192.168.61.1
 tunnel mode ipv6ip
 tunnel destination 192.168.60.2
!
interface Ethernet0/0
 ip address 192.168.61.1 255.255.255.0
!
interface Ethernet0/1
 no ip address
 ipv6 address FC00:192:168:63::1/64
!
interface Ethernet0/2
 no ip address
 shutdown
!
interface Ethernet0/3
 no ip address
 shutdown
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
ip route 0.0.0.0 0.0.0.0 192.168.61.2
!
ipv6 router ospf 1
!
!
!
!
control-plane
!
!
!
!
!
!
!
!

R61#
```

R52



```
interface Tunnel1
 no ip address
 ipv6 address FC00:192:1:12::2/64
 ipv6 ospf 1 area 0
 tunnel source 192.168.60.2
 tunnel mode ipv6ip
 tunnel destination 192.168.61.1
!
interface Ethernet0/0
 no ip address
 ipv6 address FC00:192:168:53::2/64
 ipv6 eigrp 1
 ipv6 ospf 1 area 0
 ipv6 ospf network point-to-point
!
interface Ethernet0/1
 no ip address
 ipv6 address FC00:192:168:52::1/64
 ipv6 ospf 1 area 0
 ipv6 ospf network point-to-point
!
interface Ethernet0/2
 no ip address
 ipv6 address FC00:192:168:57::2/64
 ipv6 eigrp 1
!
interface Ethernet0/3
 ip address 192.168.60.2 255.255.255.0
!

R52#show ipv6 ospf neighbor

           OSPFv3 Router with ID (0.0.0.52) (Process ID 1)

Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
192.168.61.1      0   FULL/  -        00:00:34    10              Tunnel1
0.0.0.53          0   FULL/  -        00:00:37    3               Ethernet0/0
0.0.0.5           0   FULL/  -        00:00:33    4               Ethernet0/1
R52#
```

Lets see reachability from R61 to R56



```
R61#ping FC00:10:56:1::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FC00:10:56:1::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
R61#
R61#traceroute  FC00:10:56:1::1
Type escape sequence to abort.
Tracing the route to FC00:10:56:1::1

  1 FC00:192:1:12::2 1 msec 1 msec 1 msec
  2 FC00:192:168:52::2 1 msec 1 msec 1 msec
  3 FC00:192:168:56::1 2 msec 1 msec 1 msec
R61#_
```

2.7 BGP

Border Gateway Protocol (BGP) is an Internet Engineering Task Force (IETF) standard, and the most scalable of all routing protocols. BGP is the routing protocol of the global Internet, as well as for Service Provider private networks. BGP has expanded upon its original purpose of carrying Internet reachability information, and can now carry routes for Multicast, IPv6, VPNs, and a variety of other data.

- BGP is the routing protocol used to exchange routing information between networks. It Is the largest routing protocol in the internet.

BGP Comes in two flavours :

- External BGP (EBGP)
- Internal BGP (IBGP)
- BGP is Currently in version 4,
- Runs over TCP(Port No. 179)
- Path vector protocol ,CIDR support

By Default BGP finds the best path to a network by using the best AS –Path.
BGP advertises the complete path to the advertised network. Path is sent as a LIST OF AS Number Which avoids loop.

BGP General Operations

- BGP Neighbours are manually configured
- Learns multiple paths via internal and external BGP speakers
- Picks the best path and installs in the forwarding table

EBGP (External BGP)

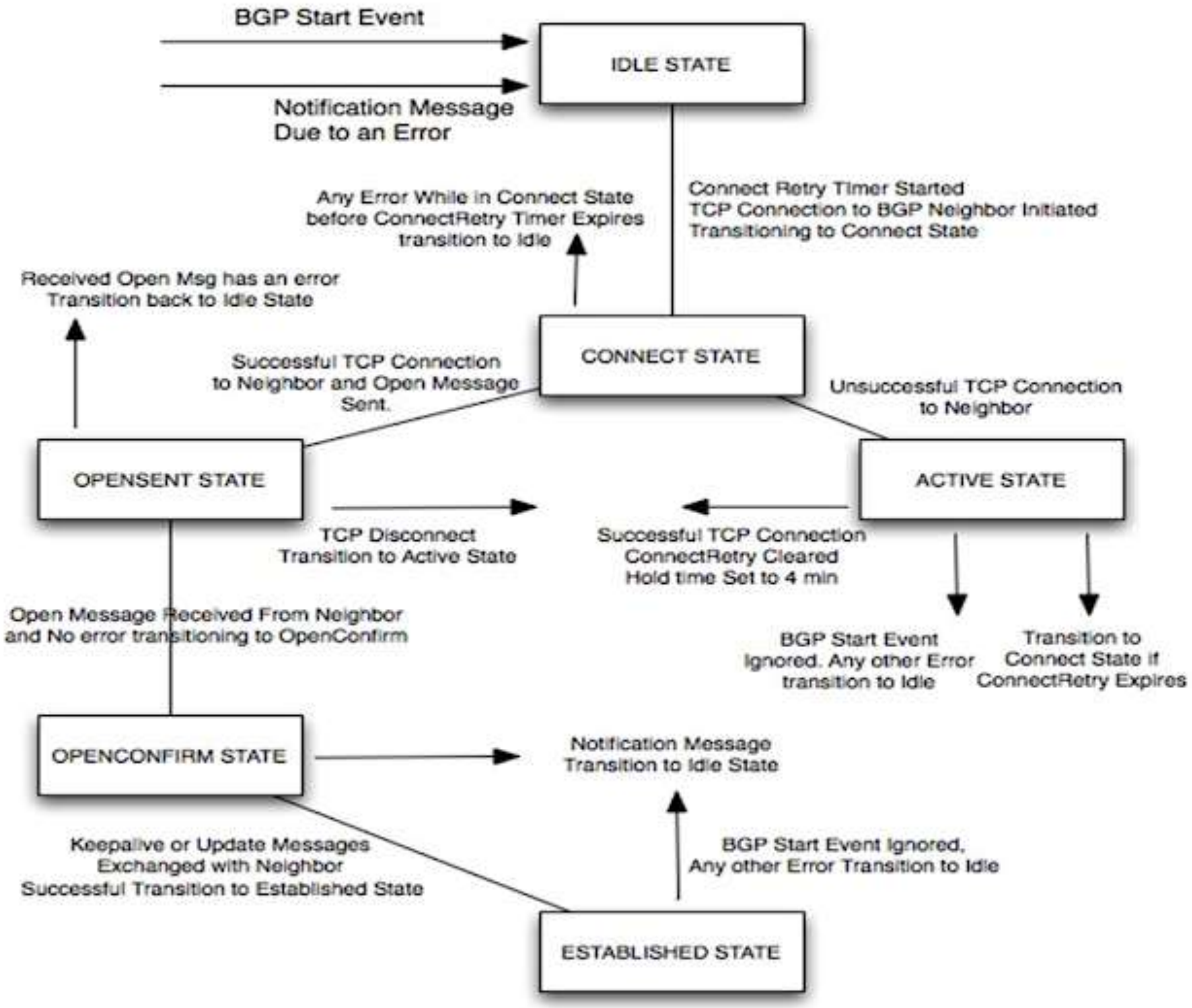
- BGP speakers in different AS
- Do not run an IGP between eBGP peers

IBGP (Internal BGP)

- BGP peer within the same AS
- iBGP speakers need to be fully meshed
- They do not pass on prefixes learned from other iBGP speakers to other iBGP peer . (The Rule of Split Horizon.)

BGP Message Type

Open: Establishes a peering session
Keep Alive: Handshake at regular intervals to maintain peering session.
Notification: Closes a peering session
Update: Advertises new routes or withdraws previously announced routes.
Each announced route is specified as a network prefix with attribute values.


BGP Start Event
Notification Message Due to an Error
IDLE STATE
Any Error While in Connect State before ConnectRetry Timer Expires transition to Idle
Connect Retry Timer Started TCP Connection to BGP Neighbor Initiated Transitioning to Connect State
Received Open Msg has an error Transition back to Idle State
CONNECT STATE
Successful TCP Connection to Neighbor and Open Message Sent.
Unsuccessful TCP Connection to Neighbor
OPENSENT STATE
ACTIVE STATE
TCP Disconnect Transition to Active State
Successful TCP Connection ConnectRetry Cleared Hold time Set to 4 min
Open Message Received From Neighbor and No error transitioning to OpenConfirm
BGP Start Event Ignored. Any other Error transition to Idle
Transition to Connect State if ConnectRetry Expires
OPENCONFIRM STATE
Notification Message Transition to Idle State
Keepalive or Update Messages Exchanged with Neighbor Successful Transition to Established State
BGP Start Event Ignored, Any other Error Transition to Idle
ESTABLISHED STATE

2.7.a Describe, implement and troubleshoot peer relationships

2.7.a [i] Peer-group, template

Concepts of peer group and peer template on Cisco routers that would help you making your configuration process more efficient as well as optimizing the process of route advertisement. You will be learning the similarities and differences between the two methods

2.7.a [ii] Active, passive

When a BGP speaker is configured as active, it may end up on either the active or passive side of the connection that eventually gets established. Once the TCP connection is completed, it doesn't matter which end was active and which was passive. The only difference is in which side of the TCP connection has port number 179.

2.7.a [iii] States, timers

2.7.a [iv] Dynamic neighbors

2.7.b Implement and troubleshoot IBGP and EBGP

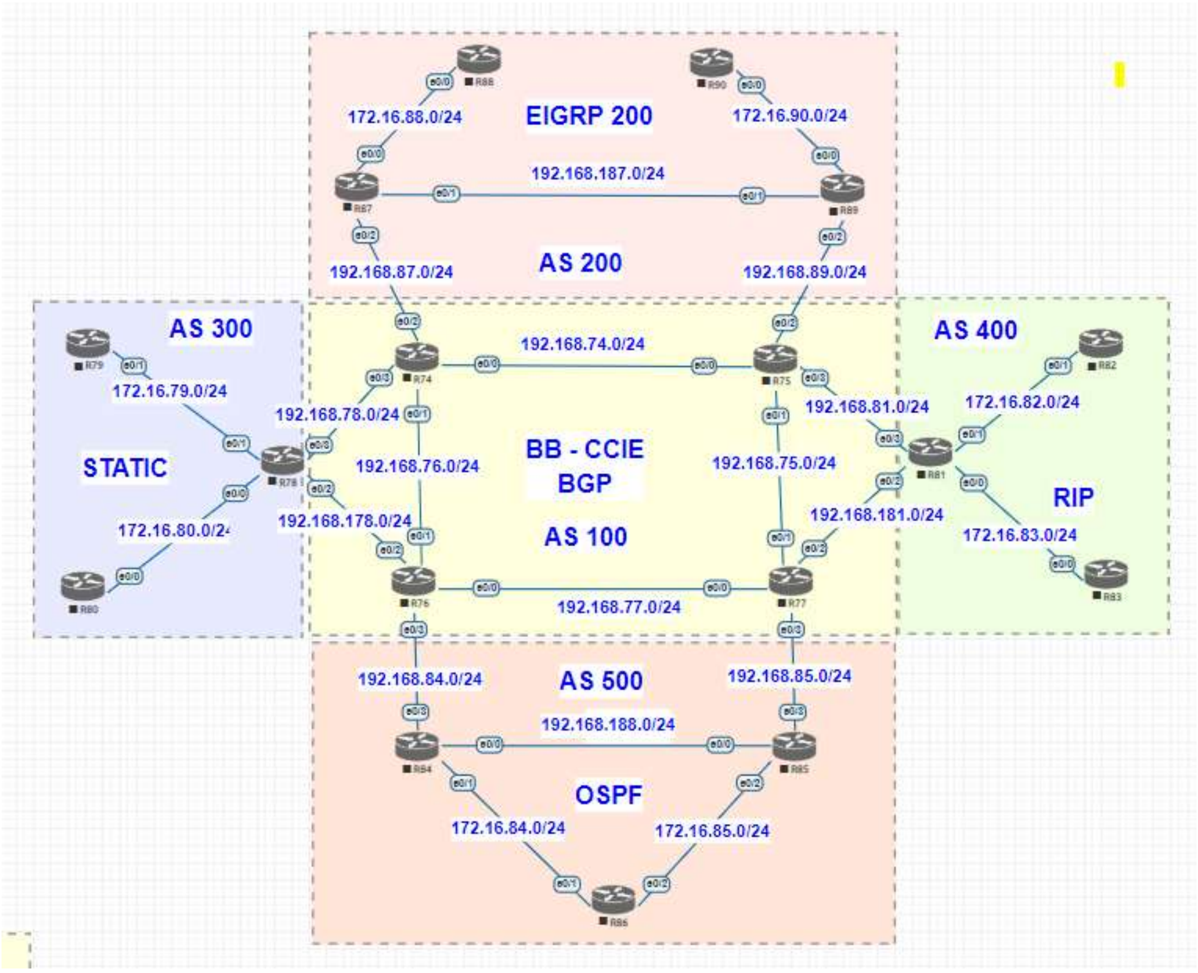
2.7.b [i] EBGP, IBGP

Covered above

2.7.b [ii] 4 bytes AS number

2.7.b [iii] Private AS

There’s recently been a change in the format that the AS number use:
Original 2-byte field
-Values 0 – 65535
-**Public ASNs 1 – 64511**
- **Private ASNs 64512 – 65535 (1024 addresses) Similar to RFC 1918 IP address**
Currently 4- byte field



2.7.c Explain attributes and best-path selection

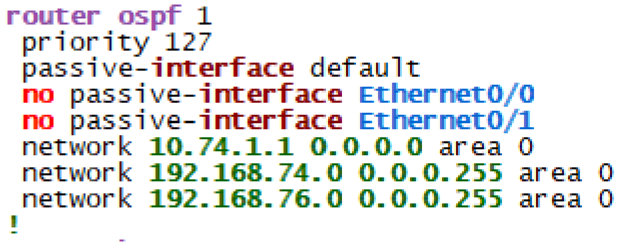
2.7.d Implement, optimize and troubleshoot routing policies

2.7.d [i] Attribute manipulation

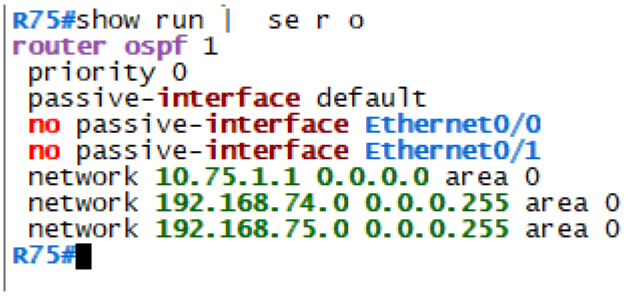
2.7.d [ii] Conditional advertisement

2.7.d [iii] Outbound route filtering
2.7.d [iv] Communities, extended communities
2.7.d [v] Multi-homing

Now we configured R74, R75, R76, R77 Full mesh BGP, underlay IGP using OSPF
(below config for R74 same will be rest of the router – changed router IP and network statement)

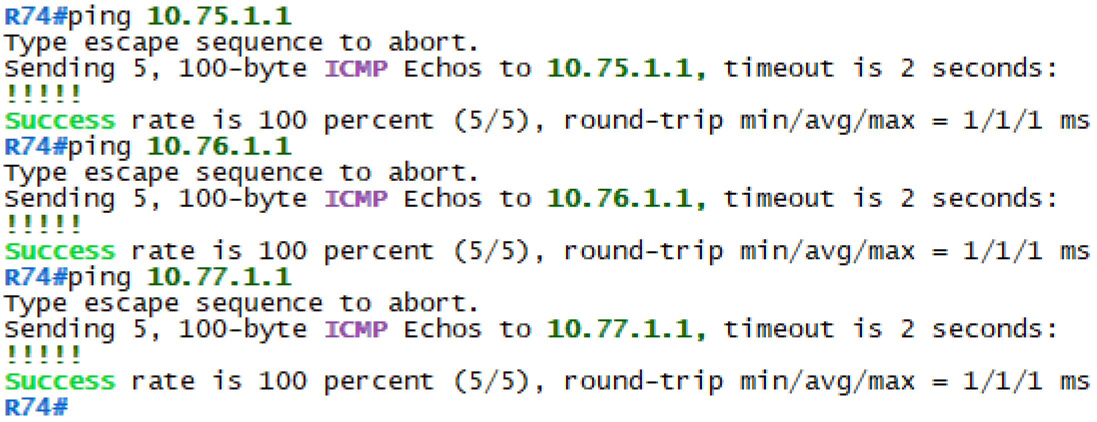


```
router ospf 1
 priority 127
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 network 10.74.1.1 0.0.0.0 area 0
 network 192.168.74.0 0.0.0.255 area 0
 network 192.168.76.0 0.0.0.255 area 0
!
```



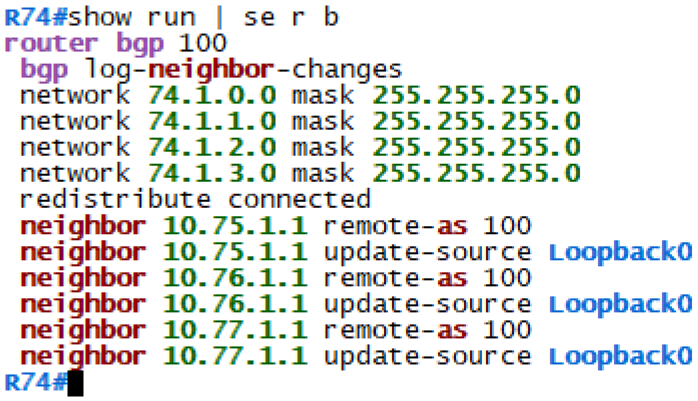
```
R75#show run |  se r o
router ospf 1
 priority 0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 network 10.75.1.1 0.0.0.0 area 0
 network 192.168.74.0 0.0.0.255 area 0
 network 192.168.75.0 0.0.0.255 area 0
R75#
```

Check the reachability to loopback address since we going to use loopback 0 for BGP

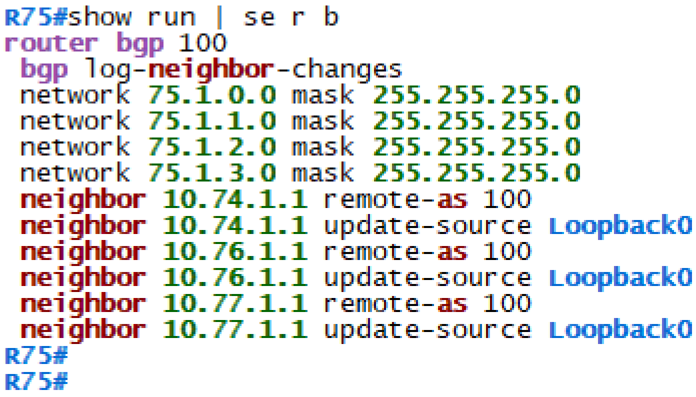


```
R74#ping 10.75.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.75.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R74#ping 10.76.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.76.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R74#ping 10.77.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.77.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R74#
```

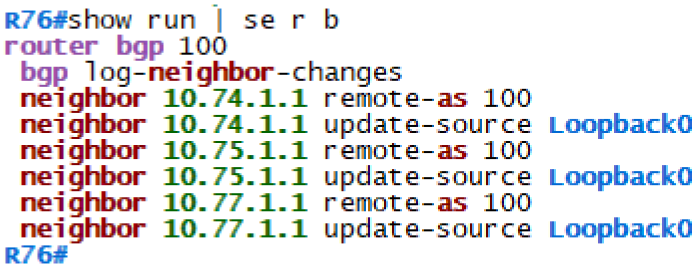
We configure BGP Full mesh



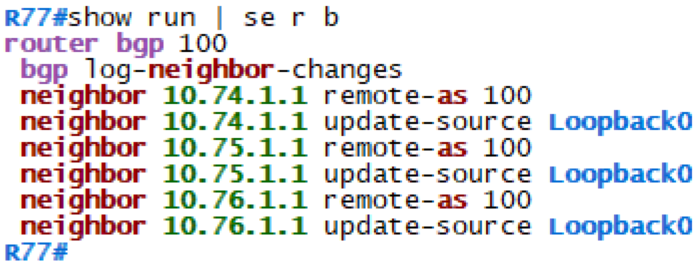
```
R74#show run | se r b
router bgp 100
 bgp log-neighbor-changes
 network 74.1.0.0 mask 255.255.255.0
 network 74.1.1.0 mask 255.255.255.0
 network 74.1.2.0 mask 255.255.255.0
 network 74.1.3.0 mask 255.255.255.0
 redistribute connected
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.76.1.1 remote-as 100
 neighbor 10.76.1.1 update-source Loopback0
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
R74#
```



```
R75#show run | se r b
router bgp 100
 bgp log-neighbor-changes
 network 75.1.0.0 mask 255.255.255.0
 network 75.1.1.0 mask 255.255.255.0
 network 75.1.2.0 mask 255.255.255.0
 network 75.1.3.0 mask 255.255.255.0
 neighbor 10.74.1.1 remote-as 100
 neighbor 10.74.1.1 update-source Loopback0
 neighbor 10.76.1.1 remote-as 100
 neighbor 10.76.1.1 update-source Loopback0
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
R75#
R75#
```



```
R76#show run | se r b
router bgp 100
 bgp log-neighbor-changes
 neighbor 10.74.1.1 remote-as 100
 neighbor 10.74.1.1 update-source Loopback0
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
R76#
```



```
R77#show run | se r b
router bgp 100
 bgp log-neighbor-changes
 neighbor 10.74.1.1 remote-as 100
 neighbor 10.74.1.1 update-source Loopback0
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.76.1.1 remote-as 100
 neighbor 10.76.1.1 update-source Loopback0
R77#
```

Now we should see Full mesh BGP neighbourship

```
R74#show ip bgp summary
BGP router identifier 10.74.1.1, local AS number 100
BGP table version is 34, main routing table version 34
13 network entries using 1820 bytes of memory
13 path entries using 1040 bytes of memory
3/3 BGP path/bestpath attribute entries using 432 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 3292 total bytes of memory
BGP activity 17/4 prefixes, 17/4 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.75.1.1       4          100    9437    9446       34    0    0 5d22h           4
10.76.1.1       4          100    9429    9434       34    0    0 5d22h           0
10.77.1.1       4          100      45      59       34    0    0 00:32:53        0
R74#
```

BGP Transport:

BGP uses TCP port 179 for transport

-implies the BGP needs IGP First.

BGP Neighbor statement tell process to….

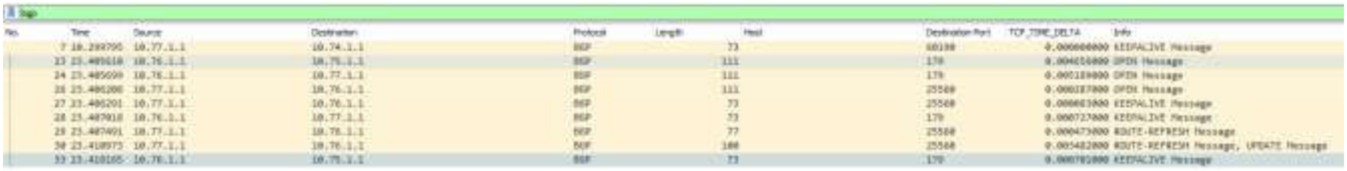
-listen for remote address via TCP 179

-initiate a session to remote address via TCP 179

-if collision, higher router-id becomes TCP client. Can happen if client and server try to establish at same time Normally the client will initiate the session over port 179.

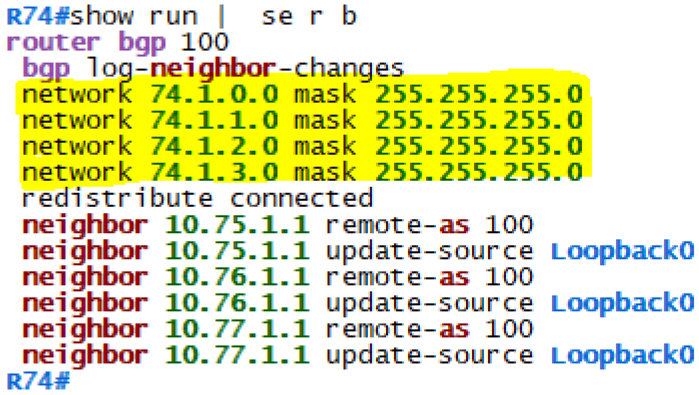
```
R74#show tcp brief
TCB       Local Address             Foreign Address           (state)
C5478A48  10.74.1.1.60190           10.77.1.1.179             ESTAB
C602CAD0  10.74.1.1.19322           10.75.1.1.179             ESTAB
C6029D70  10.74.1.1.44819           10.76.1.1.179             ESTAB
R74#
```

Here is the BGP handshake with wireshark



No.	Time	Source	Destination	Protocol	Length	Host	Destination Port	TCP_TIME_DELTA	Info
7	10.289795	10.77.1.1	10.74.1.1	BGP	73		60190	0.000000000	KEEPALIVE Message
23	23.405610	10.76.1.1	10.75.1.1	BGP	111		179	0.004016000	OPEN Message
24	23.405690	10.76.1.1	10.77.1.1	BGP	111		179	0.005189000	OPEN Message
26	23.406200	10.77.1.1	10.76.1.1	BGP	111		25560	0.000187000	OPEN Message
27	23.406201	10.77.1.1	10.76.1.1	BGP	73		25560	0.000083000	KEEPALIVE Message
28	23.407018	10.76.1.1	10.77.1.1	BGP	73		179	0.000727000	KEEPALIVE Message
29	23.407491	10.77.1.1	10.76.1.1	BGP	77		25560	0.000473000	ROUTE-REFRESH Message
30	23.410975	10.77.1.1	10.76.1.1	BGP	100		25560	0.005482000	ROUTE-REFRESH Message, UPDATE Message
33	23.410165	10.76.1.1	10.75.1.1	BGP	73		179	0.000781000	KEEPALIVE Message

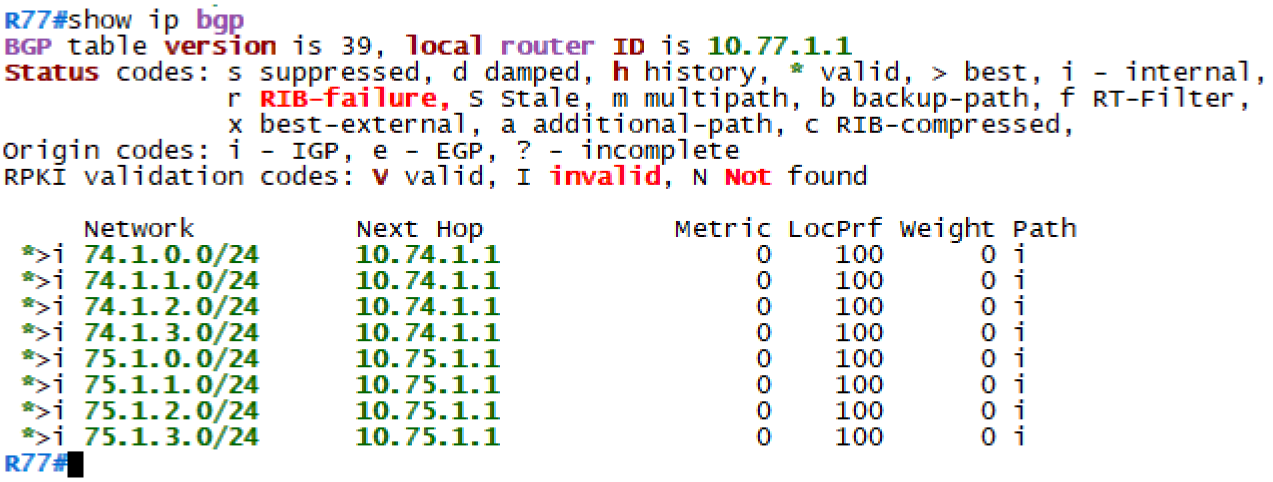
Now will advertise the networks in R74 Loopback address and R75 Loopback address and verify the same.



```
R74#show run |  se r b
router bgp 100
 bgp log-neighbor-changes
 network 74.1.0.0 mask 255.255.255.0
 network 74.1.1.0 mask 255.255.255.0
 network 74.1.2.0 mask 255.255.255.0
 network 74.1.3.0 mask 255.255.255.0
 redistribute connected
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.76.1.1 remote-as 100
 neighbor 10.76.1.1 update-source Loopback0
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
R74#
```

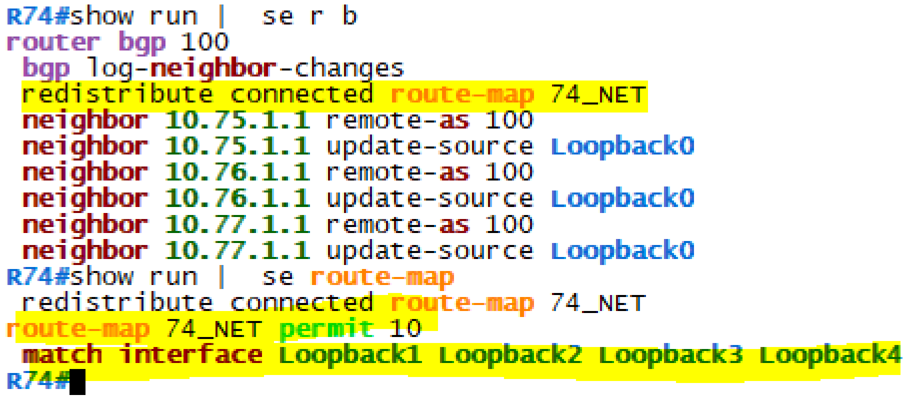
We will verify the same on R77

Now we can see the loopback 74.x 75.x network in the BGP route



```
R77#show ip bgp
BGP table version is 39, local router ID is 10.77.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>i 74.1.0.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.1.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.2.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.3.0/24      10.74.1.1                0    100      0 i
 *>i 75.1.0.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.1.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.2.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.3.0/24      10.75.1.1                0    100      0 i
R77#
```

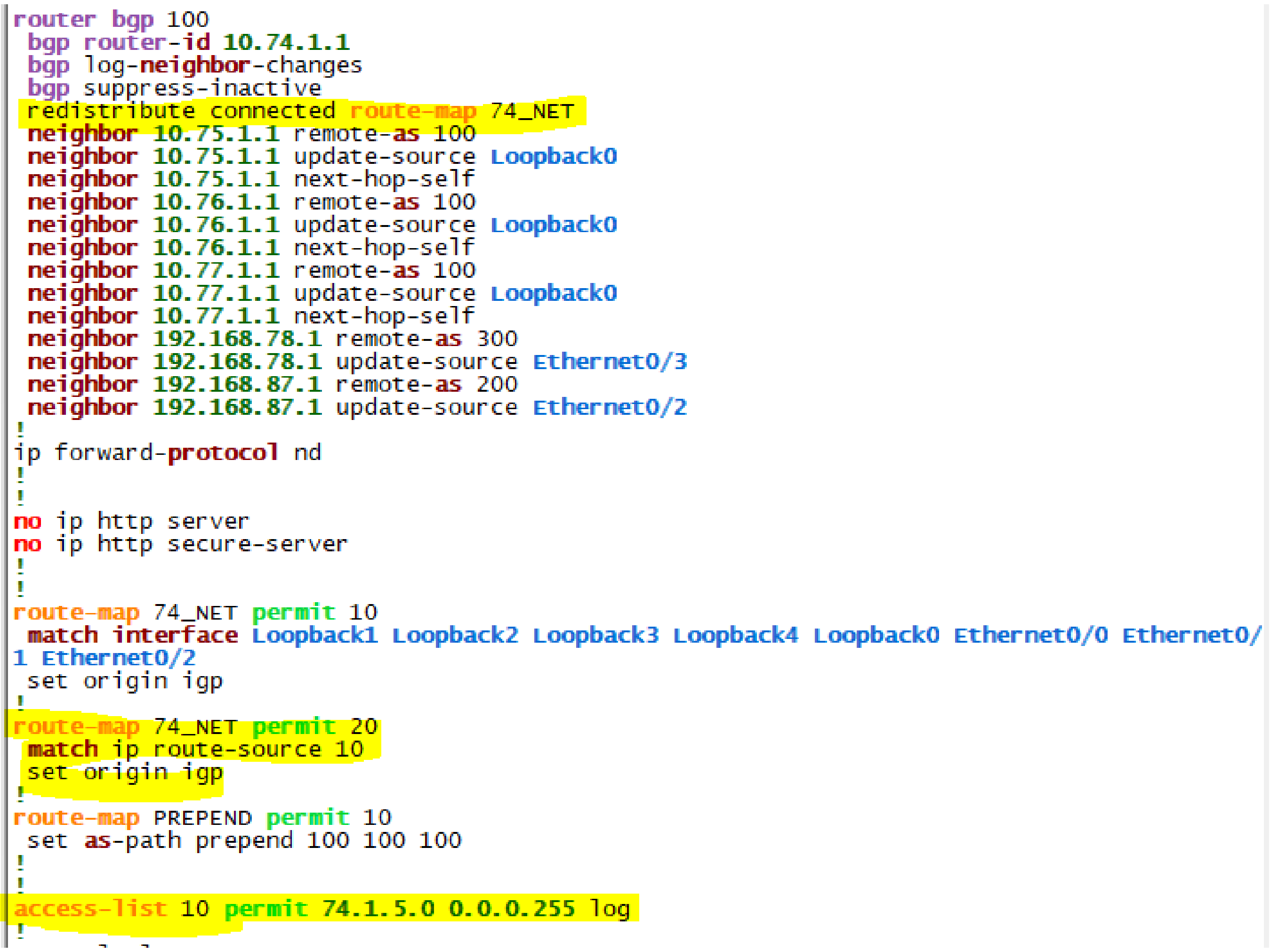
Now we will announce 74.1.0.0/24 74.1.1.0/24 74.1.2.0/24 74.1.3.0/24 using route-map instead of network statement.



```
R74#show run |  se r b
router bgp 100
 bgp log-neighbor-changes
 redistribute connected route-map 74_NET
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.76.1.1 remote-as 100
 neighbor 10.76.1.1 update-source Loopback0
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
R74#show run |  se route-map
 redistribute connected route-map 74_NET
route-map 74_NET permit 10
 match interface Loopback1 Loopback2 Loopback3 Loopback4
R74#
```

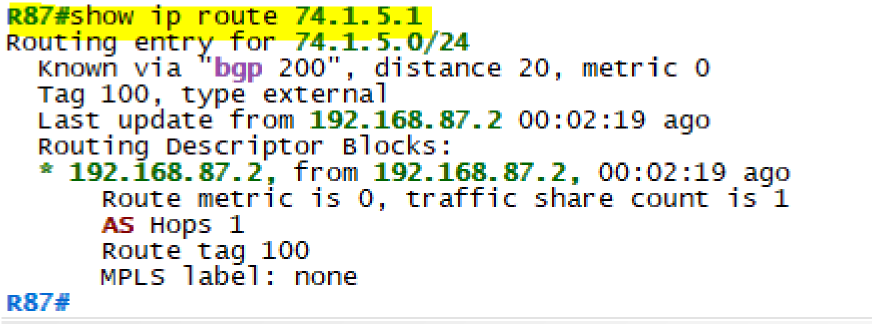
Above we have used route-map to match the interface (same can be done using ACL)
We did redistribute in to BGP

Below example on R74 we use ACL to redistribute and test

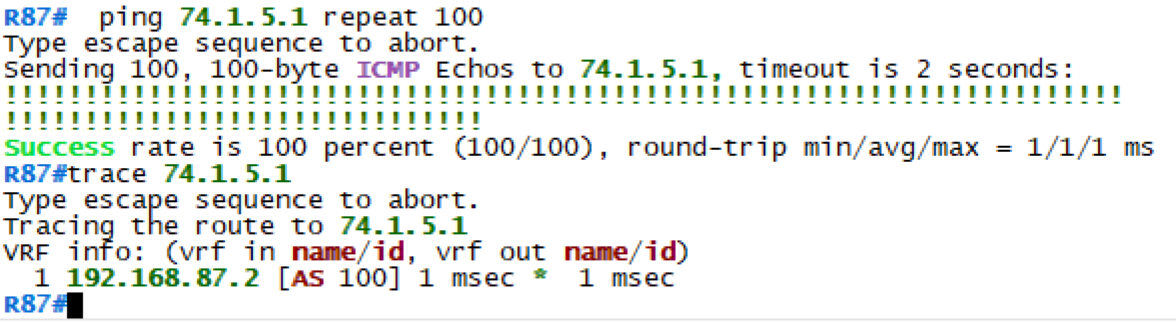


```
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 redistribute connected route-map 74_NET
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.75.1.1 next-hop-self
 neighbor 10.76.1.1 remote-as 100
 neighbor 10.76.1.1 update-source Loopback0
 neighbor 10.76.1.1 next-hop-self
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
 neighbor 10.77.1.1 next-hop-self
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
route-map 74_NET permit 10
 match interface Loopback1 Loopback2 Loopback3 Loopback4 Loopback0 Ethernet0/0 Ethernet0/
1 Ethernet0/2
 set origin igp
!
route-map 74_NET permit 20
 match ip route-source 10
 set origin igp
!
route-map PREPEND permit 10
 set as-path prepend 100 100 100
!
!
access-list 10 permit 74.1.5.0 0.0.0.255 log
!
```

We verify on R87 AS 100

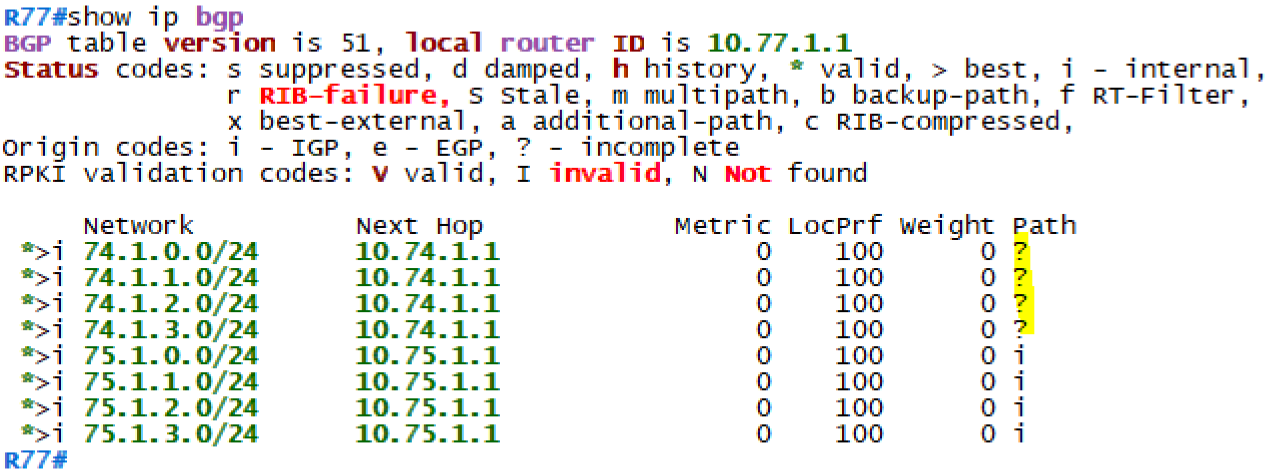


```
R87#show ip route 74.1.5.1
Routing entry for 74.1.5.0/24
  Known via "bgp 200", distance 20, metric 0
  Tag 100, type external
  Last update from 192.168.87.2 00:02:19 ago
  Routing Descriptor Blocks:
  * 192.168.87.2, from 192.168.87.2, 00:02:19 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
      Route tag 100
      MPLS label: none
R87#
```



```
R87#  ping 74.1.5.1 repeat 100
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 74.1.5.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/1/1 ms
R87#trace 74.1.5.1
Type escape sequence to abort.
Tracing the route to 74.1.5.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.87.2 [AS 100] 1 msec *  1 msec
R87#
```

Verify the routes on R77



```
R77#show ip bgp
BGP table version is 51, local router ID is 10.77.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>i 74.1.0.0/24      10.74.1.1                0    100      0 ?
 *>i 74.1.1.0/24      10.74.1.1                0    100      0 ?
 *>i 74.1.2.0/24      10.74.1.1                0    100      0 ?
 *>i 74.1.3.0/24      10.74.1.1                0    100      0 ?
 *>i 75.1.0.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.1.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.2.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.3.0/24      10.75.1.1                0    100      0 i
R77#
```

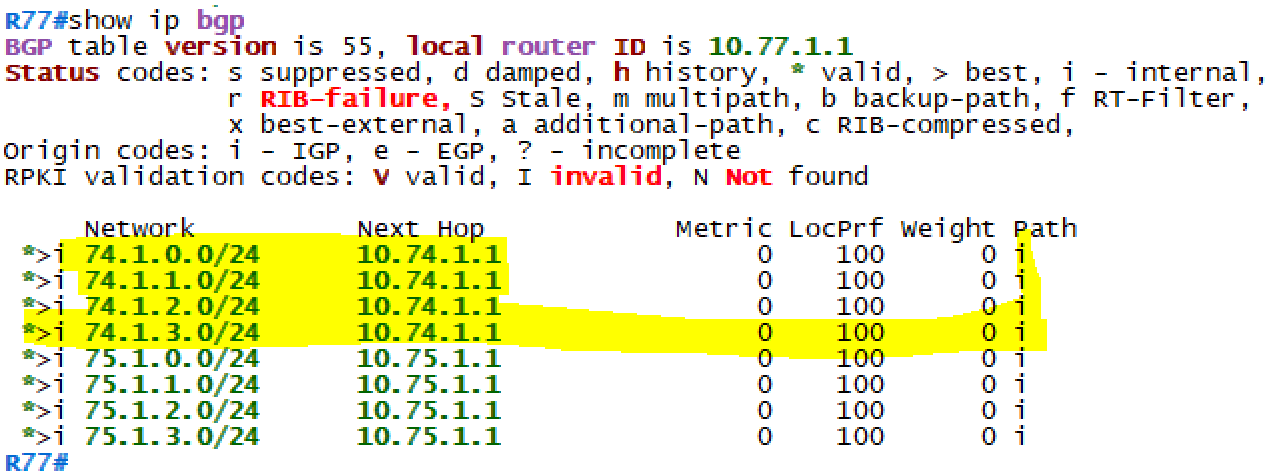
Note: now we can see the route as ? – this means ? always was redistributed

- I – means internal

Routes always preferred internal vs redistributed.

If redistributed need to be announce as internal, rather ? redistribute – change the statement as below and verify the same.

Verify on R77, now they should be coming with I – internal



```
R77#show ip bgp
BGP table version is 55, local router ID is 10.77.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>i 74.1.0.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.1.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.2.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.3.0/24      10.74.1.1                0    100      0 i
 *>i 75.1.0.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.1.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.2.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.3.0/24      10.75.1.1                0    100      0 i
R77#
```

EBGP Peering Rules:
EBGP packets default to TTL 1
-can be modified if neighbors are multiple hops away.

To accept and attempt BGP connections to external peers residing on networks that are not directly connected, use the **neighbor ebgp-multihop** command in router configuration mode.

If the router not directly connected you can use below commads for BGP for R75 and R76, they are not directly connected, with TTL1

neighbor 10.75.1.1 ebgp-multihop 1
neighbor 10.75.1.1 disable-connected-check

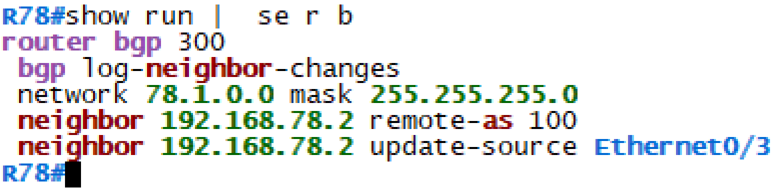
neighbor 10.76.1.1 ebgp-multihop 1
neighbor 10.76.1.1 disable-connected-check

Note: these commands are mutually exclusive - you would use one of the other, not both.

Now we have configured iBGP between AS100 Domain

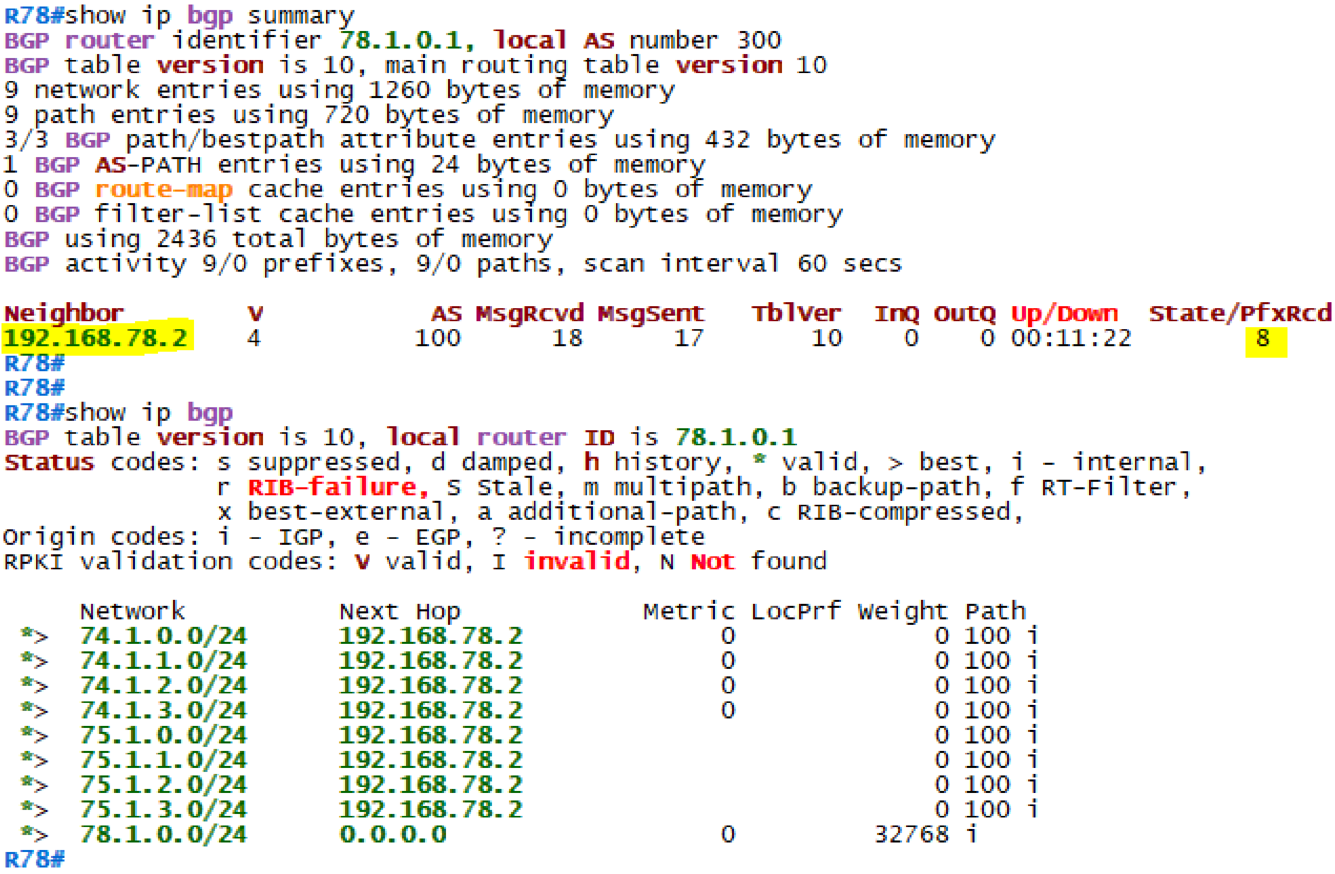
Now we going to Configure AS 300 and enable BGP relation with AS 100

R78 config as below



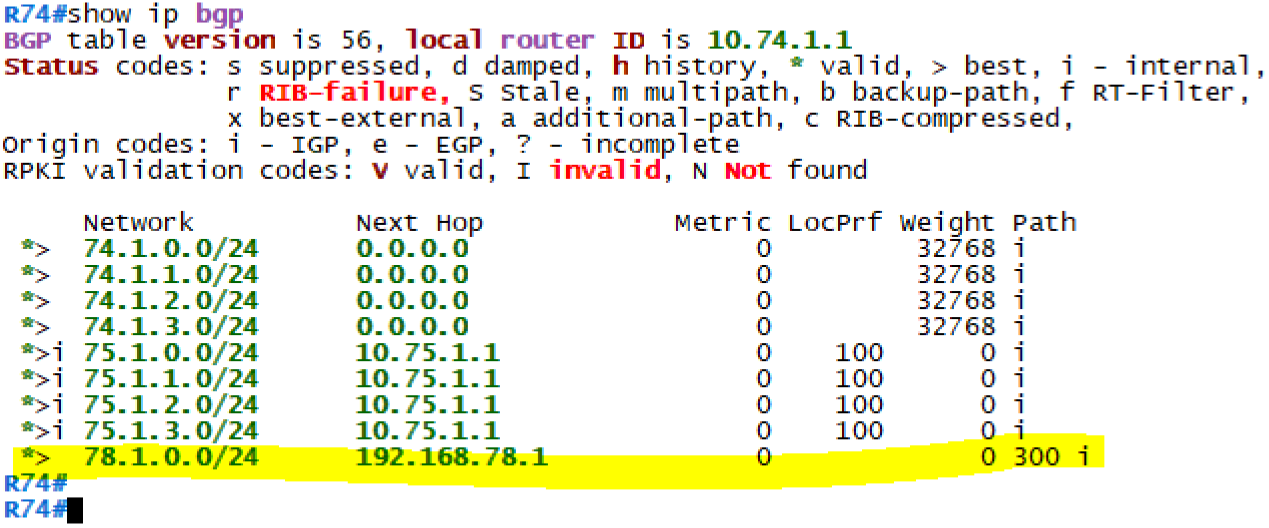
```
R78#show run |  se r b
router bgp 300
 bgp log-neighbor-changes
 network 78.1.0.0 mask 255.255.255.0
 neighbor 192.168.78.2 remote-as 100
 neighbor 192.168.78.2 update-source Ethernet0/3
R78#
```

Verify BGP peer with R74



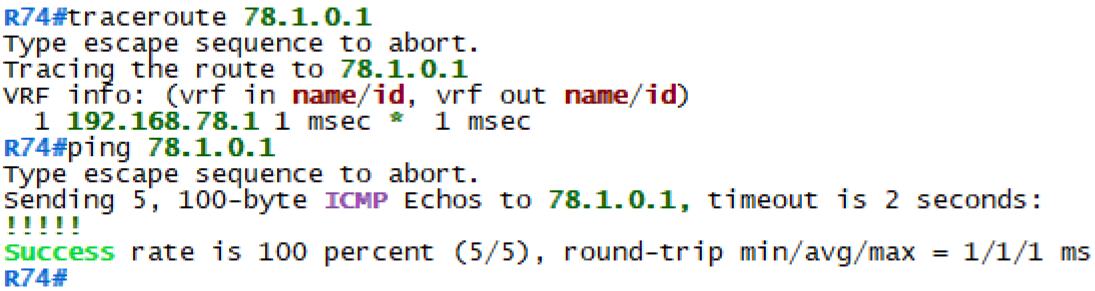
```
R78#show ip bgp summary
BGP router identifier 78.1.0.1, local AS number 300
BGP table version is 10, main routing table version 10
9 network entries using 1260 bytes of memory
9 path entries using 720 bytes of memory
3/3 BGP path/bestpath attribute entries using 432 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 2436 total bytes of memory
BGP activity 9/0 prefixes, 9/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.78.2    4          100      18      17       10    0    0 00:11:22        8
R78#
R78#
R78#show ip bgp
BGP table version is 10, local router ID is 78.1.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  74.1.0.0/24      192.168.78.2             0             0 100 i
 *>  74.1.1.0/24      192.168.78.2             0             0 100 i
 *>  74.1.2.0/24      192.168.78.2             0             0 100 i
 *>  74.1.3.0/24      192.168.78.2             0             0 100 i
 *>  75.1.0.0/24      192.168.78.2                           0 100 i
 *>  75.1.1.0/24      192.168.78.2                           0 100 i
 *>  75.1.2.0/24      192.168.78.2                           0 100 i
 *>  75.1.3.0/24      192.168.78.2                           0 100 i
 *>  78.1.0.0/24      0.0.0.0                  0         32768 i
R78#
```

Since we have announced R78 loopback address in to BGP verify the routes on R74 and IBGP network.



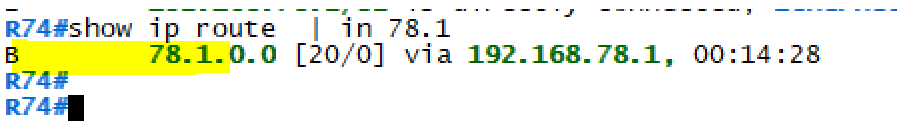
```
R74#show ip bgp
BGP table version is 56, local router ID is 10.74.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  74.1.0.0/24      0.0.0.0                  0         32768 i
 *>  74.1.1.0/24      0.0.0.0                  0         32768 i
 *>  74.1.2.0/24      0.0.0.0                  0         32768 i
 *>  74.1.3.0/24      0.0.0.0                  0         32768 i
 *>i 75.1.0.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.1.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.2.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.3.0/24      10.75.1.1                0    100      0 i
 *>  78.1.0.0/24      192.168.78.1             0             0 300 i
R74#
R74#
```

Check the reachability.



```
R74#traceroute 78.1.0.1
Type escape sequence to abort.
Tracing the route to 78.1.0.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.78.1 1 msec *  1 msec
R74#ping 78.1.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 78.1.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R74#
```

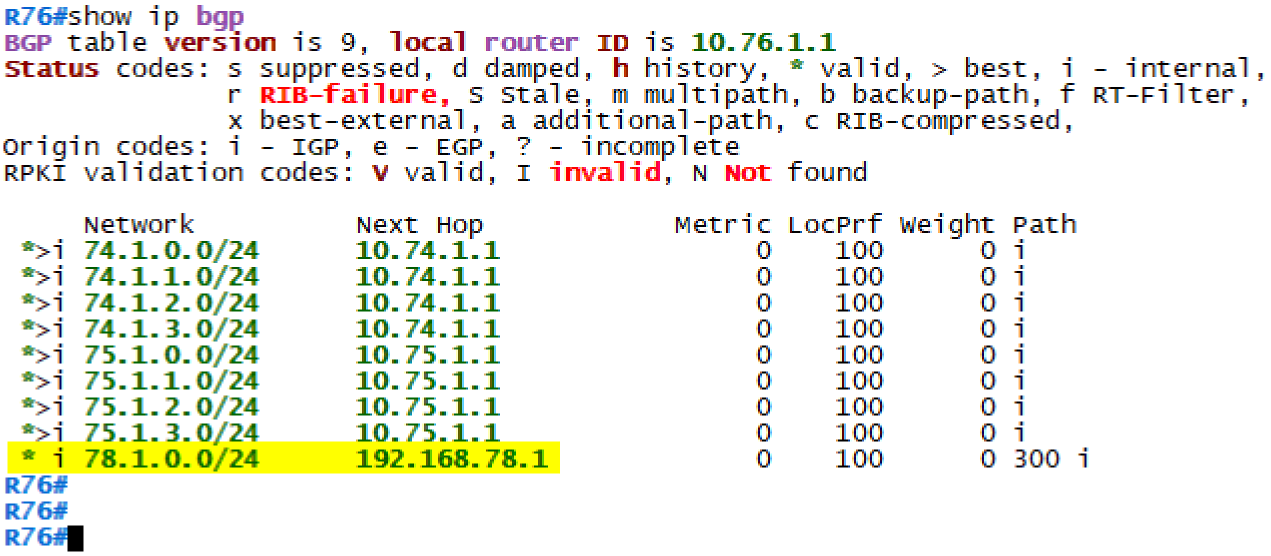
Check the routing table in R74



```
R74#show ip route  | in 78.1
B        78.1.0.0 [20/0] via 192.168.78.1, 00:14:28
R74#
R74#
```

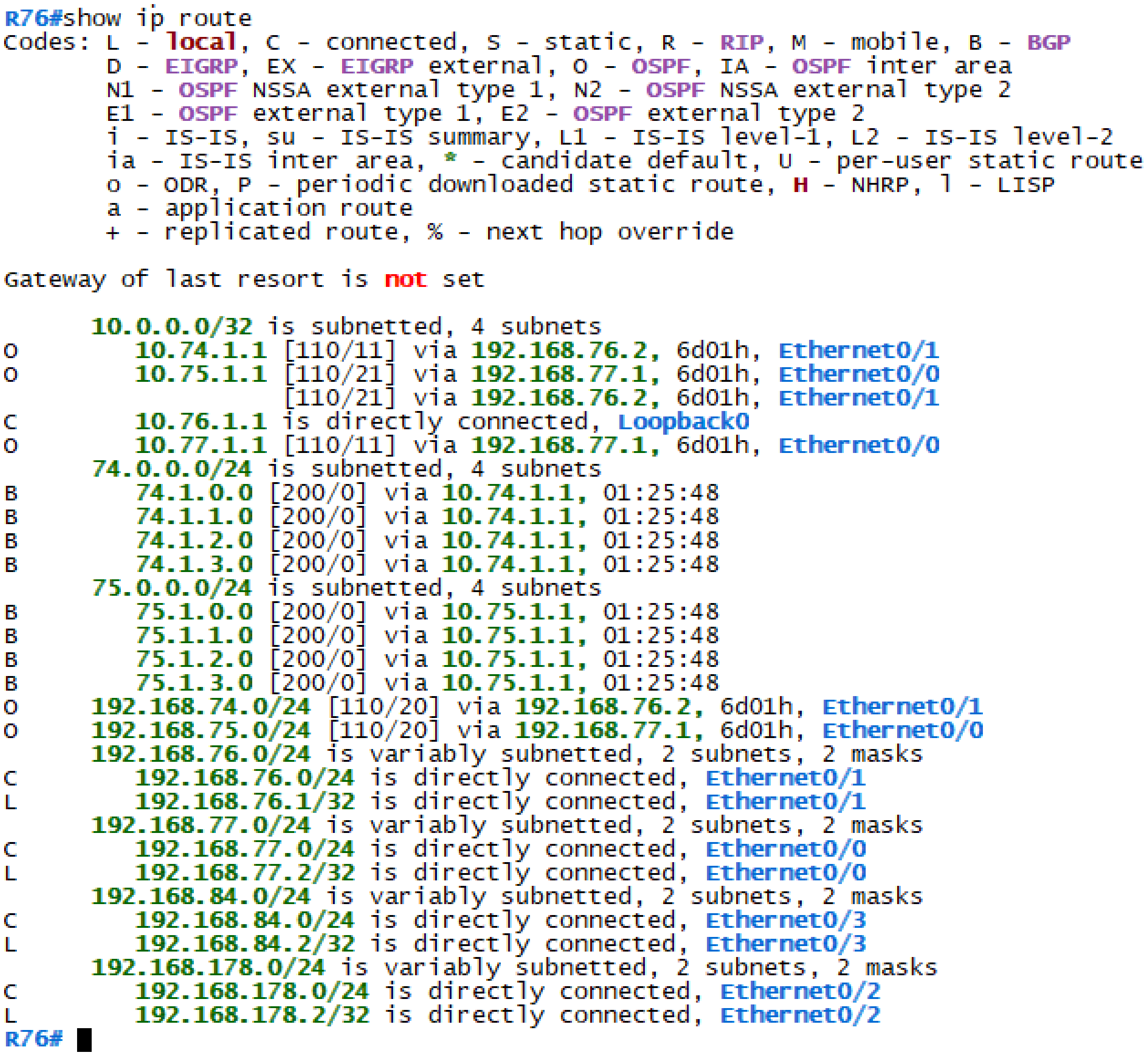
Now the same routes are propagated across iBGP network AS100.

Now we have issue here – Lets check in R76 for the route of 78.1.0.0/24



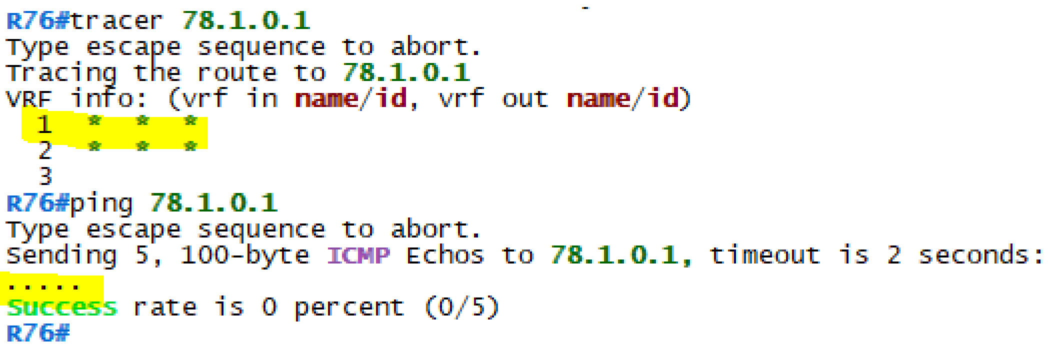
```
R76#show ip bgp
BGP table version is 9, local router ID is 10.76.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>i 74.1.0.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.1.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.2.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.3.0/24      10.74.1.1                0    100      0 i
 *>i 75.1.0.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.1.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.2.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.3.0/24      10.75.1.1                0    100      0 i
 * i 78.1.0.0/24      192.168.78.1             0    100      0 300 i
R76#
R76#
R76#
```

We can see BGP route, but when we check in routing table and reachability it fails.



```
R76#show ip route
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/32 is subnetted, 4 subnets
O        10.74.1.1 [110/11] via 192.168.76.2, 6d01h, Ethernet0/1
O        10.75.1.1 [110/21] via 192.168.77.1, 6d01h, Ethernet0/0
                   [110/21] via 192.168.76.2, 6d01h, Ethernet0/1
C        10.76.1.1 is directly connected, Loopback0
O        10.77.1.1 [110/11] via 192.168.77.1, 6d01h, Ethernet0/0
      74.0.0.0/24 is subnetted, 4 subnets
B        74.1.0.0 [200/0] via 10.74.1.1, 01:25:48
B        74.1.1.0 [200/0] via 10.74.1.1, 01:25:48
B        74.1.2.0 [200/0] via 10.74.1.1, 01:25:48
B        74.1.3.0 [200/0] via 10.74.1.1, 01:25:48
      75.0.0.0/24 is subnetted, 4 subnets
B        75.1.0.0 [200/0] via 10.75.1.1, 01:25:48
B        75.1.1.0 [200/0] via 10.75.1.1, 01:25:48
B        75.1.2.0 [200/0] via 10.75.1.1, 01:25:48
B        75.1.3.0 [200/0] via 10.75.1.1, 01:25:48
O     192.168.74.0/24 [110/20] via 192.168.76.2, 6d01h, Ethernet0/1
O     192.168.75.0/24 [110/20] via 192.168.77.1, 6d01h, Ethernet0/0
      192.168.76.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.76.0/24 is directly connected, Ethernet0/1
L        192.168.76.1/32 is directly connected, Ethernet0/1
      192.168.77.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.77.0/24 is directly connected, Ethernet0/0
L        192.168.77.2/32 is directly connected, Ethernet0/0
      192.168.84.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.84.0/24 is directly connected, Ethernet0/3
L        192.168.84.2/32 is directly connected, Ethernet0/3
      192.168.178.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.178.0/24 is directly connected, Ethernet0/2
L        192.168.178.2/32 is directly connected, Ethernet0/2
R76#
```

Now 78.1.0.0/24 network in routing table and reachability fails.

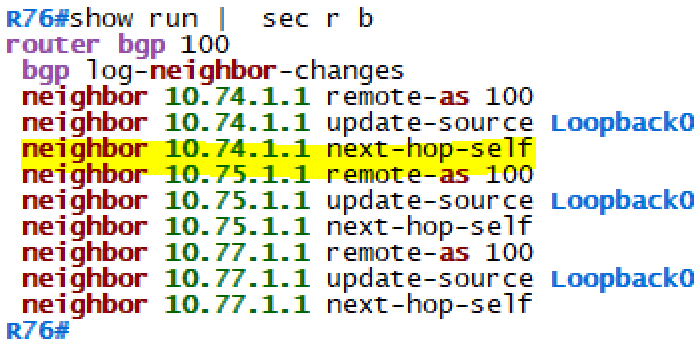


```
R76#tracer 78.1.0.1
Type escape sequence to abort.
Tracing the route to 78.1.0.1
VRF info: (vrf in name/id, vrf out name/id)
  1  *  *  *
  2  *  *  *
  3
R76#ping 78.1.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 78.1.0.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R76#
```

This is because the route learning from Point to Point interface IP address, in which R76 have no visibility of that IP.

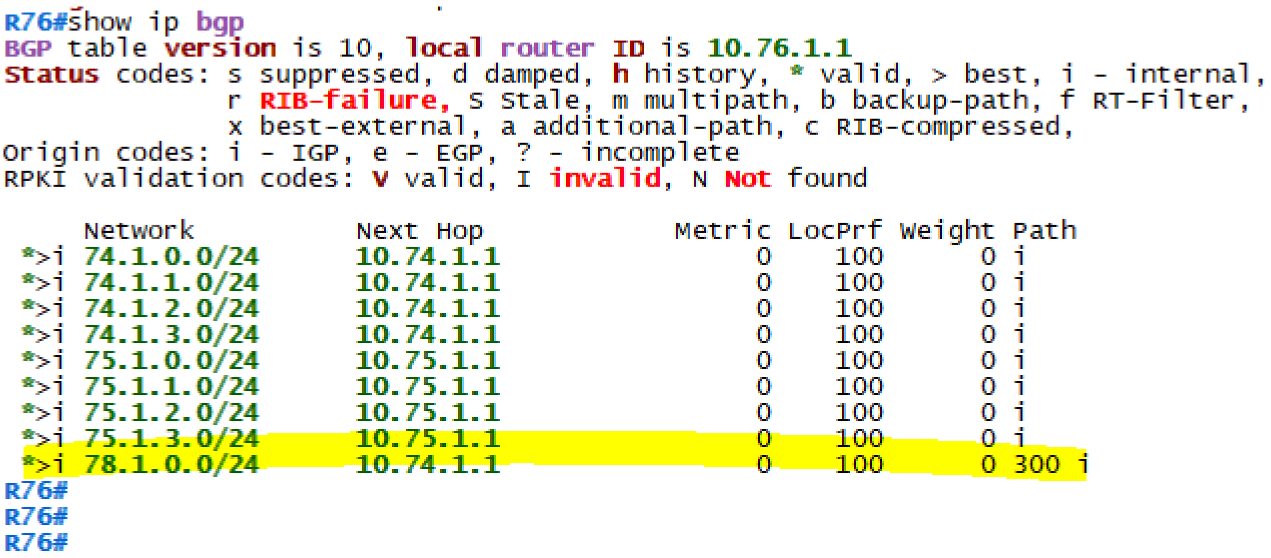
The design issue - the next hop value that BGP reports must then be recused through some other IGP routing protocol. BGP will rely on EIGRP, OSPF etc. BGP is for destinations outside our network that we're trying to reach.

So fix is always in iBGP required command **next-hop-self**

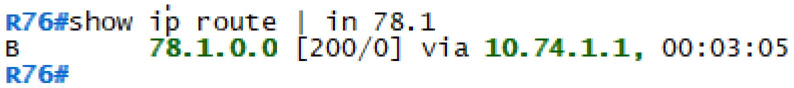


```
R76#show run |  sec r b
router bgp 100
 bgp log-neighbor-changes
 neighbor 10.74.1.1 remote-as 100
 neighbor 10.74.1.1 update-source Loopback0
 neighbor 10.74.1.1 next-hop-self
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.75.1.1 next-hop-self
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
 neighbor 10.77.1.1 next-hop-self
R76#
```

Now we verify the routing table, and reachability.



```
R76#show ip bgp
BGP table version is 10, local router ID is 10.76.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>i 74.1.0.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.1.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.2.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.3.0/24      10.74.1.1                0    100      0 i
 *>i 75.1.0.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.1.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.2.0/24      10.75.1.1                0    100      0 i
 *>i 75.1.3.0/24      10.75.1.1                0    100      0 i
 *>i 78.1.0.0/24      10.74.1.1                0    100      0 300 i
R76#
R76#
R76#
```



```
R76#show ip route | in 78.1
B        78.1.0.0 [200/0] via 10.74.1.1, 00:03:05
R76#
```

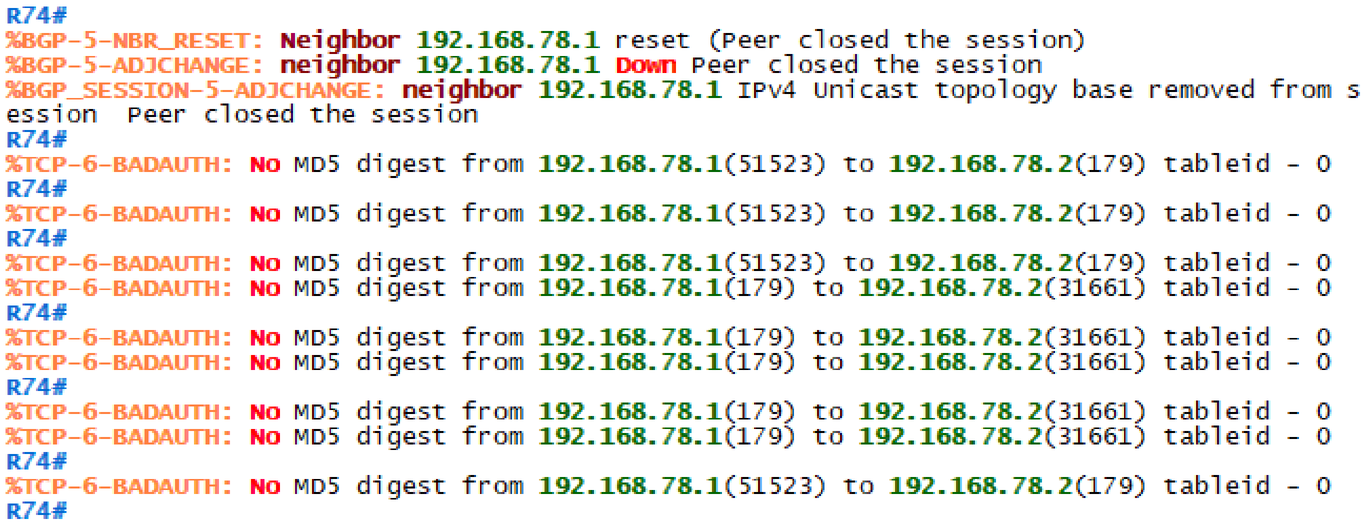
BGP Authentication

BGP can configure MD5 authentication between two BGP peers, meaning that each segment sent on the TCP connection between the peers is verified. MD5 authentication must be configured with the same password on both BGP peers; otherwise, the connection between them will not be made.

The local router will attempt to maintain the peering session using the new password until the BGP hold-down timer expires. The default time period is **180 seconds**. If the password is not entered or changed on the remote router before the hold-down timer expires, the session will time out.

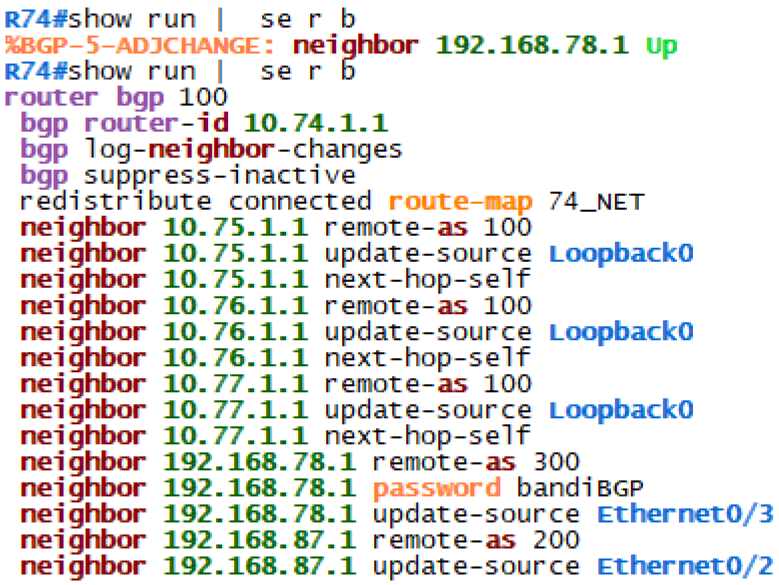
Now we will configure Authentication between R74 and R78, that is AS100 and AS 300

WE configure R74 and observe on R74 errors (since R78 has not configured with Password)

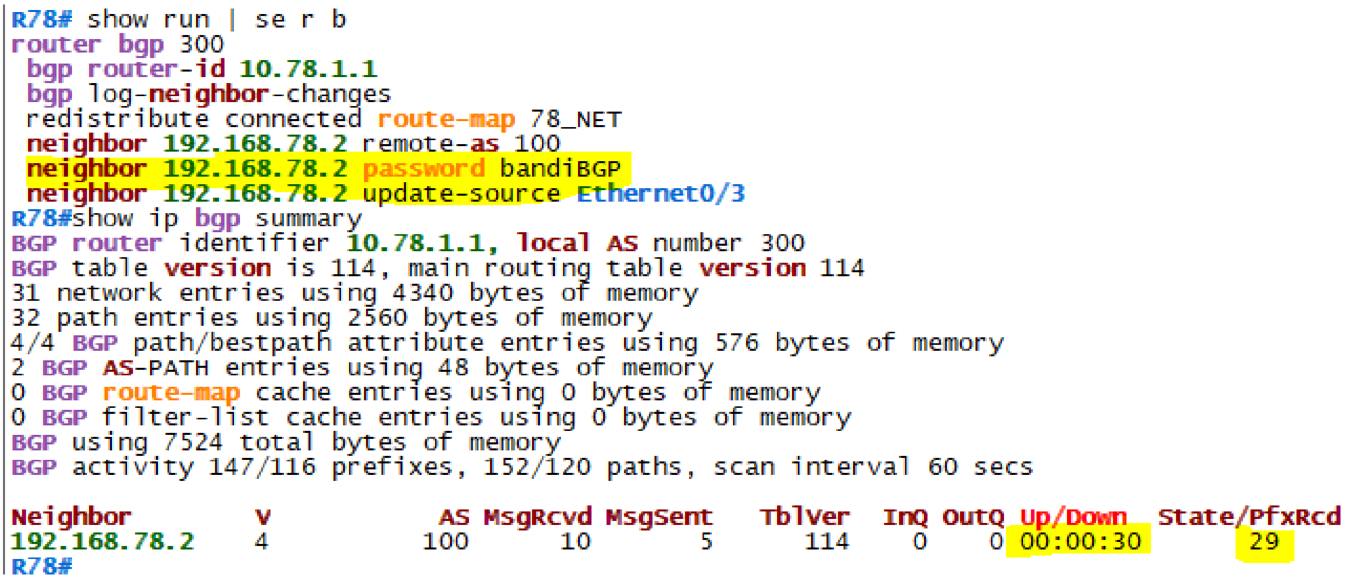


```
R74#
%BGP-5-NBR_RESET: Neighbor 192.168.78.1 reset (Peer closed the session)
%BGP-5-ADJCHANGE: neighbor 192.168.78.1 Down Peer closed the session
%BGP_SESSION-5-ADJCHANGE: neighbor 192.168.78.1 IPv4 Unicast topology base removed from s
ession  Peer closed the session
R74#
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(51523) to 192.168.78.2(179) tableid - 0
R74#
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(51523) to 192.168.78.2(179) tableid - 0
R74#
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(51523) to 192.168.78.2(179) tableid - 0
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(179) to 192.168.78.2(31661) tableid - 0
R74#
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(179) to 192.168.78.2(31661) tableid - 0
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(179) to 192.168.78.2(31661) tableid - 0
R74#
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(179) to 192.168.78.2(31661) tableid - 0
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(179) to 192.168.78.2(31661) tableid - 0
R74#
%TCP-6-BADAUTH: No MD5 digest from 192.168.78.1(51523) to 192.168.78.2(179) tableid - 0
R74#
```

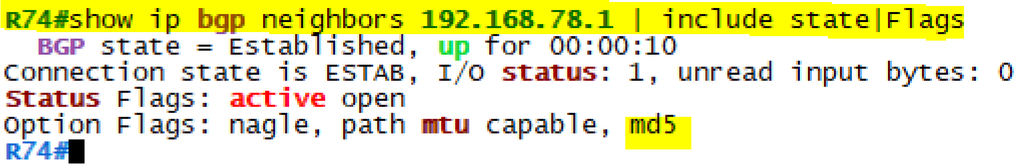
R74



```
R74#show run |  se r b
%BGP-5-ADJCHANGE: neighbor 192.168.78.1 Up
R74#show run |  se r b
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 redistribute connected route-map 74_NET
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.75.1.1 next-hop-self
 neighbor 10.76.1.1 remote-as 100
 neighbor 10.76.1.1 update-source Loopback0
 neighbor 10.76.1.1 next-hop-self
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
 neighbor 10.77.1.1 next-hop-self
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
```



```
R78# show run | se r b
router bgp 300
 bgp router-id 10.78.1.1
 bgp log-neighbor-changes
 redistribute connected route-map 78_NET
 neighbor 192.168.78.2 remote-as 100
 neighbor 192.168.78.2 password bandiBGP
 neighbor 192.168.78.2 update-source Ethernet0/3
R78#show ip bgp summary
BGP router identifier 10.78.1.1, local AS number 300
BGP table version is 114, main routing table version 114
31 network entries using 4340 bytes of memory
32 path entries using 2560 bytes of memory
4/4 BGP path/bestpath attribute entries using 576 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 7524 total bytes of memory
BGP activity 147/116 prefixes, 152/120 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.78.2    4          100      10       5      114    0    0 00:00:30       29
R78#
```



```
R74#show ip bgp neighbors 192.168.78.1 | include state|Flags
  BGP state = Established, up for 00:00:10
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Status Flags: active open
Option Flags: nagle, path mtu capable, md5
R74#
```

BGP TTL Security

eBGP peers need to be (by default) directly connected. That is, the BGP packets generated by a BGP speaker have a TTL of 1. When a BGP peer receives the packet, it decrements the TTL on ingress and process the packet normally. If the BGP peer is more than one layer 3 hop away, the first router to receive the packet from the BGP speaker will decrement the TTL of the packet. So in the case of a BGP peer multiple hops away, the eBGP packet sent by the BGP speaker will have a TTL of one. The next router in the path to the peer will decrement the TTL to 0, realize it can't forward the packet, drop it, and generate an ICMP TTL expired in transit back to Router 1.

So we need to use **neighbour X.x.x.x ttl-security hop 2** (example if the router 2 hops away)

BGP Peer Groups

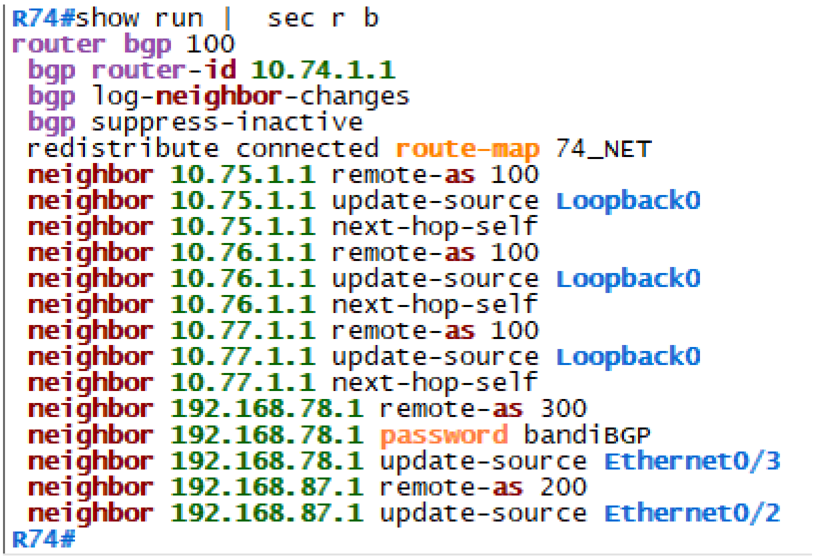
You can group BGP neighbors who share the same outbound policies together in what is called a BGP peer group. Instead of configuring each neighbor with the same policy individually, a peer group allows you to group the policies which can be applied to individual peers thus making efficient update calculation along with simplified configuration.

Requirements of Peer Groups

- Peer groups have these requirements:
- All members of a peer group must share identical outbound announcement policies (such as distribute-list, filter-list, and route-map), except for default-originate, which is handled on a per-peer basis even for peer group members.
- You can customize the inbound update policy for any member of a peer group.
- A peer group must be either internal (with internal BGP (iBGP) members) or external (with external BGP (eBGP) members). Members of an external peer group have different autonomous system (AS) numbers.

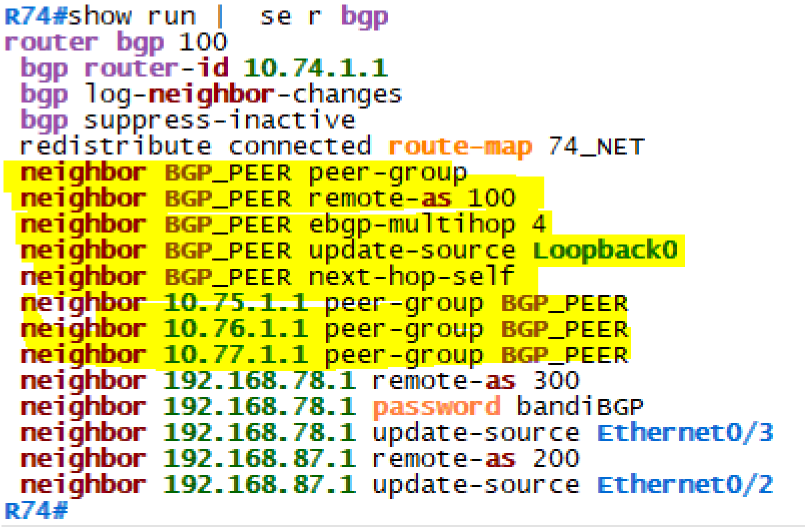
On our configuration we have R74 peering with Many Other Routers like R75, R76, R77
Now we configure peer groups on R74 and analyse the outcome.

Exiting Configuration:



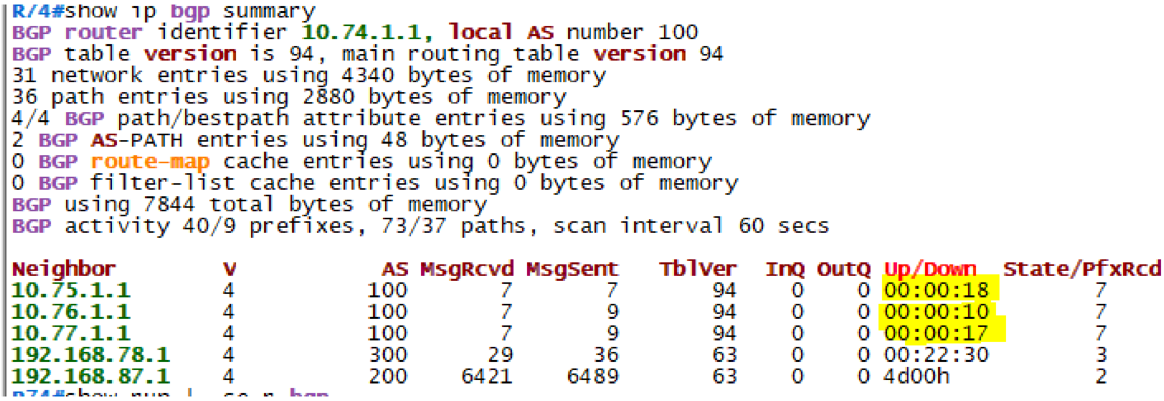
```
R74#show run |  sec r b
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 redistribute connected route-map 74_NET
 neighbor 10.75.1.1 remote-as 100
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.75.1.1 next-hop-self
 neighbor 10.76.1.1 remote-as 100
 neighbor 10.76.1.1 update-source Loopback0
 neighbor 10.76.1.1 next-hop-self
 neighbor 10.77.1.1 remote-as 100
 neighbor 10.77.1.1 update-source Loopback0
 neighbor 10.77.1.1 next-hop-self
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
R74#
```

We will make now peer group config with R75, R76, R77

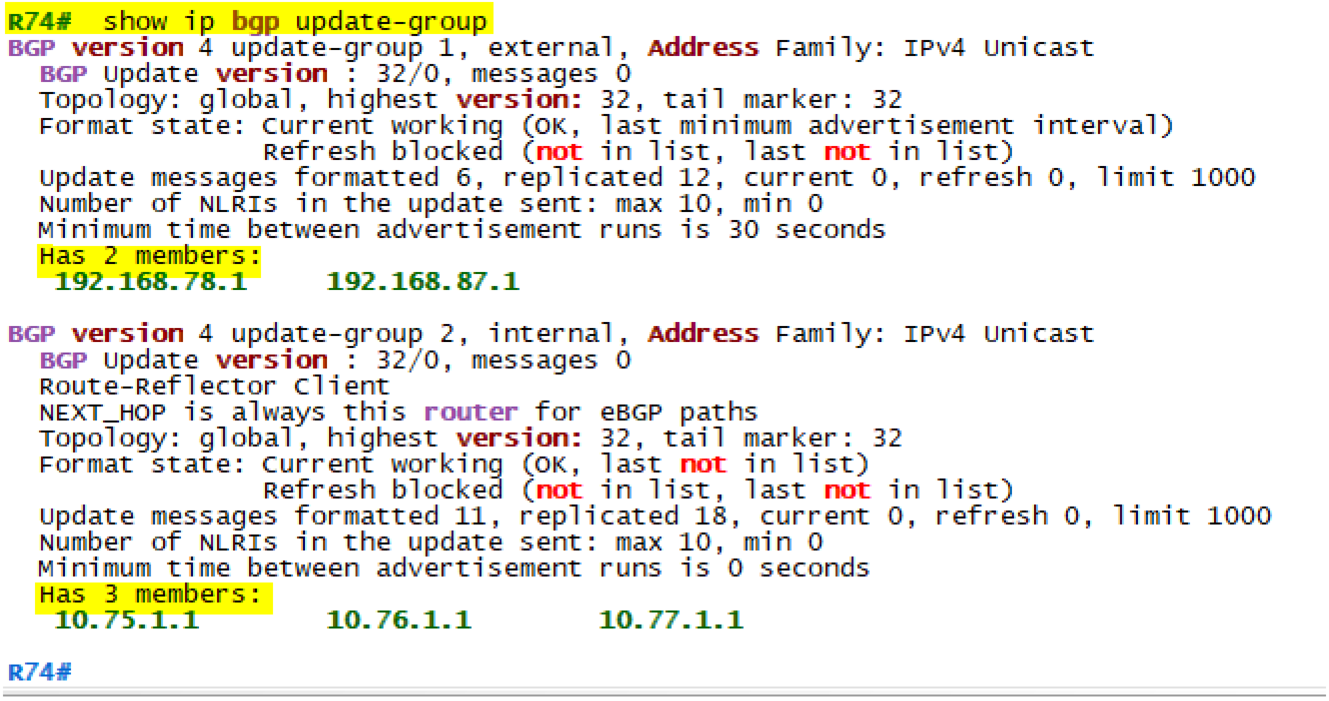


```
R74#show run |  se r bgp
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER next-hop-self
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
R74#
```

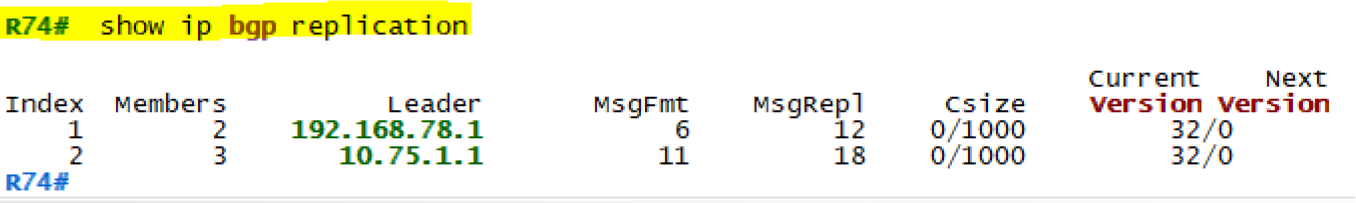
Check bgp summary



```
R74#show ip bgp summary
BGP router identifier 10.74.1.1, local AS number 100
BGP table version is 94, main routing table version 94
31 network entries using 4340 bytes of memory
36 path entries using 2880 bytes of memory
4/4 BGP path/bestpath attribute entries using 576 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 7844 total bytes of memory
BGP activity 40/9 prefixes, 73/37 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.75.1.1       4          100       7       7       94    0    0 00:00:18        7
10.76.1.1       4          100       7       9       94    0    0 00:00:10        7
10.77.1.1       4          100       7       9       94    0    0 00:00:17        7
192.168.78.1    4          300      29      36       63    0    0 00:22:30        3
192.168.87.1    4          200    6421    6489       63    0    0 4d00h           2
```



```
R74#  show ip bgp update-group
BGP version 4 update-group 1, external, Address Family: IPv4 Unicast
  BGP Update version : 32/0, messages 0
  Topology: global, highest version: 32, tail marker: 32
  Format state: Current working (OK, last minimum advertisement interval)
                Refresh blocked (not in list, last not in list)
  Update messages formatted 6, replicated 12, current 0, refresh 0, limit 1000
  Number of NLRIs in the update sent: max 10, min 0
  Minimum time between advertisement runs is 30 seconds
  Has 2 members:
   192.168.78.1     192.168.87.1

BGP version 4 update-group 2, internal, Address Family: IPv4 Unicast
  BGP Update version : 32/0, messages 0
  Route-Reflector Client
  NEXT_HOP is always this router for eBGP paths
  Topology: global, highest version: 32, tail marker: 32
  Format state: Current working (OK, last not in list)
                Refresh blocked (not in list, last not in list)
  Update messages formatted 11, replicated 18, current 0, refresh 0, limit 1000
  Number of NLRIs in the update sent: max 10, min 0
  Minimum time between advertisement runs is 0 seconds
  Has 3 members:
   10.75.1.1        10.76.1.1        10.77.1.1

R74#
```



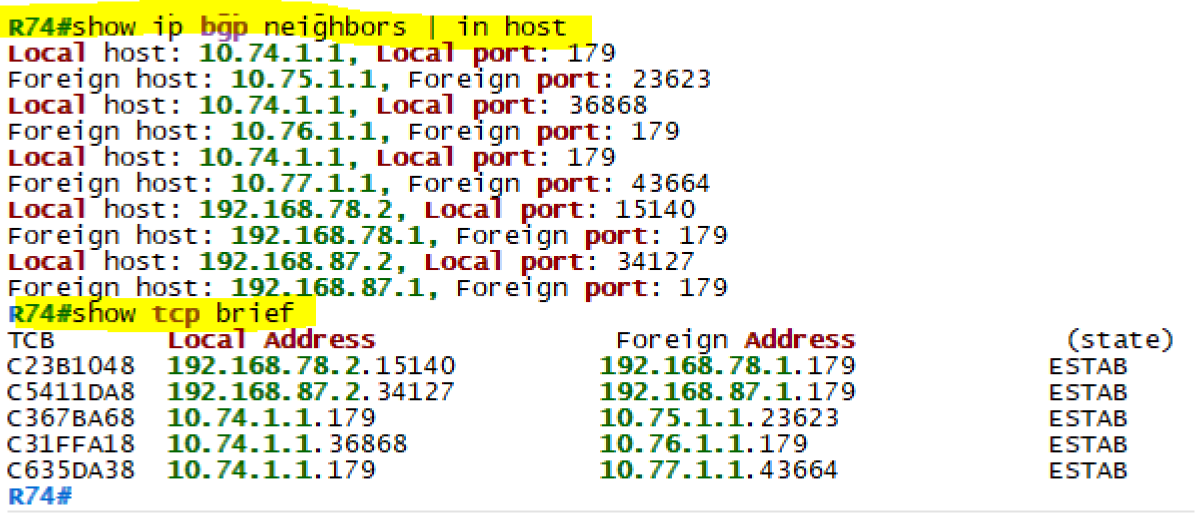
```
R74#  show ip bgp replication

                                                                  Current    Next
Index  Members          Leader        MsgFmt     MsgRepl     Csize   Version Version
    1        2    192.168.78.1             6          12    0/1000      32/0
    2        3       10.75.1.1            11          18    0/1000      32/0
R74#
```

BGP Active / Passive

Neighbours with the lowest BGP router identifier will establish the connection to the remote peer via TCP port 179, the source port will be random. We can modify this behaviour with a few simple commands.

neighbour x.x.x.x transport connection-mode passive (or active)

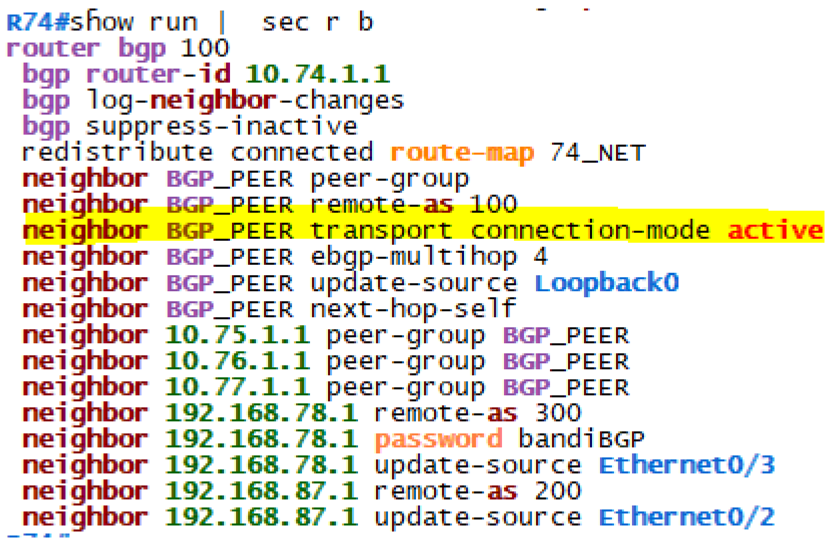
let's verify here who is active and passive you can use below 2 command to verify



```
R74#show ip bgp neighbors | in host
Local host: 10.74.1.1, Local port: 179
Foreign host: 10.75.1.1, Foreign port: 23623
Local host: 10.74.1.1, Local port: 36868
Foreign host: 10.76.1.1, Foreign port: 179
Local host: 10.74.1.1, Local port: 179
Foreign host: 10.77.1.1, Foreign port: 43664
Local host: 192.168.78.2, Local port: 15140
Foreign host: 192.168.78.1, Foreign port: 179
Local host: 192.168.87.2, Local port: 34127
Foreign host: 192.168.87.1, Foreign port: 179
R74#show tcp brief
TCB       Local Address               Foreign Address             (state)
C23B1048  192.168.78.2.15140         192.168.78.1.179            ESTAB
C5411DA8  192.168.87.2.34127         192.168.87.1.179            ESTAB
C367BA68  10.74.1.1.179              10.75.1.1.23623             ESTAB
C31FFA18  10.74.1.1.36868            10.76.1.1.179               ESTAB
C635DA38  10.74.1.1.179              10.77.1.1.43664             ESTAB
R74#
```

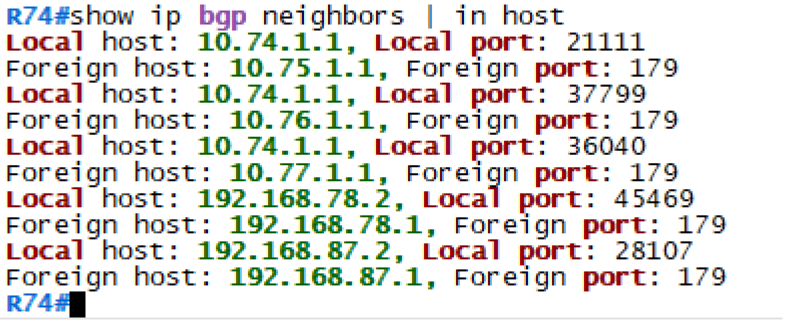
R74 listening from R75 and R77 on the above example (that means it is passive)
R74 initiated connected to R76 that means it is BGP active.

Now we make R74 as active (I have changed peer group so all the config same)

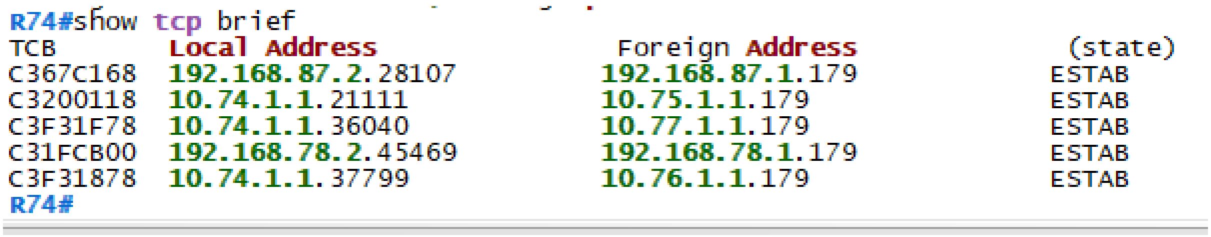


```
R74#show run |  sec r b
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER transport connection-mode active
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER next-hop-self
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
```

We verify here: R74 is intiating the connection, other routers listening on port 179



```
R74#show ip bgp neighbors | in host
Local host: 10.74.1.1, Local port: 21111
Foreign host: 10.75.1.1, Foreign port: 179
Local host: 10.74.1.1, Local port: 37799
Foreign host: 10.76.1.1, Foreign port: 179
Local host: 10.74.1.1, Local port: 36040
Foreign host: 10.77.1.1, Foreign port: 179
Local host: 192.168.78.2, Local port: 45469
Foreign host: 192.168.78.1, Foreign port: 179
Local host: 192.168.87.2, Local port: 28107
Foreign host: 192.168.87.1, Foreign port: 179
R74#
```



```
R74#show tcp brief
TCB        Local Address              Foreign Address           (state)
C367C168   192.168.87.2.28107         192.168.87.1.179          ESTAB
C3200118   10.74.1.1.21111            10.75.1.1.179             ESTAB
C3F31F78   10.74.1.1.36040            10.77.1.1.179             ESTAB
C31FCB00   192.168.78.2.45469         192.168.78.1.179          ESTAB
C3F31878   10.74.1.1.37799            10.76.1.1.179             ESTAB
R74#
```

BGP path MTU discovery :

When a host generates Data, the packetization layer (TCP/UDP) will decide the packet size based on the MTU size of the outgoing interface. When the packet traverses along the path to ultimate destination, it may get fragmented if the MTU of outgoing interface on any router is less than the packet size. Packet fragmentation on intermittent router is always considered inefficient as it may result in below:

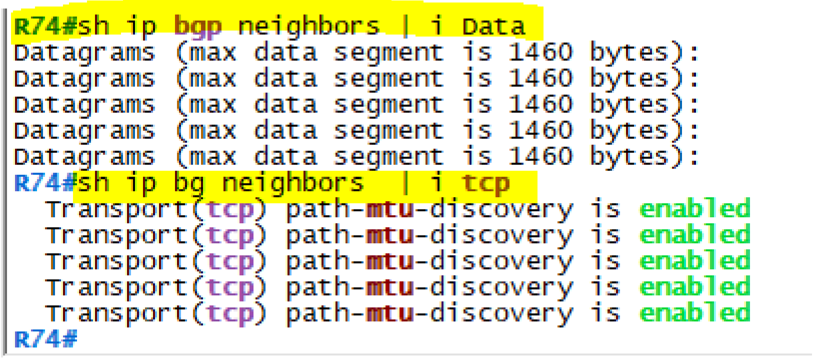
1. One fragment lost will result in entire packet sent from the source.

2. Introduce CPU/buffer burden.

Path MTU Discovery is introduced to reduce the chances of IP packet getting fragmented along the path. The ultimate source will use this feature to identify the lowest MTU along the path to destination and will decide the packet size.

BGP support for Path MTU Discovery

Introducing Path MTU Discovery on BGP session allows the BGP router to discover the best MTU size along the path to neighbor resulting in efficient way of exchanging BGP packets.



```
R74#sh ip bgp neighbors | i Data
Datagrams (max data segment is 1460 bytes):
Datagrams (max data segment is 1460 bytes):
Datagrams (max data segment is 1460 bytes):
Datagrams (max data segment is 1460 bytes):
Datagrams (max data segment is 1460 bytes):
R74#sh ip bg neighbors  | i tcp
  Transport(tcp) path-mtu-discovery is enabled
  Transport(tcp) path-mtu-discovery is enabled
  Transport(tcp) path-mtu-discovery is enabled
  Transport(tcp) path-mtu-discovery is enabled
  Transport(tcp) path-mtu-discovery is enabled
R74#
```

2.7.e Implement and troubleshoot scalability
2.7.e [i] Route-reflector, cluster

Within BGP there are two common ways to build a BGP network.

- Full Mesh (we have configured as of now above example)
- Route Reflector.

BGP Route Reflector

A route reflector can have three types of peers:
EBGP peers

client peers,
nonclient peers.

EBGP peers are neighbors in a different AS number, including peers in
different Sub-ASs in confederation.
Client peers are iBGP neighbors that have the route-reflector-client statement configured.
Non-client peers are normal iBGP peers that do not have the route-reflector-client statement configured.
Routing advertisements sent from the route reflector must conform to the following three
rules:

1. Routes learned from EBGP peers can be sent to other EBGP peers, clients, and nonclients.
2. Routes learned from client peers can be sent to EBGP peers, other client peers, and
non-clients.
3. Routes learned from non-client peers can be sent to EBGP peers, and client peers, but not other non-clients.

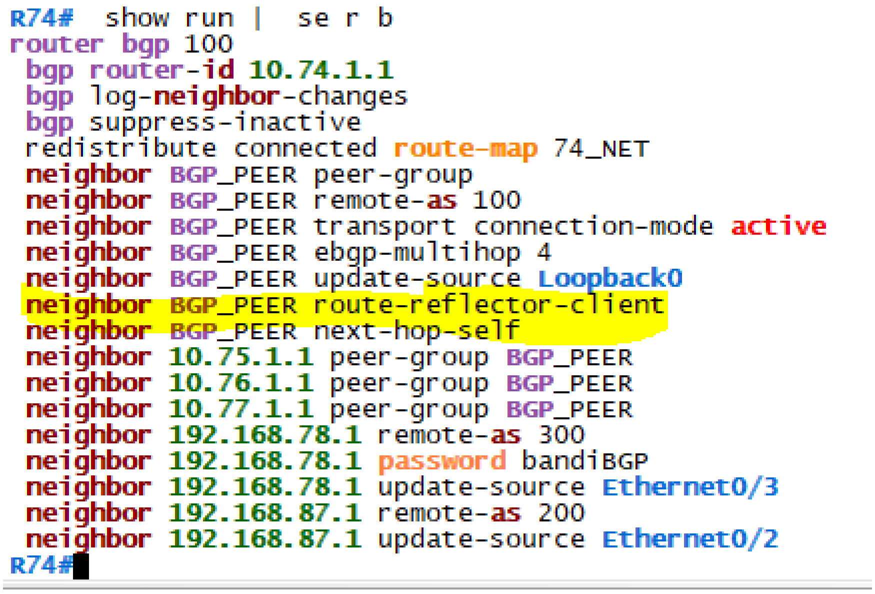
In the simplest of route-reflection designs, a central peering point is chosen for all devices in the iBGP domain, and all peers of this device are defined as clients.

Now we make R74 as Route-Reflector and R75,R76, R77 as clients

2Ways to configure- either use neighour command mention client as below :

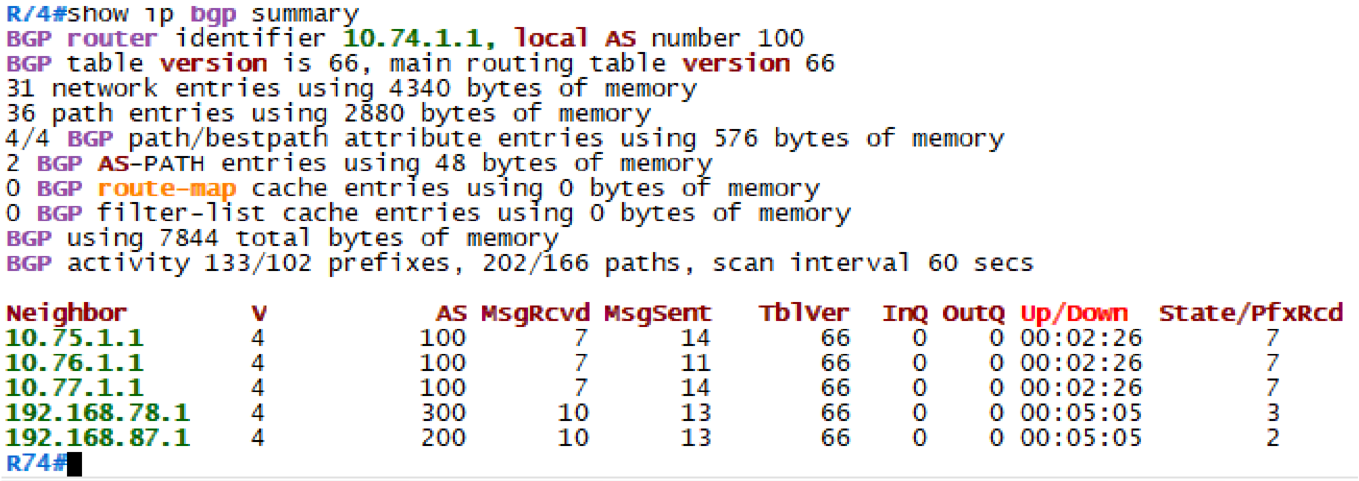
neighbor x.x.x.x route-reflector-client

or add route-reflector-client to peer-group as below



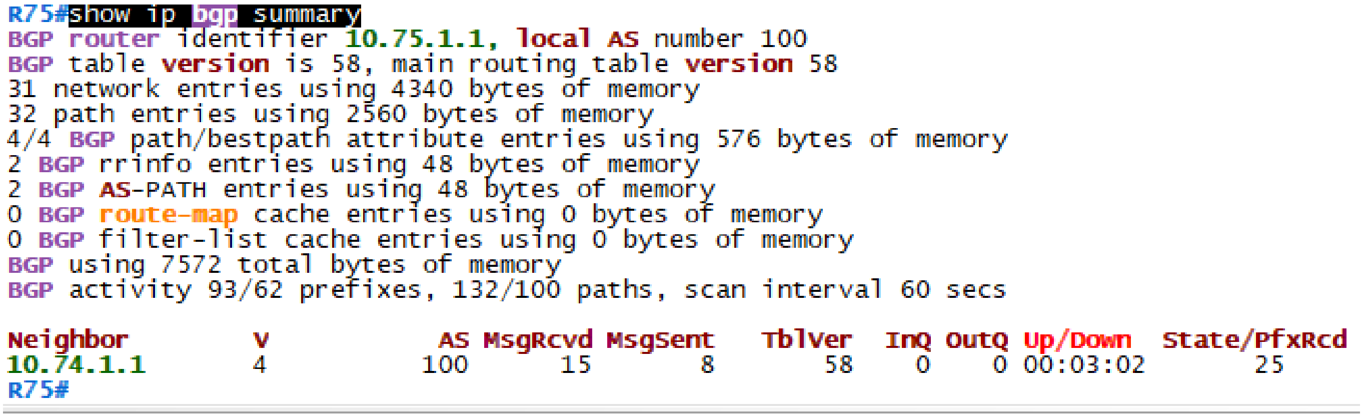
```
R74#  show run |  se r b
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER transport connection-mode active
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER route-reflector-client
 neighbor BGP_PEER next-hop-self
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
R74#
```

R74 has all peers

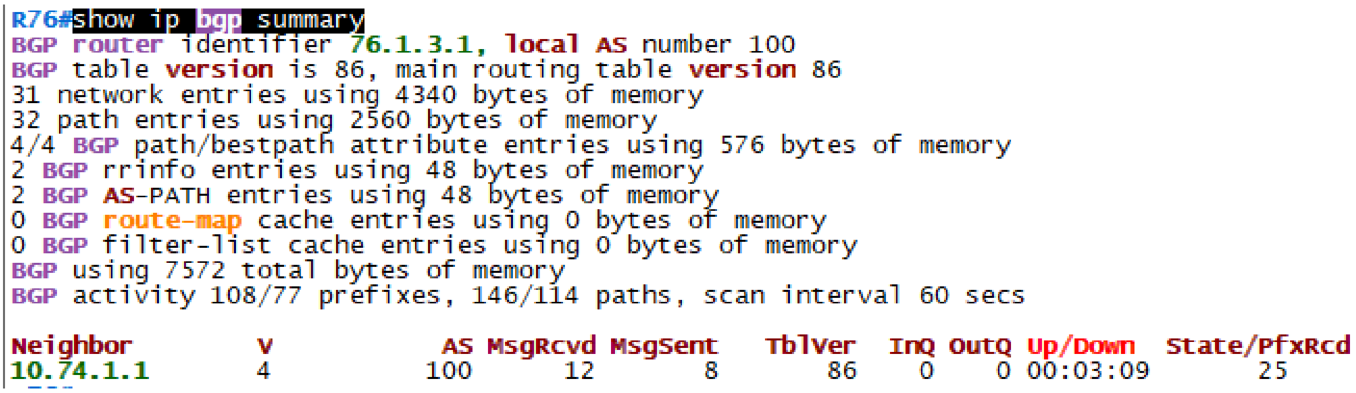


```
R74#show ip bgp summary
BGP router identifier 10.74.1.1, local AS number 100
BGP table version is 66, main routing table version 66
31 network entries using 4340 bytes of memory
36 path entries using 2880 bytes of memory
4/4 BGP path/bestpath attribute entries using 576 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 7844 total bytes of memory
BGP activity 133/102 prefixes, 202/166 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.75.1.1       4          100       7      14       66    0    0 00:02:26        7
10.76.1.1       4          100       7      11       66    0    0 00:02:26        7
10.77.1.1       4          100       7      14       66    0    0 00:02:26        7
192.168.78.1    4          300      10      13       66    0    0 00:05:05        3
192.168.87.1    4          200      10      13       66    0    0 00:05:05        2
R74#
```

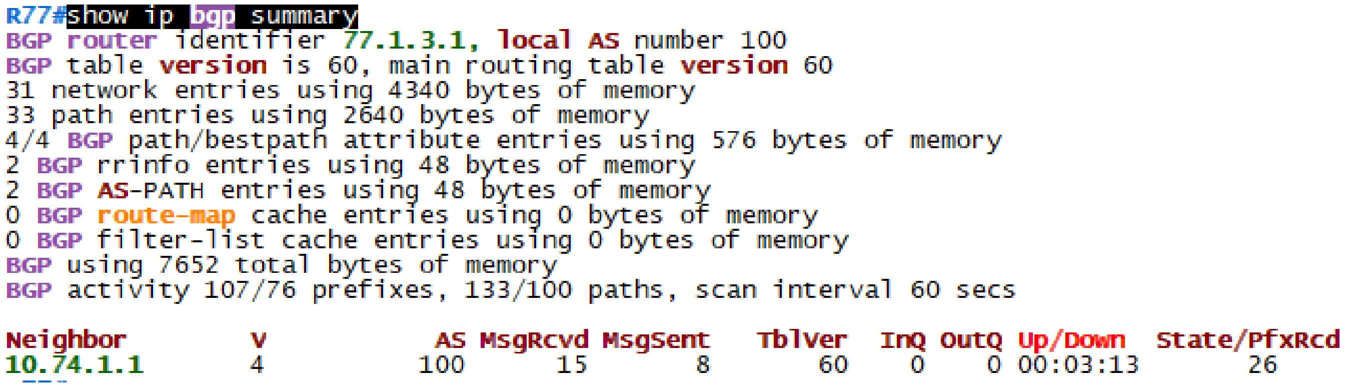
R75 R76 ad R77 only peer with R74



```
R75#show ip bgp summary
BGP router identifier 10.75.1.1, local AS number 100
BGP table version is 58, main routing table version 58
31 network entries using 4340 bytes of memory
32 path entries using 2560 bytes of memory
4/4 BGP path/bestpath attribute entries using 576 bytes of memory
2 BGP rrinfo entries using 48 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 7572 total bytes of memory
BGP activity 93/62 prefixes, 132/100 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.74.1.1       4          100      15       8       58    0    0 00:03:02       25
R75#
```

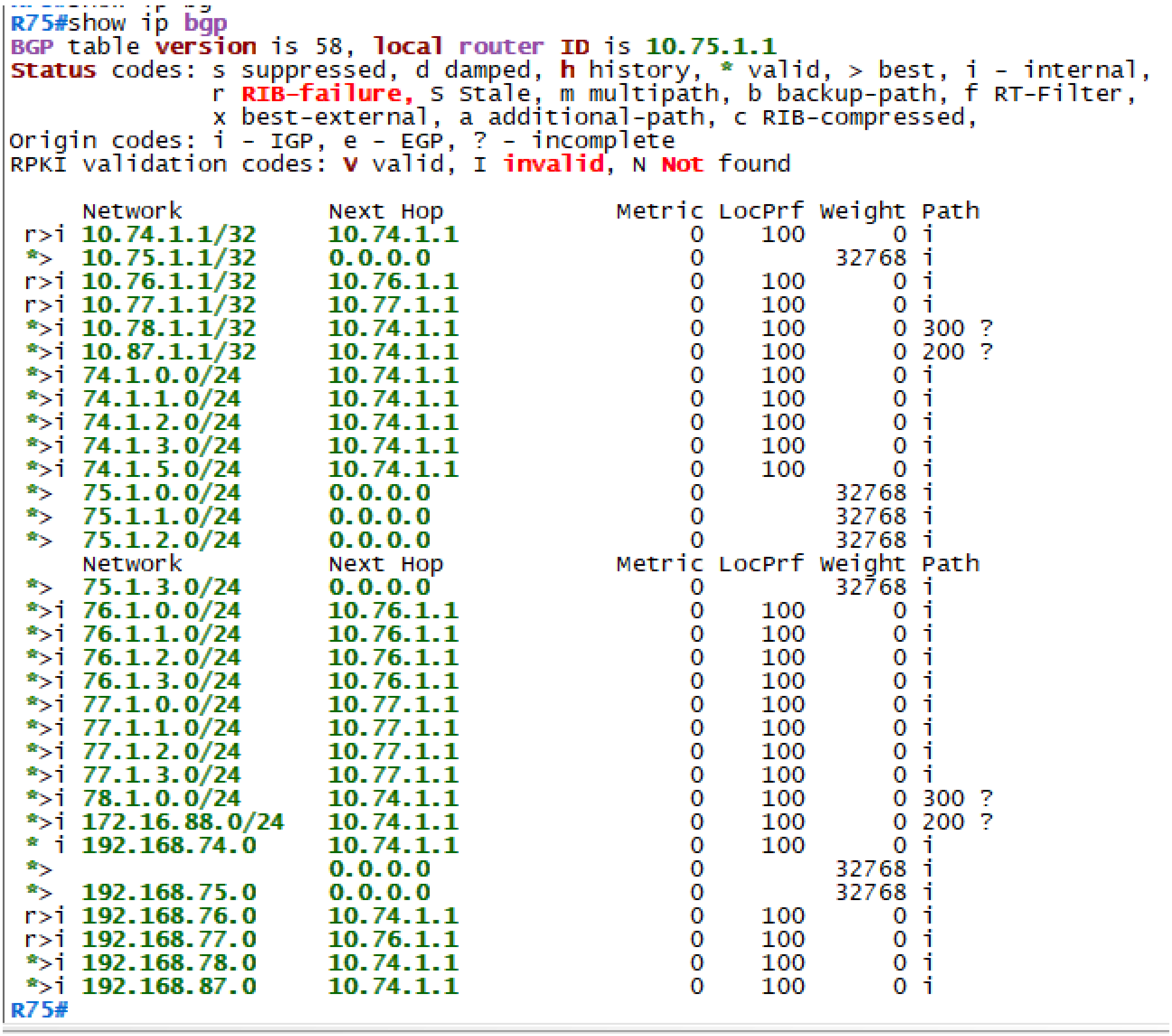


```
R76#show ip bgp summary
BGP router identifier 76.1.3.1, local AS number 100
BGP table version is 86, main routing table version 86
31 network entries using 4340 bytes of memory
32 path entries using 2560 bytes of memory
4/4 BGP path/bestpath attribute entries using 576 bytes of memory
2 BGP rrinfo entries using 48 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 7572 total bytes of memory
BGP activity 108/77 prefixes, 146/114 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.74.1.1       4          100      12       8       86    0    0 00:03:09       25
```



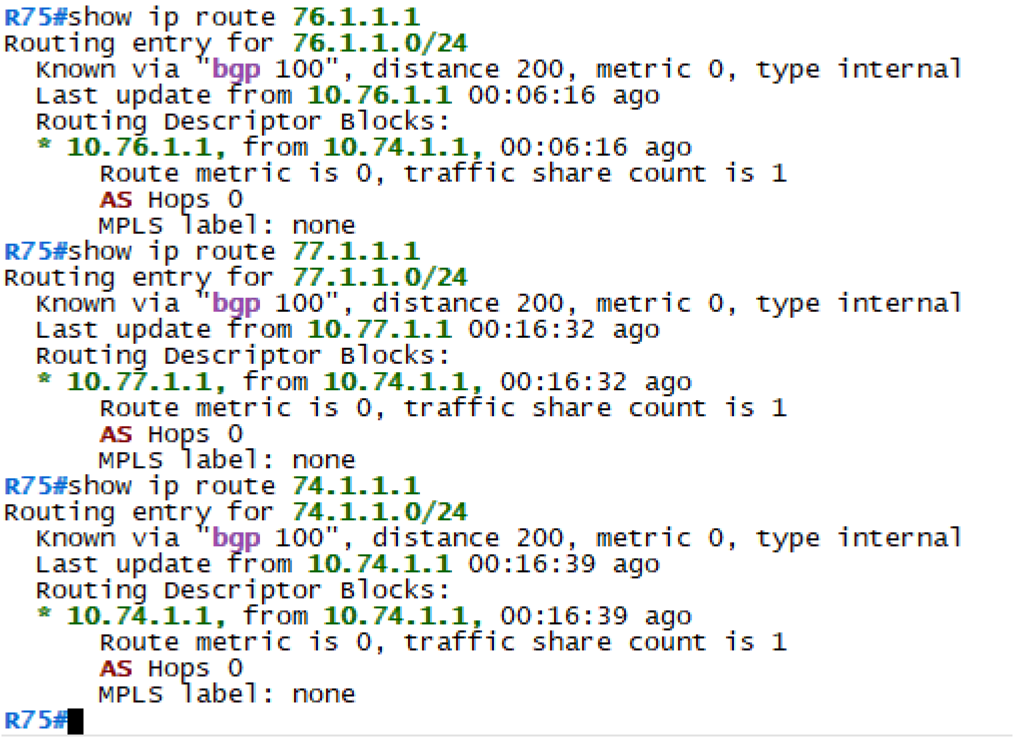
```
R77#show ip bgp summary
BGP router identifier 77.1.3.1, local AS number 100
BGP table version is 60, main routing table version 60
31 network entries using 4340 bytes of memory
33 path entries using 2640 bytes of memory
4/4 BGP path/bestpath attribute entries using 576 bytes of memory
2 BGP rrinfo entries using 48 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 7652 total bytes of memory
BGP activity 107/76 prefixes, 133/100 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.74.1.1       4          100      15       8       60    0    0 00:03:13       26
```

And routes learned from other peers



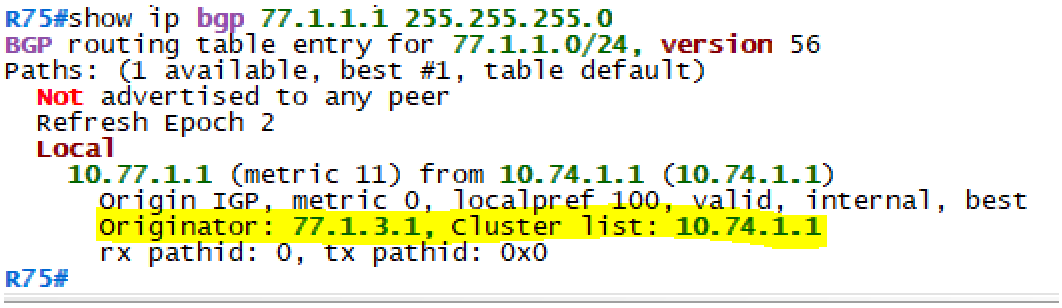
```
R75#show ip bgp
BGP table version is 58, local router ID is 10.75.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 r>i 10.74.1.1/32     10.74.1.1                0    100      0 i
 *>  10.75.1.1/32     0.0.0.0                  0         32768 i
 r>i 10.76.1.1/32     10.76.1.1                0    100      0 i
 r>i 10.77.1.1/32     10.77.1.1                0    100      0 i
 *>i 10.78.1.1/32     10.74.1.1                0    100      0 300 ?
 *>i 10.87.1.1/32     10.74.1.1                0    100      0 200 ?
 *>i 74.1.0.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.1.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.2.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.3.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.5.0/24      10.74.1.1                0    100      0 i
 *>  75.1.0.0/24      0.0.0.0                  0         32768 i
 *>  75.1.1.0/24      0.0.0.0                  0         32768 i
 *>  75.1.2.0/24      0.0.0.0                  0         32768 i
     Network          Next Hop            Metric LocPrf Weight Path
 *>  75.1.3.0/24      0.0.0.0                  0         32768 i
 *>i 76.1.0.0/24      10.76.1.1                0    100      0 i
 *>i 76.1.1.0/24      10.76.1.1                0    100      0 i
 *>i 76.1.2.0/24      10.76.1.1                0    100      0 i
 *>i 76.1.3.0/24      10.76.1.1                0    100      0 i
 *>i 77.1.0.0/24      10.77.1.1                0    100      0 i
 *>i 77.1.1.0/24      10.77.1.1                0    100      0 i
 *>i 77.1.2.0/24      10.77.1.1                0    100      0 i
 *>i 77.1.3.0/24      10.77.1.1                0    100      0 i
 *>i 78.1.0.0/24      10.74.1.1                0    100      0 300 ?
 *>i 172.16.88.0/24   10.74.1.1                0    100      0 200 ?
 * i 192.168.74.0     10.74.1.1                0    100      0 i
 *>                   0.0.0.0                  0         32768 i
 *>  192.168.75.0     0.0.0.0                  0         32768 i
 r>i 192.168.76.0     10.74.1.1                0    100      0 i
 r>i 192.168.77.0     10.76.1.1                0    100      0 i
 *>i 192.168.78.0     10.74.1.1                0    100      0 i
 *>i 192.168.87.0     10.74.1.1                0    100      0 i
R75#
```

Observe that routes are learned from the orginal routers



```
R75#show ip route 76.1.1.1
Routing entry for 76.1.1.0/24
  Known via "bgp 100", distance 200, metric 0, type internal
  Last update from 10.76.1.1 00:06:16 ago
  Routing Descriptor Blocks:
  * 10.76.1.1, from 10.74.1.1, 00:06:16 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0
      MPLS label: none
R75#show ip route 77.1.1.1
Routing entry for 77.1.1.0/24
  Known via "bgp 100", distance 200, metric 0, type internal
  Last update from 10.77.1.1 00:16:32 ago
  Routing Descriptor Blocks:
  * 10.77.1.1, from 10.74.1.1, 00:16:32 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0
      MPLS label: none
R75#show ip route 74.1.1.1
Routing entry for 74.1.1.0/24
  Known via "bgp 100", distance 200, metric 0, type internal
  Last update from 10.74.1.1 00:16:39 ago
  Routing Descriptor Blocks:
  * 10.74.1.1, from 10.74.1.1, 00:16:39 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0
      MPLS label: none
R75#
```

Verification

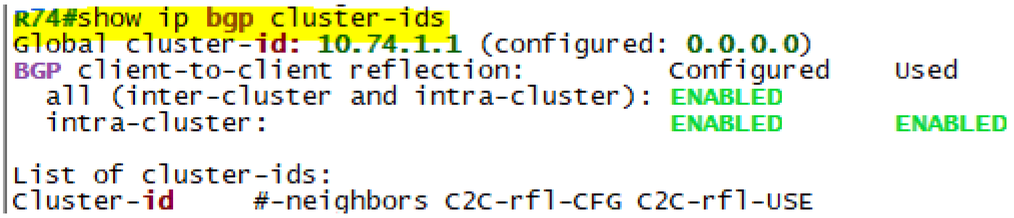


```
R75#show ip bgp 77.1.1.1 255.255.255.0
BGP routing table entry for 77.1.1.0/24, version 56
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 2
  Local
    10.77.1.1 (metric 11) from 10.74.1.1 (10.74.1.1)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator: 77.1.3.1, Cluster list: 10.74.1.1
      rx pathid: 0, tx pathid: 0x0
R75#
```

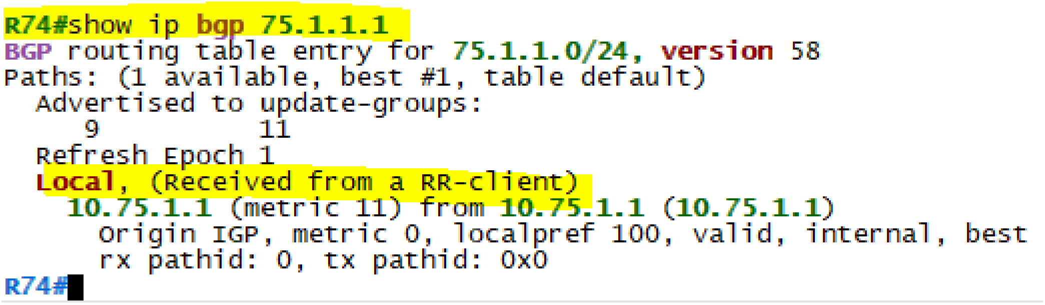
Route-Reflector does not have loop prevention, so we need to implement loop prevention 2 mothods.

1. BGP router-id (bgp router-id x.x.x.x)
2. Cluster ID (bgp cluster-id x.x.x.x)

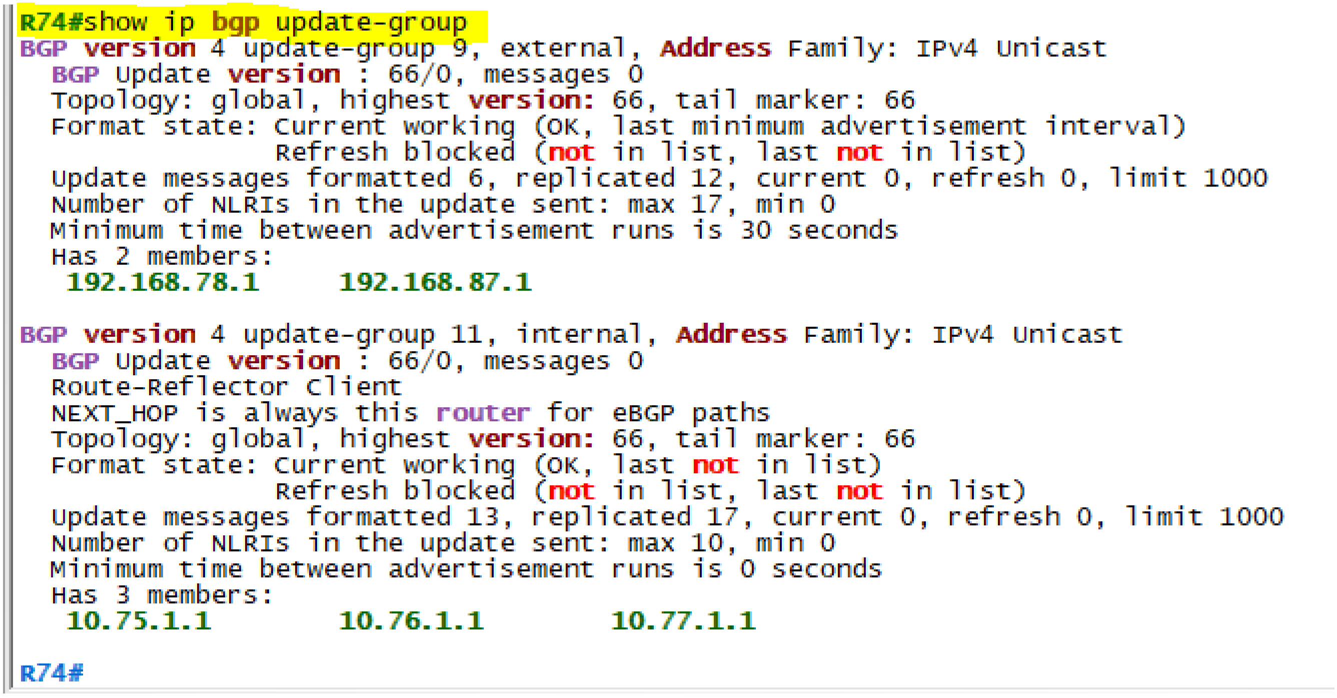
Route Reflector Clusters



```
R74#show ip bgp cluster-ids
Global cluster-id: 10.74.1.1 (configured: 0.0.0.0)
BGP client-to-client reflection:         Configured    Used
  all (inter-cluster and intra-cluster): ENABLED
  intra-cluster:                         ENABLED       ENABLED

List of cluster-ids:
Cluster-id     #-neighbors C2C-rfl-CFG C2C-rfl-USE
```



```
R74#show ip bgp 75.1.1.1
BGP routing table entry for 75.1.1.0/24, version 58
Paths: (1 available, best #1, table default)
  Advertised to update-groups:
     9          11
  Refresh Epoch 1
  Local, (Received from a RR-client)
    10.75.1.1 (metric 11) from 10.75.1.1 (10.75.1.1)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      rx pathid: 0, tx pathid: 0x0
R74#
```



```
R74#show ip bgp update-group
BGP version 4 update-group 9, external, Address Family: IPv4 Unicast
  BGP Update version : 66/0, messages 0
  Topology: global, highest version: 66, tail marker: 66
  Format state: Current working (OK, last minimum advertisement interval)
                Refresh blocked (not in list, last not in list)
  Update messages formatted 6, replicated 12, current 0, refresh 0, limit 1000
  Number of NLRIs in the update sent: max 17, min 0
  Minimum time between advertisement runs is 30 seconds
  Has 2 members:
   192.168.78.1     192.168.87.1

BGP version 4 update-group 11, internal, Address Family: IPv4 Unicast
  BGP Update version : 66/0, messages 0
  Route-Reflector Client
  NEXT_HOP is always this router for eBGP paths
  Topology: global, highest version: 66, tail marker: 66
  Format state: Current working (OK, last not in list)
                Refresh blocked (not in list, last not in list)
  Update messages formatted 13, replicated 17, current 0, refresh 0, limit 1000
  Number of NLRIs in the update sent: max 10, min 0
  Minimum time between advertisement runs is 0 seconds
  Has 3 members:
   10.75.1.1        10.76.1.1        10.77.1.1

R74#
```

For reference clustering

https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/200153-BGP-Route-Reflection-and-Multiple-Cluste.html

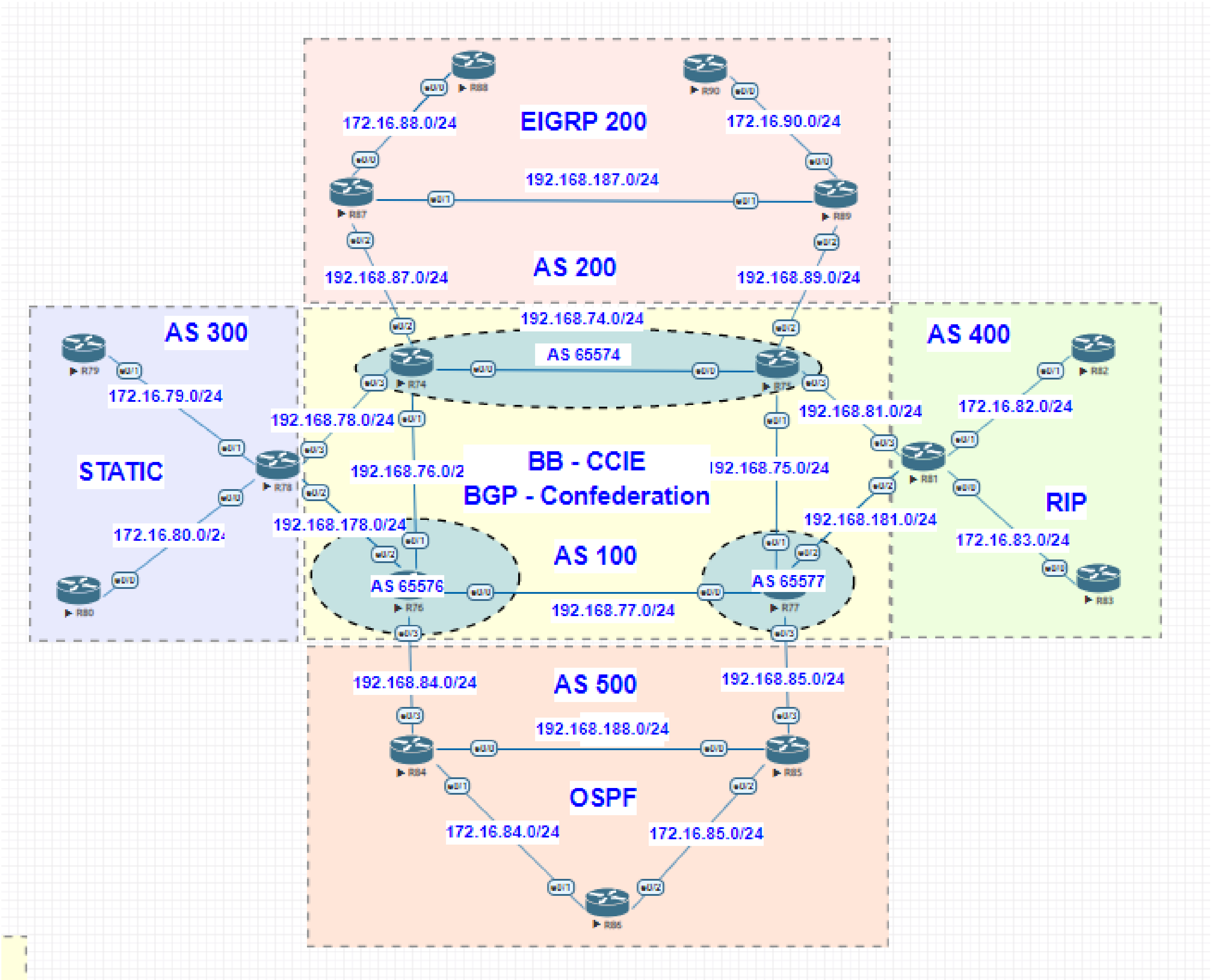
2.7.e [ii] Confederations

BGP Confederation is to divide an AS into multiple AS's and assign the whole group to a single confederation. Each AS alone has iBGP fully meshed. To the outside world, the confederation appears to be a single AS.

The configuration steps for BGP confederations are very straight forward.

- Define the confederation ID (The real AS number)
- Define the confederation peer
- Define the neighbors

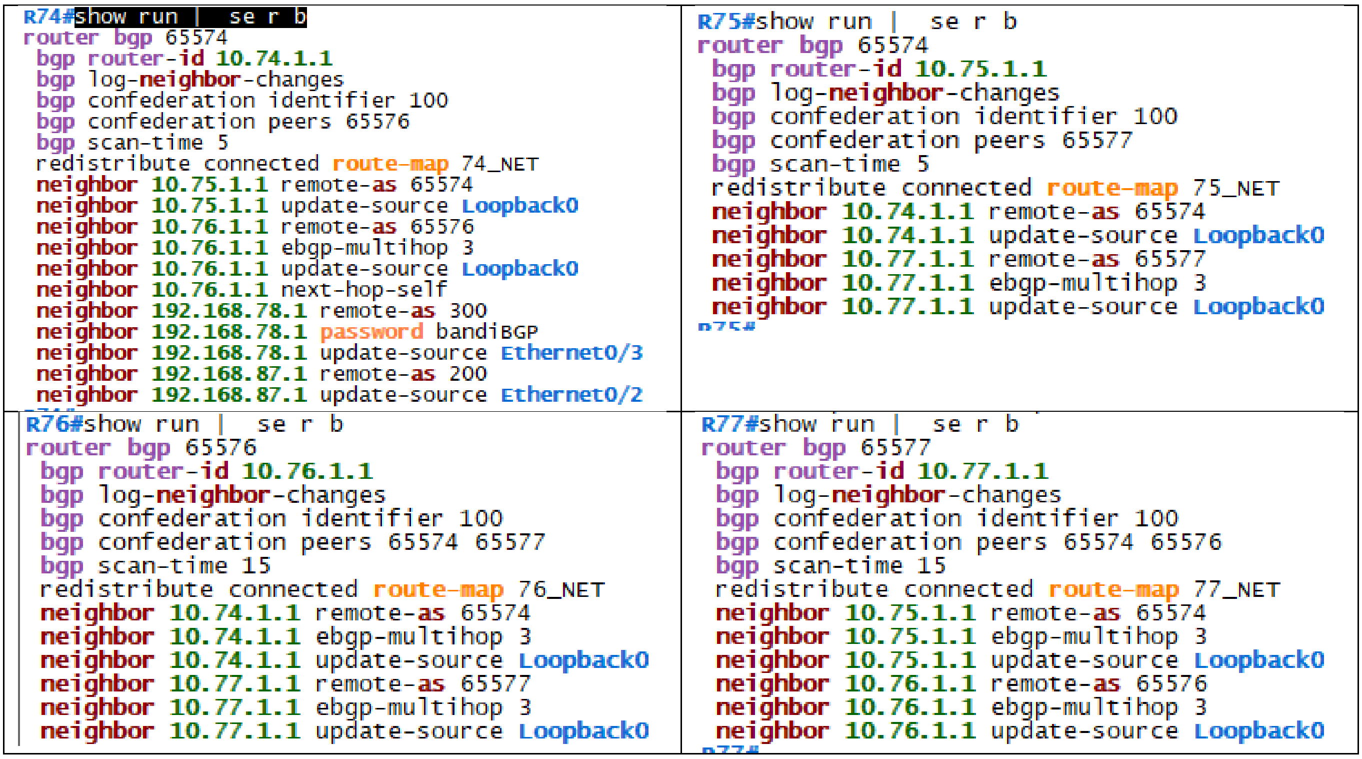
Even though the engineer can break up the autonomous system into smaller, more manageable autonomous systems, the only thing that changes is the AS_CONFED_SEQ and AS_CONFED_SET values.(If aggregation is used) Other attributes, such as med, local preference, and next hop are not changed. When designing BGP confederations, it's common practice to use the private AS range to denote a sub-autonomous system. The private AS numbers available for use is 64512 – 65535.



In the example lab, we configure confederation,
R74, R75 belong to AS 65574
R76 belong to AS 65576
R77 belong to AS 65577
They are part of AS 100

Note : "directly connected check" is performed for all eBGP peers having either **ebgp-multihop 1** or no **ebgp-multihop** configured. If **ebgp-multihop 2** or more is configured for a neighbor, this check is not performed

Here is the config



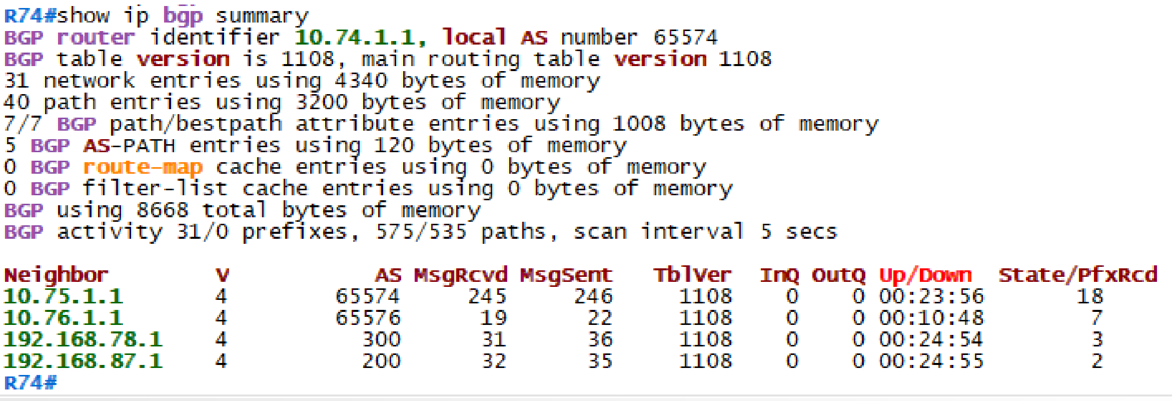
```
R74#show run |  se r b
router bgp 65574
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp confederation identifier 100
 bgp confederation peers 65576
 bgp scan-time 5
 redistribute connected route-map 74_NET
 neighbor 10.75.1.1 remote-as 65574
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.76.1.1 remote-as 65576
 neighbor 10.76.1.1 ebgp-multihop 3
 neighbor 10.76.1.1 update-source Loopback0
 neighbor 10.76.1.1 next-hop-self
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
R74#
```

```
R75#show run |  se r b
router bgp 65574
 bgp router-id 10.75.1.1
 bgp log-neighbor-changes
 bgp confederation identifier 100
 bgp confederation peers 65577
 bgp scan-time 5
 redistribute connected route-map 75_NET
 neighbor 10.74.1.1 remote-as 65574
 neighbor 10.74.1.1 update-source Loopback0
 neighbor 10.77.1.1 remote-as 65577
 neighbor 10.77.1.1 ebgp-multihop 3
 neighbor 10.77.1.1 update-source Loopback0
R75#
```

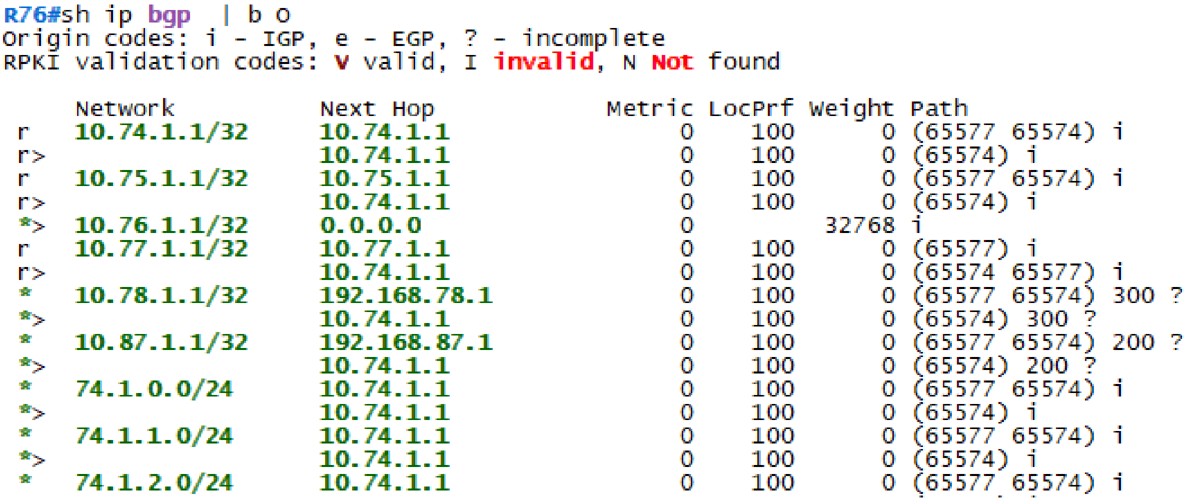
```
R76#show run |  se r b
router bgp 65576
 bgp router-id 10.76.1.1
 bgp log-neighbor-changes
 bgp confederation identifier 100
 bgp confederation peers 65574 65577
 bgp scan-time 15
 redistribute connected route-map 76_NET
 neighbor 10.74.1.1 remote-as 65574
 neighbor 10.74.1.1 ebgp-multihop 3
 neighbor 10.74.1.1 update-source Loopback0
 neighbor 10.77.1.1 remote-as 65577
 neighbor 10.77.1.1 ebgp-multihop 3
 neighbor 10.77.1.1 update-source Loopback0
```

```
R77#show run |  se r b
router bgp 65577
 bgp router-id 10.77.1.1
 bgp log-neighbor-changes
 bgp confederation identifier 100
 bgp confederation peers 65574 65576
 bgp scan-time 15
 redistribute connected route-map 77_NET
 neighbor 10.75.1.1 remote-as 65574
 neighbor 10.75.1.1 ebgp-multihop 3
 neighbor 10.75.1.1 update-source Loopback0
 neighbor 10.76.1.1 remote-as 65576
 neighbor 10.76.1.1 ebgp-multihop 3
 neighbor 10.76.1.1 update-source Loopback0
R77#
```

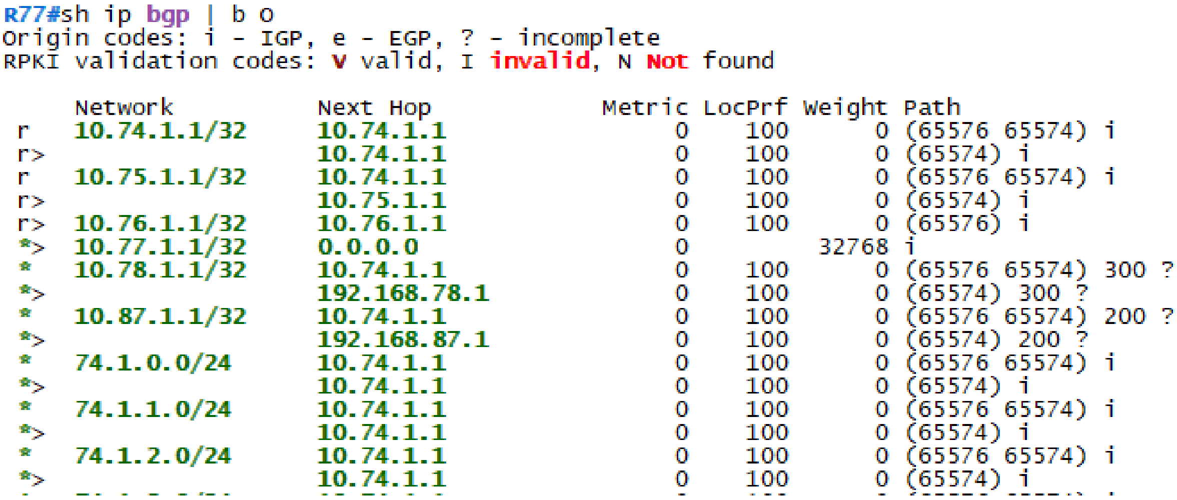
Verification :



```
R74#show ip bgp summary
BGP router identifier 10.74.1.1, local AS number 65574
BGP table version is 1108, main routing table version 1108
31 network entries using 4340 bytes of memory
40 path entries using 3200 bytes of memory
7/7 BGP path/bestpath attribute entries using 1008 bytes of memory
5 BGP AS-PATH entries using 120 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 8668 total bytes of memory
BGP activity 31/0 prefixes, 575/535 paths, scan interval 5 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.75.1.1       4        65574     245     246     1108    0    0 00:23:56       18
10.76.1.1       4        65576      19      22     1108    0    0 00:10:48        7
192.168.78.1    4          300      31      36     1108    0    0 00:24:54        3
192.168.87.1    4          200      32      35     1108    0    0 00:24:55        2
R74#
```

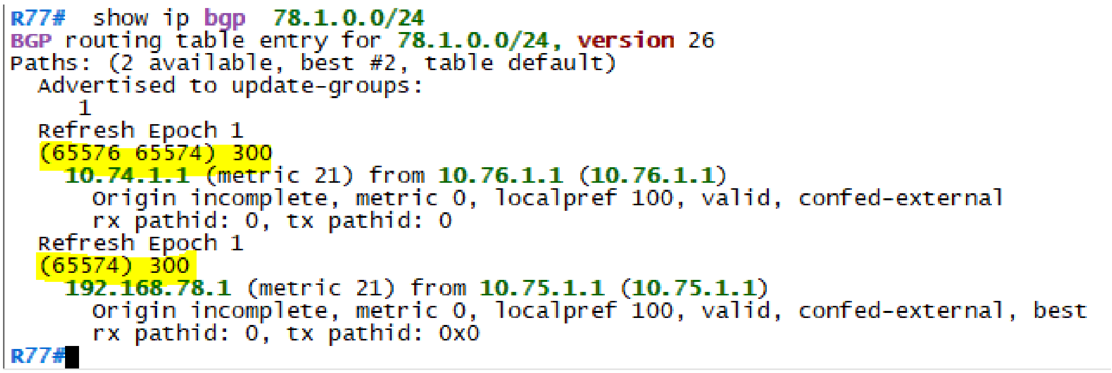


```
R76#sh ip bgp  | b o
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 r   10.74.1.1/32     10.74.1.1                0    100      0 (65577 65574) i
 r>                   10.74.1.1                0    100      0 (65574) i
 r   10.75.1.1/32     10.75.1.1                0    100      0 (65577 65574) i
 r>                   10.74.1.1                0    100      0 (65574) i
 *>  10.76.1.1/32     0.0.0.0                  0         32768 i
 r   10.77.1.1/32     10.77.1.1                0    100      0 (65577) i
 r>                   10.74.1.1                0    100      0 (65574 65577) i
 *   10.78.1.1/32     192.168.78.1             0    100      0 (65577 65574) 300 ?
 *>                   10.74.1.1                0    100      0 (65574) 300 ?
 *   10.87.1.1/32     192.168.87.1             0    100      0 (65577 65574) 200 ?
 *>                   10.74.1.1                0    100      0 (65574) 200 ?
 *   74.1.0.0/24      10.74.1.1                0    100      0 (65577 65574) i
 *>                   10.74.1.1                0    100      0 (65574) i
 *   74.1.1.0/24      10.74.1.1                0    100      0 (65577 65574) i
 *>                   10.74.1.1                0    100      0 (65574) i
 *   74.1.2.0/24      10.74.1.1                0    100      0 (65577 65574) i
```



```
R77#sh ip bgp | b o
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 r   10.74.1.1/32     10.74.1.1                0    100      0 (65576 65574) i
 r>                   10.74.1.1                0    100      0 (65574) i
 r   10.75.1.1/32     10.74.1.1                0    100      0 (65576 65574) i
 r>                   10.75.1.1                0    100      0 (65574) i
 r>  10.76.1.1/32     10.76.1.1                0    100      0 (65576) i
 *>  10.77.1.1/32     0.0.0.0                  0         32768 i
 *   10.78.1.1/32     10.74.1.1                0    100      0 (65576 65574) 300 ?
 *>                   192.168.78.1             0    100      0 (65574) 300 ?
 *   10.87.1.1/32     10.74.1.1                0    100      0 (65576 65574) 200 ?
 *>                   192.168.87.1             0    100      0 (65574) 200 ?
 *   74.1.0.0/24      10.74.1.1                0    100      0 (65576 65574) i
 *>                   10.74.1.1                0    100      0 (65574) i
 *   74.1.1.0/24      10.74.1.1                0    100      0 (65576 65574) i
 *>                   10.74.1.1                0    100      0 (65574) i
 *   74.1.2.0/24      10.74.1.1                0    100      0 (65576 65574) i
 *>                   10.74.1.1                0    100      0 (65574) i
```

Best route selection:



```
R77#  show ip bgp  78.1.0.0/24
BGP routing table entry for 78.1.0.0/24, version 26
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  (65576 65574) 300
    10.74.1.1 (metric 21) from 10.76.1.1 (10.76.1.1)
      Origin incomplete, metric 0, localpref 100, valid, confed-external
      rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  (65574) 300
    192.168.78.1 (metric 21) from 10.75.1.1 (10.75.1.1)
      Origin incomplete, metric 0, localpref 100, valid, confed-external, best
      rx pathid: 0, tx pathid: 0x0
R77#
```

2.7.e [iii] Aggregation, AS set

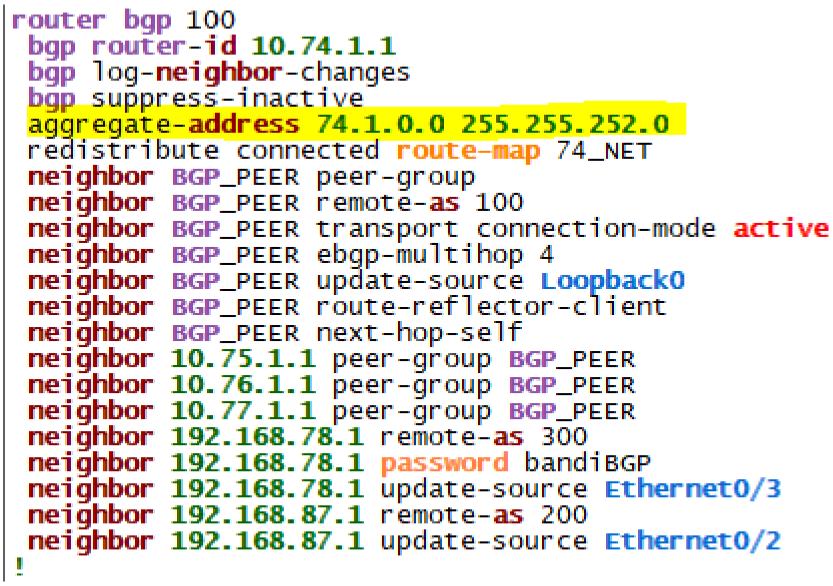
Route aggregation is the key for information hiding. It is critical to BGP because of the tremendous amount of routing information passed on the Internet. There are three basic ways to do summarization in BGP:

- Create a summary prefix in IGP and advertise it into BGP using the network command. This is usually accomplished by creating a static route to Null0 in the routing table of the advertising router. This is a common way to advertise local prefixes into BGP. However, you cannot summarize external BGP prefixes using this method.
- Use auto-summarization. As discussed in another task, this method summarizes networks to their classful boundaries and only applies to redistributed prefixes or when using the classful network command. It is not used in modern networks.
- Summarize prefixes found in BGP tables by using the aggregate-address command. This is the most flexible way to do summarization, because it may be applied to any paths learned by the BGP speaker.

The summarization in BGP can be done with the aggregate command. However, this aggregate command when enabled the router will automatically send this aggregated updated to all of its neighbor by default. Sometimes, we do not want to advertise the aggregated route to all of the neighbor. We use the unsuppressed-map command to help us accomplish this.

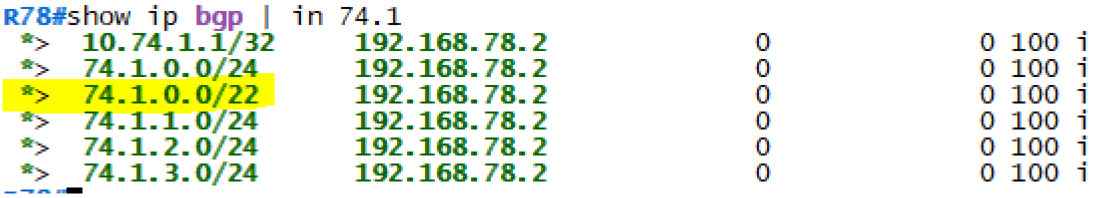
Suppress-Map
Unsuppress-Map
Inject-Map
Advertise-Map
Attribute-Map
Exist-Map
Non-exist-map

R74 we aggregate for the loopback address 74.1.0.0/22 to other AS



```
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 aggregate-address 74.1.0.0 255.255.252.0
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER transport connection-mode active
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER route-reflector-client
 neighbor BGP_PEER next-hop-self
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
!
```

Let’s check verification: (you see summary not blocking here for the specifics).

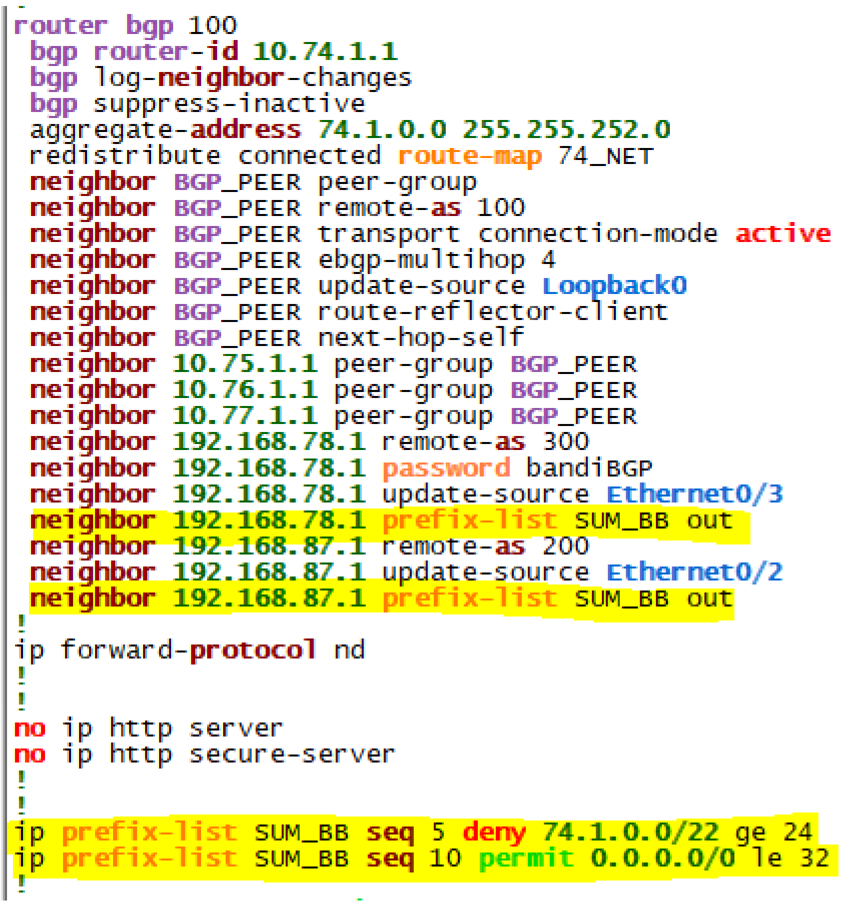


```
R78#show ip bgp | in 74.1
 *>  10.74.1.1/32     192.168.78.2          0             0 100 i
 *>  74.1.0.0/24      192.168.78.2          0             0 100 i
 *>  74.1.0.0/22      192.168.78.2          0             0 100 i
 *>  74.1.1.0/24      192.168.78.2          0             0 100 i
 *>  74.1.2.0/24      192.168.78.2          0             0 100 i
 *>  74.1.3.0/24      192.168.78.2          0             0 100 i
```

Now we observe that aggregated address send summarise and also sending other routes.
So BGP has flexibility to how we like to use summarisation. We can do inside and outside – specific to neighbour With filter

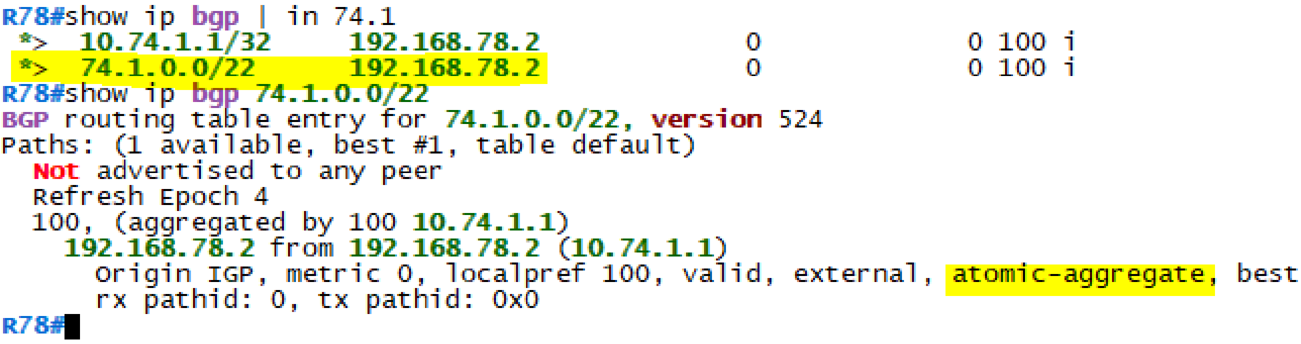
Aggregation with Filter:

Let’s filter the specifics route summarisation to R78

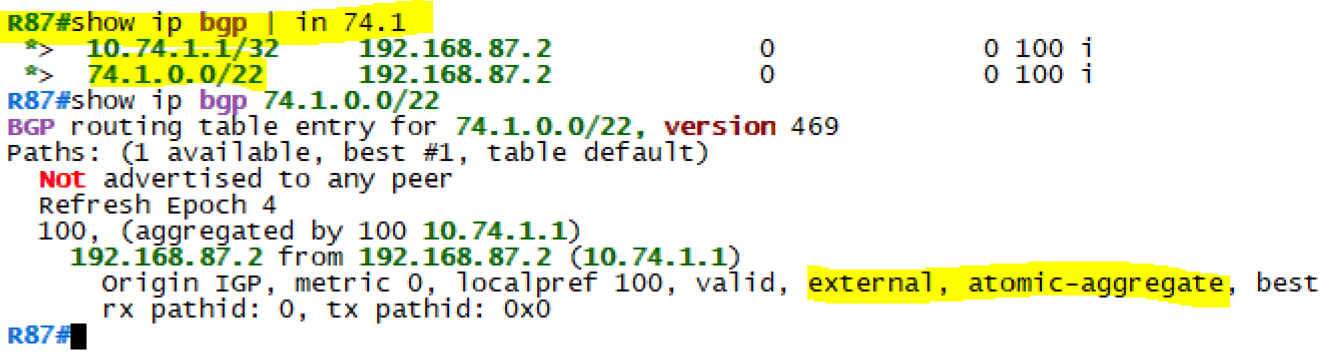


```
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 aggregate-address 74.1.0.0 255.255.252.0
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER transport connection-mode active
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER route-reflector-client
 neighbor BGP_PEER next-hop-self
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.78.1 prefix-list SUM_BB out
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
 neighbor 192.168.87.1 prefix-list SUM_BB out
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
ip prefix-list SUM_BB seq 5 deny 74.1.0.0/22 ge 24
ip prefix-list SUM_BB seq 10 permit 0.0.0.0/0 le 32
!
```

Now we verify AS 200 (R78) and AS 300 (R87)

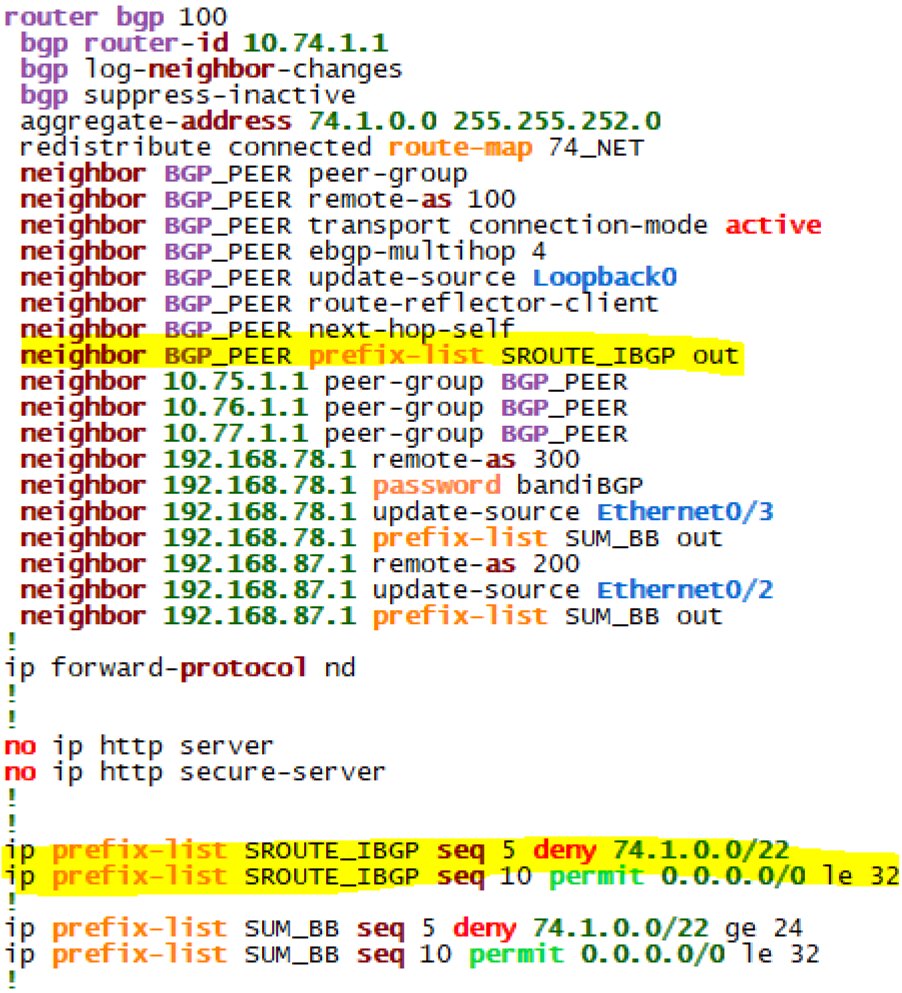


```
R78#show ip bgp | in 74.1
 *>  10.74.1.1/32      192.168.78.2             0             0 100 i
 *>  74.1.0.0/22       192.168.78.2             0             0 100 i
R78#show ip bgp 74.1.0.0/22
BGP routing table entry for 74.1.0.0/22, version 524
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 4
  100, (aggregated by 100 10.74.1.1)
    192.168.78.2 from 192.168.78.2 (10.74.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
      rx pathid: 0, tx pathid: 0x0
R78#
```



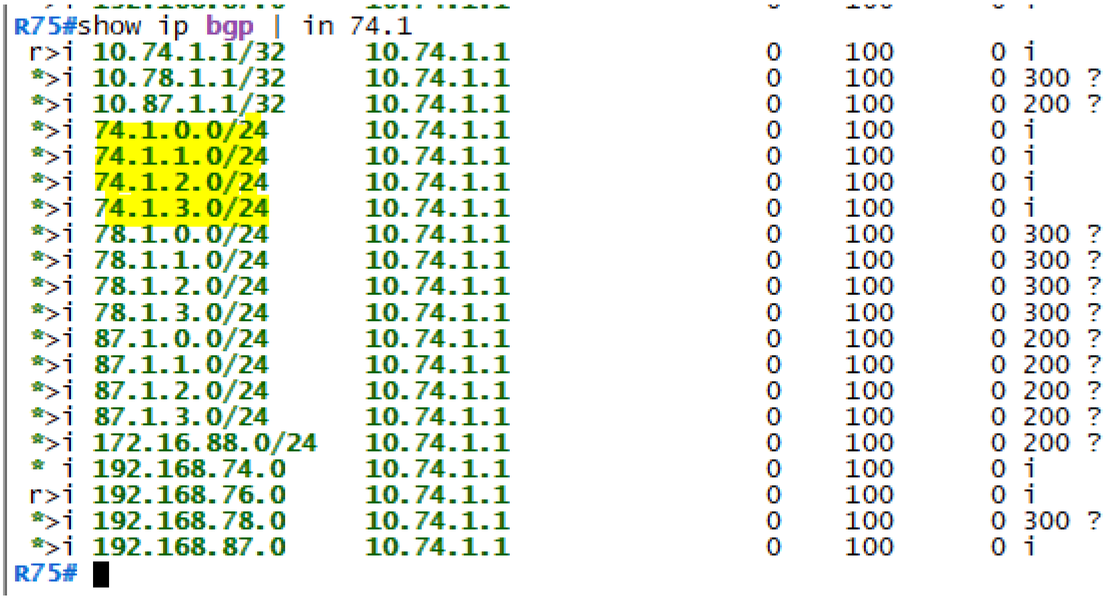
```
R87#show ip bgp | in 74.1
 *>  10.74.1.1/32      192.168.87.2             0             0 100 i
 *>  74.1.0.0/22       192.168.87.2             0             0 100 i
R87#show ip bgp 74.1.0.0/22
BGP routing table entry for 74.1.0.0/22, version 469
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 4
  100, (aggregated by 100 10.74.1.1)
    192.168.87.2 from 192.168.87.2 (10.74.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
      rx pathid: 0, tx pathid: 0x0
R87#
```

For now, internal iBGP we send specific router and deny Summary.



```
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 aggregate-address 74.1.0.0 255.255.252.0
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER transport connection-mode active
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER route-reflector-client
 neighbor BGP_PEER next-hop-self
 neighbor BGP_PEER prefix-list SROUTE_IBGP out
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.78.1 prefix-list SUM_BB out
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
 neighbor 192.168.87.1 prefix-list SUM_BB out
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
ip prefix-list SROUTE_IBGP seq 5 deny 74.1.0.0/22
ip prefix-list SROUTE_IBGP seq 10 permit 0.0.0.0/0 le 32
!
ip prefix-list SUM_BB seq 5 deny 74.1.0.0/22 ge 24
ip prefix-list SUM_BB seq 10 permit 0.0.0.0/0 le 32
!
```

Verify on iBGP router R75 for the 74.x.x.x network routes. (you do not see summary route 74.1.0.0/22) compare to previous output, which has 74.1.0.0/22 and /24 both the routes on routing tables.

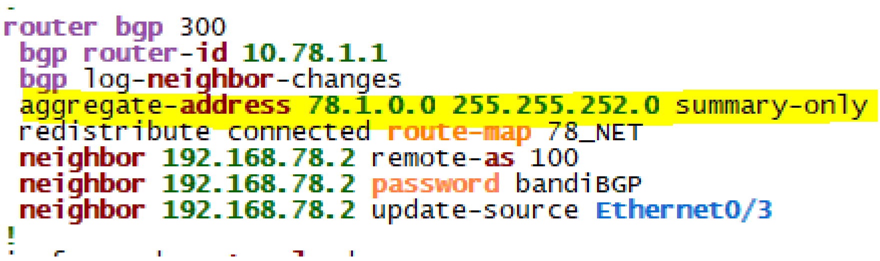


```
R75#show ip bgp | in 74.1
 r>i 10.74.1.1/32     10.74.1.1                0    100      0 i
 *>i 10.78.1.1/32     10.74.1.1                0    100      0 300 ?
 *>i 10.87.1.1/32     10.74.1.1                0    100      0 200 ?
 *>i 74.1.0.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.1.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.2.0/24      10.74.1.1                0    100      0 i
 *>i 74.1.3.0/24      10.74.1.1                0    100      0 i
 *>i 78.1.0.0/24      10.74.1.1                0    100      0 300 ?
 *>i 78.1.1.0/24      10.74.1.1                0    100      0 300 ?
 *>i 78.1.2.0/24      10.74.1.1                0    100      0 300 ?
 *>i 78.1.3.0/24      10.74.1.1                0    100      0 300 ?
 *>i 87.1.0.0/24      10.74.1.1                0    100      0 200 ?
 *>i 87.1.1.0/24      10.74.1.1                0    100      0 200 ?
 *>i 87.1.2.0/24      10.74.1.1                0    100      0 200 ?
 *>i 87.1.3.0/24      10.74.1.1                0    100      0 200 ?
 *>i 172.16.88.0/24   10.74.1.1                0    100      0 200 ?
 * i 192.168.74.0     10.74.1.1                0    100      0 i
 r>i 192.168.76.0     10.74.1.1                0    100      0 i
 *>i 192.168.78.0     10.74.1.1                0    100      0 300 ?
 *>i 192.168.87.0     10.74.1.1                0    100      0 i
R75#
```

Aggregation with automatic blocking for the specific routes (like IGP)

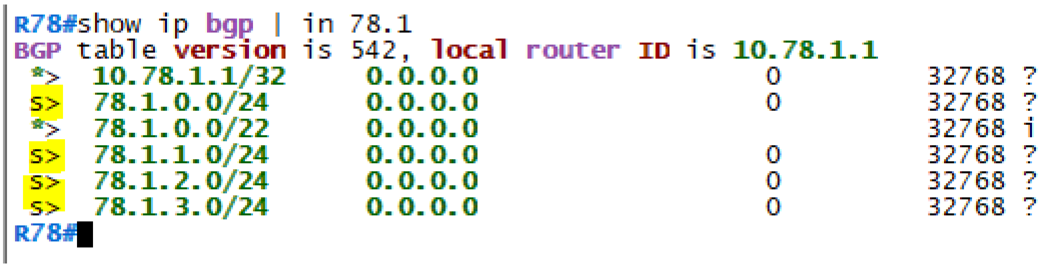
use the option summary-only after the aggregate-address command. The BGP process will automatically suppress advertisement of the prefixes in the BGP table encompassed by the new summary address. This is probably the most common use of the aggregation command, because usually only the summarized prefix should be advertised.

On R78 we do this summary and verify on R74 same.



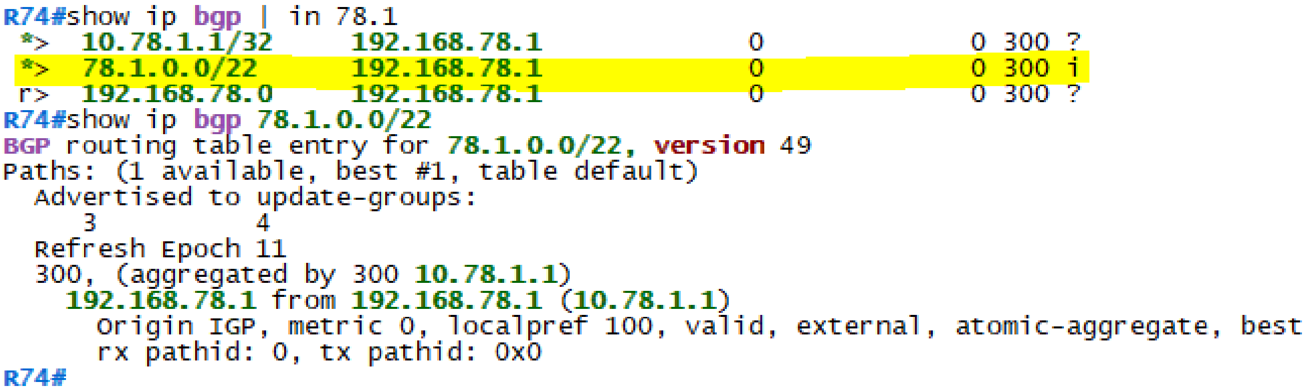
```
router bgp 300
 bgp router-id 10.78.1.1
 bgp log-neighbor-changes
 aggregate-address 78.1.0.0 255.255.252.0 summary-only
 redistribute connected route-map 78_NET
 neighbor 192.168.78.2 remote-as 100
 neighbor 192.168.78.2 password bandiBGP
 neighbor 192.168.78.2 update-source Ethernet0/3
!
```

You can do suppression with **s** in front of route



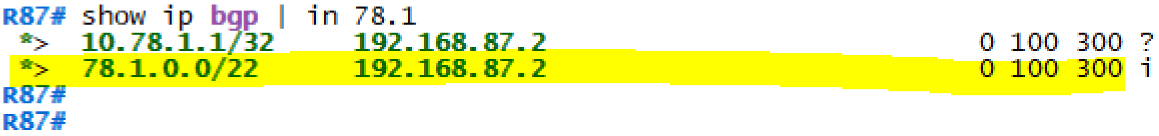
```
R78#show ip bgp | in 78.1
BGP table version is 542, local router ID is 10.78.1.1
 *>  10.78.1.1/32      0.0.0.0                  0         32768 ?
 s>  78.1.0.0/24       0.0.0.0                  0         32768 ?
 *>  78.1.0.0/22       0.0.0.0                            32768 i
 s>  78.1.1.0/24       0.0.0.0                  0         32768 ?
 s>  78.1.2.0/24       0.0.0.0                  0         32768 ?
 s>  78.1.3.0/24       0.0.0.0                  0         32768 ?
R78#
```

Now we verify on R74



```
R74#show ip bgp | in 78.1
 *>  10.78.1.1/32     192.168.78.1             0             0 300 ?
 *>  78.1.0.0/22      192.168.78.1             0             0 300 i
 r>  192.168.78.0     192.168.78.1             0             0 300 ?
R74#show ip bgp 78.1.0.0/22
BGP routing table entry for 78.1.0.0/22, version 49
Paths: (1 available, best #1, table default)
  Advertised to update-groups:
     3          4
  Refresh Epoch 11
  300, (aggregated by 300 10.78.1.1)
    192.168.78.1 from 192.168.78.1 (10.78.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
      rx pathid: 0, tx pathid: 0x0
R74#
```

Let's check in R87 AS 300



```
R87# show ip bgp | in 78.1
 *>  10.78.1.1/32     192.168.87.2                           0 100 300 ?
 *>  78.1.0.0/22      192.168.87.2                           0 100 300 i
R87#
R87#
```

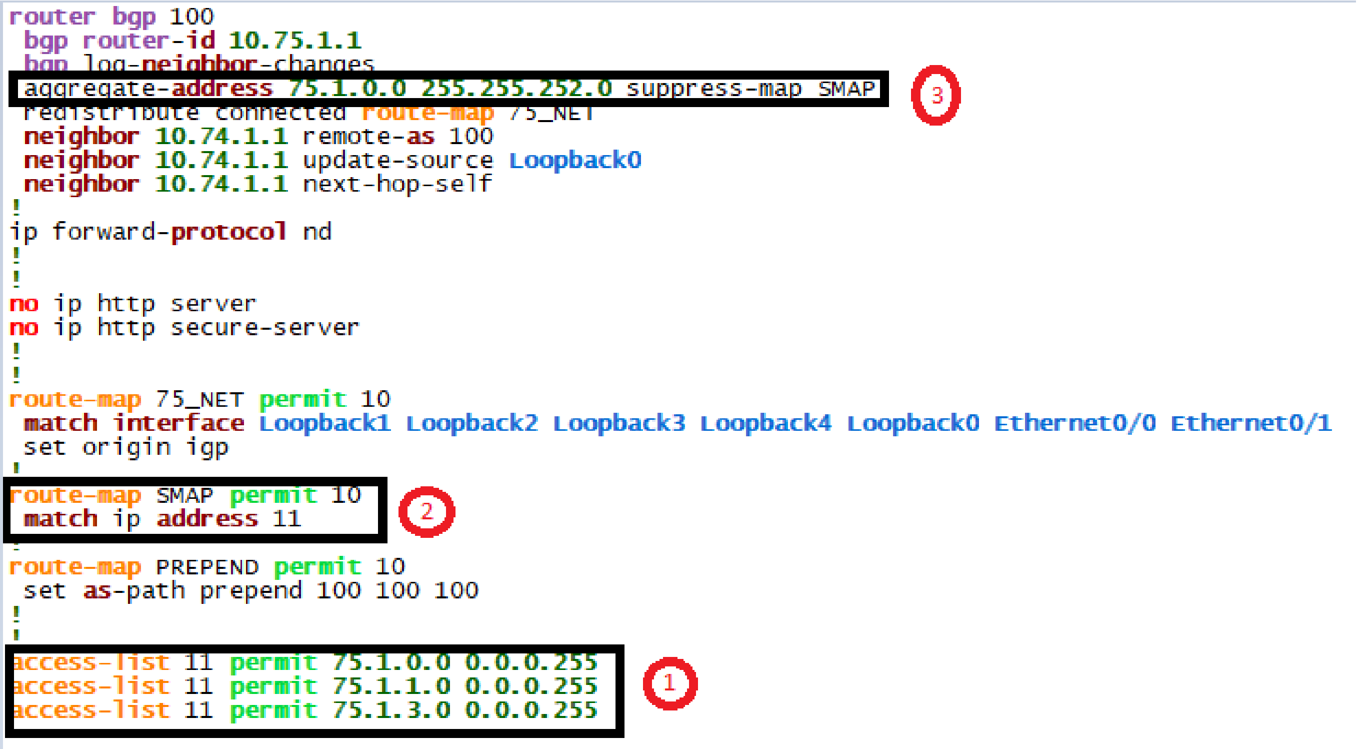
BGP Aggregation - Suppress Map (same like IGP Leak-map)

A suppress-map is a route map that's used for doing partial suppression of component routes of an aggregate. The default, when aggregating with BGP, is to advertise both the aggregate and all the component routes. If you want to advertise just the aggregate, you use the **summary-only** keyword at the end of the aggregate. But if you want to suppress some, but not all, of the component routes, you use a suppress-map.

R75 we configure and verify the same.

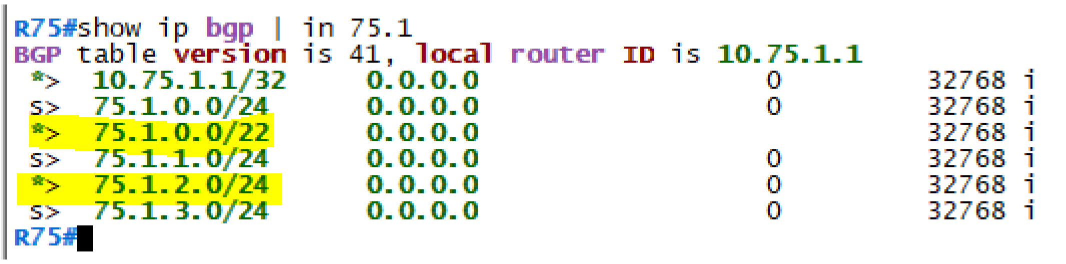
Steps :

1. Create an ACL to match the IP address not required to suppress or suppress
2. Create Route-map call acl creates
3. Use aggregate in BGP process with supress-map with route-map



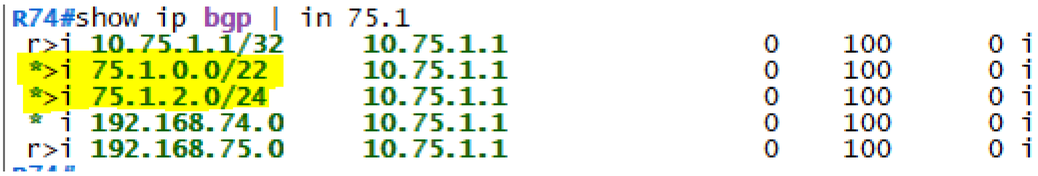
```
router bgp 100
 bgp router-id 10.75.1.1
 bgp log-neighbor-changes
 aggregate-address 75.1.0.0 255.255.252.0 suppress-map SMAP
 redistribute connected route-map 75_NET
 neighbor 10.74.1.1 remote-as 100
 neighbor 10.74.1.1 update-source Loopback0
 neighbor 10.74.1.1 next-hop-self
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
route-map 75_NET permit 10
 match interface Loopback1 Loopback2 Loopback3 Loopback4 Loopback0 Ethernet0/0 Ethernet0/1
 set origin igp
!
route-map SMAP permit 10
 match ip address 11
!
route-map PREPEND permit 10
 set as-path prepend 100 100 100
!
!
access-list 11 permit 75.1.0.0 0.0.0.255
access-list 11 permit 75.1.1.0 0.0.0.255
access-list 11 permit 75.1.3.0 0.0.0.255
```

Verification Local and other iBGP routers



```
R75#show ip bgp | in 75.1
BGP table version is 41, local router ID is 10.75.1.1
 *>   10.75.1.1/32     0.0.0.0                  0         32768 i
 s>   75.1.0.0/24      0.0.0.0                  0         32768 i
 *>   75.1.0.0/22      0.0.0.0                            32768 i
 s>   75.1.1.0/24      0.0.0.0                  0         32768 i
 *>   75.1.2.0/24      0.0.0.0                  0         32768 i
 s>   75.1.3.0/24      0.0.0.0                  0         32768 i
R75#
```

R74



```
R74#show ip bgp | in 75.1
 r>i 10.75.1.1/32     10.75.1.1                0    100      0 i
 *>i 75.1.0.0/22      10.75.1.1                0    100      0 i
 *>i 75.1.2.0/24      10.75.1.1                0    100      0 i
 * i 192.168.74.0     10.75.1.1                0    100      0 i
 r>i 192.168.75.0     10.75.1.1                0    100      0 i
```

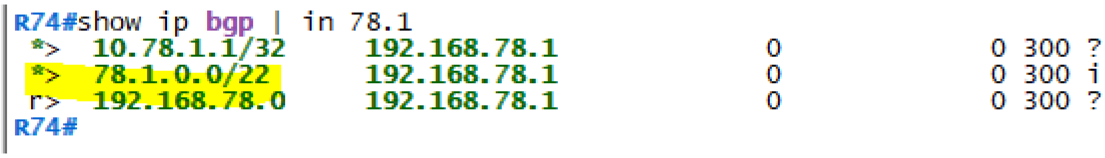
BGP Aggregation - Unsuppress Map

Unsuppress Map is opposite of a suppress-map. Let's say you have complete suppression via summary-only, or you globally suppressed a prefix with a suppress-map. Let's say you have 20 eBGP neighbors, but want to leak that specific suppressed prefix to only specific neighbour.

This scenario demonstrates one of the common uses for the aggregate-address command. Local networks are advertised into BGP and aggregated by the border BGP speakers.
It is often desirable to load-balance traffic ingress to the local AS, so that traffic to some subnets enters via one BGP peer and the other peer is used as the entry point for other subnets. Generally, to accomplish this, you need to advertise all specific prefixes on both uplinks and use AS_PATH prepending to modify prefixes' preference. This scheme implements load balancing and provides backup in case of any uplink failures.

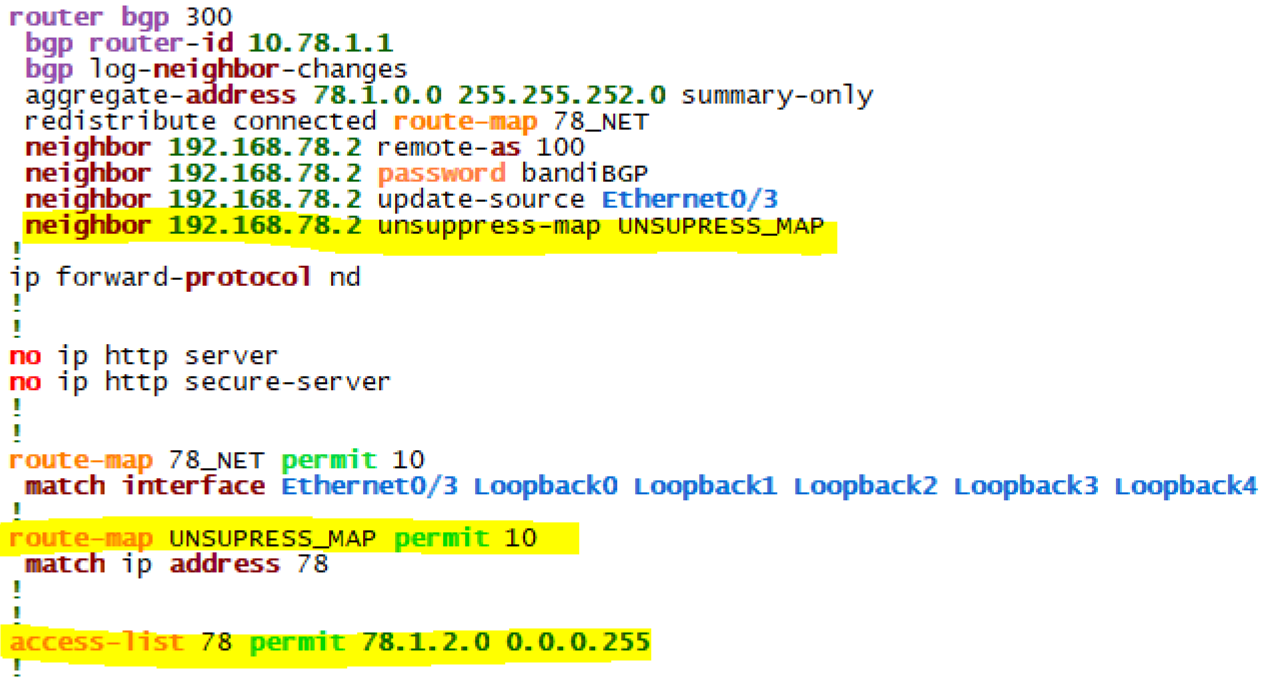
R78 AS 300 we have configured summary only for 76.1.0.0/22 for all routers, now we do unsuppress 78.1.2.0/24 to Router 74 and verify

Before supress on R74



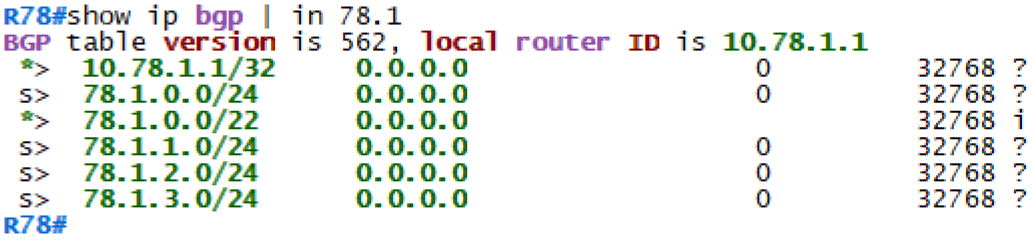
```
R74#show ip bgp | in 78.1
 *>  10.78.1.1/32     192.168.78.1             0             0 300 ?
 *>  78.1.0.0/22      192.168.78.1             0             0 300 i
 r>  192.168.78.0     192.168.78.1             0             0 300 ?
R74#
```

We configure unsupress on R78 and Verify the same.



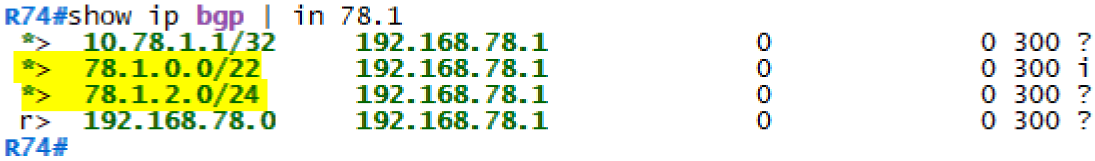
```
router bgp 300
 bgp router-id 10.78.1.1
 bgp log-neighbor-changes
 aggregate-address 78.1.0.0 255.255.252.0 summary-only
 redistribute connected route-map 78_NET
 neighbor 192.168.78.2 remote-as 100
 neighbor 192.168.78.2 password bandiBGP
 neighbor 192.168.78.2 update-source Ethernet0/3
 neighbor 192.168.78.2 unsuppress-map UNSUPRESS_MAP
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
route-map 78_NET permit 10
 match interface Ethernet0/3 Loopback0 Loopback1 Loopback2 Loopback3 Loopback4
!
route-map UNSUPRESS_MAP permit 10
 match ip address 78
!
!
access-list 78 permit 78.1.2.0 0.0.0.255
!
```

We can see all the specific routes on R78 still supressed

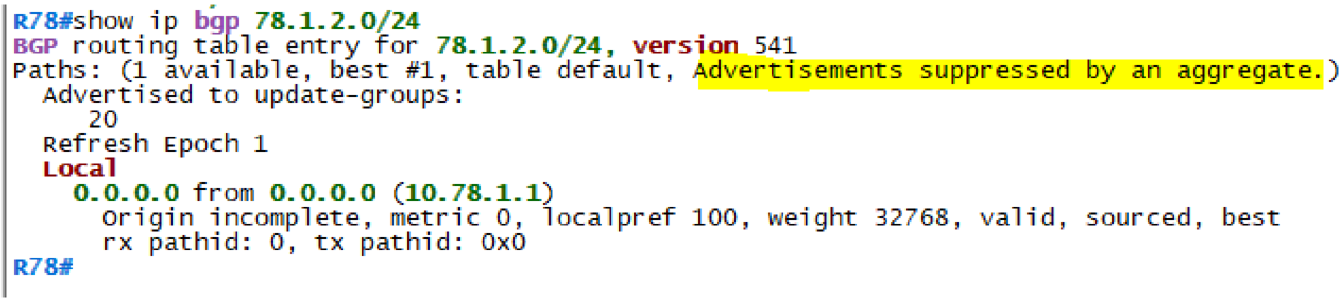


```
R78#show ip bgp | in 78.1
BGP table version is 562, local router ID is 10.78.1.1
 *>  10.78.1.1/32     0.0.0.0                  0         32768 ?
 s>  78.1.0.0/24      0.0.0.0                  0         32768 ?
 *>  78.1.0.0/22      0.0.0.0                            32768 i
 s>  78.1.1.0/24      0.0.0.0                  0         32768 ?
 s>  78.1.2.0/24      0.0.0.0                  0         32768 ?
 s>  78.1.3.0/24      0.0.0.0                  0         32768 ?
R78#
```

R74 we can see summary and unsupress route



```
R74#show ip bgp | in 78.1
 *>  10.78.1.1/32     192.168.78.1             0             0 300 ?
 *>  78.1.0.0/22      192.168.78.1             0             0 300 i
 *>  78.1.2.0/24      192.168.78.1             0             0 300 ?
 r>  192.168.78.0     192.168.78.1             0             0 300 ?
R74#
```



```
R78#show ip bgp 78.1.2.0/24
BGP routing table entry for 78.1.2.0/24, version 541
Paths: (1 available, best #1, table default, Advertisements suppressed by an aggregate.)
  Advertised to update-groups:
     20
  Refresh Epoch 1
  Local
    0.0.0.0 from 0.0.0.0 (10.78.1.1)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
      rx pathid: 0, tx pathid: 0x0
R78#
```

BGP Aggregation - AS-Set

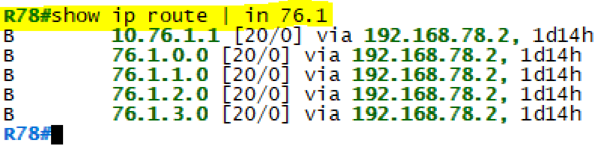
When advertising aggregates in BGP, it’s possible to specify the ‘as-set’ command as part of the aggregate. This forces the BGP process to advertise the prefix with a list of AS numbers that make up the aggregate.

It is important to remember that aggregation hides information previously found in the specific prefixes. This includes all attributes, such as NEXT_HOP, AS_PATH, and so on. The new prefix appears to be originated from within the local AS where aggregation is perfpormed. This causes no problems if all specific prefixes belong to the local AS. However, when you summarize prefixes learned from other ASs, information hiding may result in the following dangerous consequences:
Suboptimal routing, caused by loss of path information, such as AS_PATH, MED and so on.
Routing loops, because removing the AS_PATH attribute and replacing it with an empty list prevents the BGP loop-detection mechanism from working properly.
The second issue is more dangerous. To prevent it, it is possible to insert a special new member into the AS_PATH of the newly created summary prefix. This element is called AS_SET and contains the AS numbers found in all AS_PATHs of the specific prefixes. This list of AS numbers is unordered, unlike the regular AS_SEQUENCE element. Its only use is for routing loop prevention; when BGP receives a prefix, it scans the AS_PATH attribute. If the local AS number is found in any of the AS_SET or AS_SEQUENCE elements, the prefix is dropped.
By default, the aggregated address in BGP will not include the AS-Set information.
To force the use of this information, specify the as-set option as follows: aggregateaddress <subnet> <mask> as-set .

BGP Filtering with Standard Access-Lists

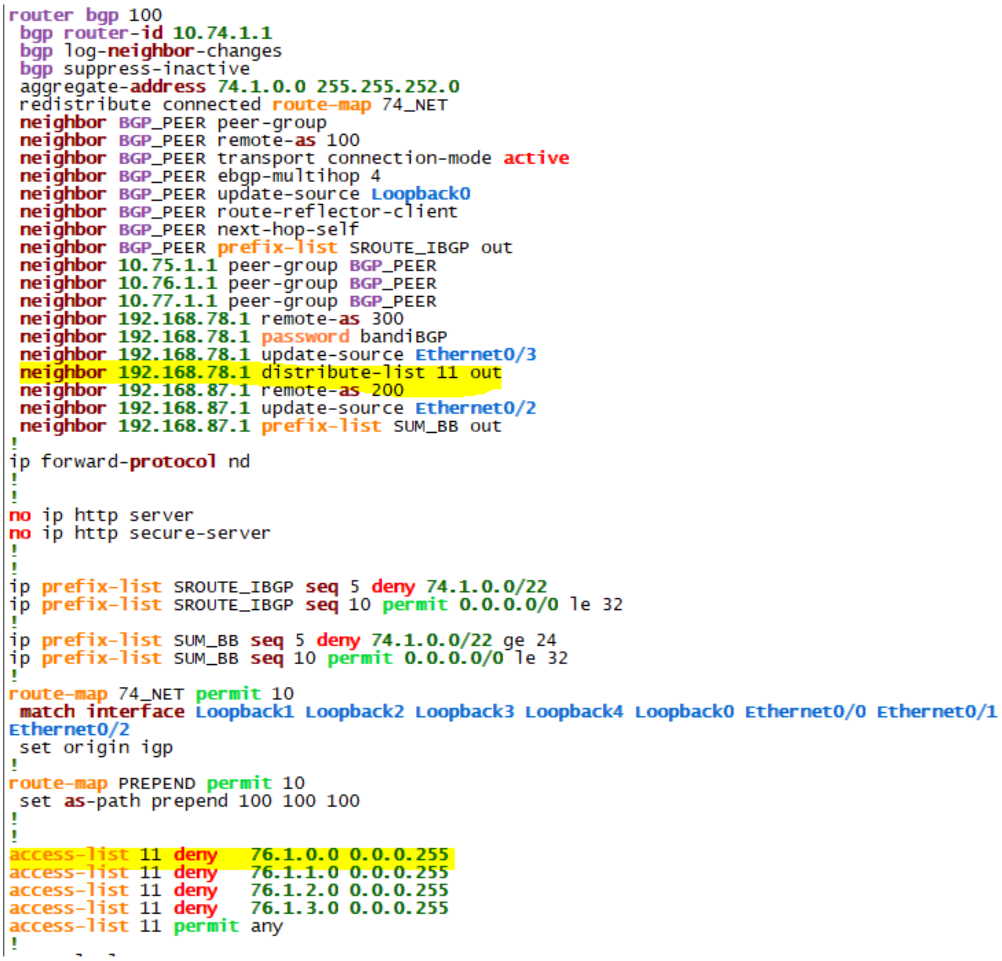
We use here R74 to filter router learn from R76 towards R78 that is AS 300

Before we apply the standard acl filter lets verify on R78 routes related R76



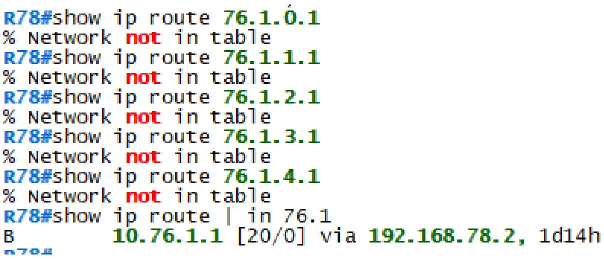
```
R78#show ip route | in 76.1
B        10.76.1.1 [20/0] via 192.168.78.2, 1d14h
B        76.1.0.0 [20/0] via 192.168.78.2, 1d14h
B        76.1.1.0 [20/0] via 192.168.78.2, 1d14h
B        76.1.2.0 [20/0] via 192.168.78.2, 1d14h
B        76.1.3.0 [20/0] via 192.168.78.2, 1d14h
R78#
```

Now we apply acl and filter in R74 and verify the same in R78



```
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 aggregate-address 74.1.0.0 255.255.252.0
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER transport connection-mode active
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER route-reflector-client
 neighbor BGP_PEER next-hop-self
 neighbor BGP_PEER prefix-list SROUTE_IBGP out
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.78.1 distribute-list 11 out
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
 neighbor 192.168.87.1 prefix-list SUM_BB out
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
ip prefix-list SROUTE_IBGP seq 5 deny 74.1.0.0/22
ip prefix-list SROUTE_IBGP seq 10 permit 0.0.0.0/0 le 32

ip prefix-list SUM_BB seq 5 deny 74.1.0.0/22 ge 24
ip prefix-list SUM_BB seq 10 permit 0.0.0.0/0 le 32
!
route-map 74_NET permit 10
 match interface Loopback1 Loopback2 Loopback3 Loopback4 Loopback0 Ethernet0/0 Ethernet0/1
Ethernet0/2
 set origin igp
!
route-map PREPEND permit 10
 set as-path prepend 100 100 100
!
!
access-list 11 deny   76.1.0.0 0.0.0.255
access-list 11 deny   76.1.1.0 0.0.0.255
access-list 11 deny   76.1.2.0 0.0.0.255
access-list 11 deny   76.1.3.0 0.0.0.255
access-list 11 permit any
!
```

R78 routing table: (we have no route on R78)



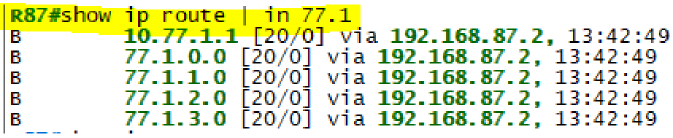
```
R78#show ip route 76.1.0.1
% Network not in table
R78#show ip route 76.1.1.1
% Network not in table
R78#show ip route 76.1.2.1
% Network not in table
R78#show ip route 76.1.3.1
% Network not in table
R78#show ip route 76.1.4.1
% Network not in table
R78#show ip route | in 76.1
B        10.76.1.1 [20/0] via 192.168.78.2, 1d14h
```

You can also achieve on R78 by mentioning **in**(since we learning routes from R74)

BGP Filtering with Prefix-Lists

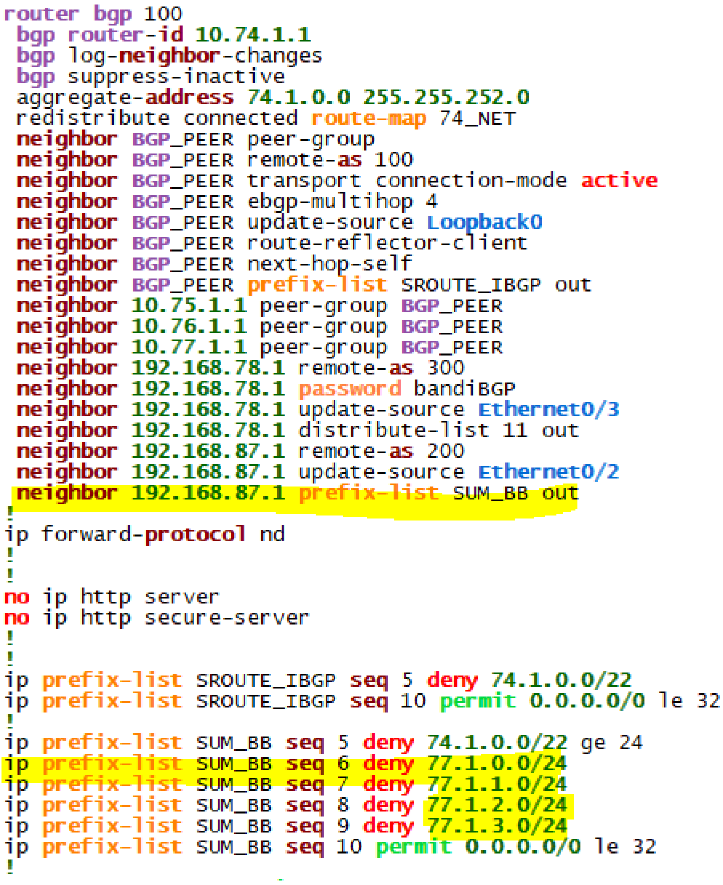
We use here R74 to filter router learn from R77 towards R878 that is AS 200

Before we apply pre-fix list



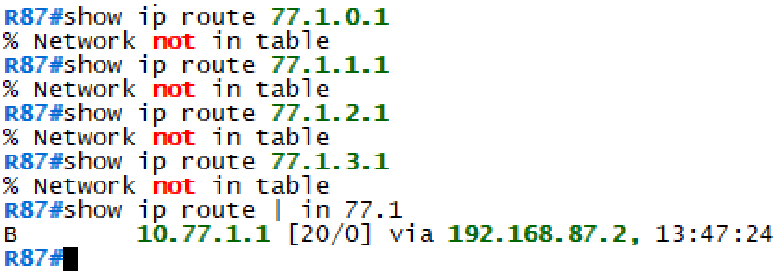
```
R87#show ip route | in 77.1
B        10.77.1.1 [20/0] via 192.168.87.2, 13:42:49
B        77.1.0.0 [20/0] via 192.168.87.2, 13:42:49
B        77.1.1.0 [20/0] via 192.168.87.2, 13:42:49
B        77.1.2.0 [20/0] via 192.168.87.2, 13:42:49
B        77.1.3.0 [20/0] via 192.168.87.2, 13:42:49
```

R74 I have used existing prefix list since it was already configured for some other excise.



```
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 bgp suppress-inactive
 aggregate-address 74.1.0.0 255.255.252.0
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER transport connection-mode active
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER route-reflector-client
 neighbor BGP_PEER next-hop-self
 neighbor BGP_PEER prefix-list SROUTE_IBGP out
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.78.1 distribute-list 11 out
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
 neighbor 192.168.87.1 prefix-list SUM_BB out
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
ip prefix-list SROUTE_IBGP seq 5 deny 74.1.0.0/22
ip prefix-list SROUTE_IBGP seq 10 permit 0.0.0.0/0 le 32
!
ip prefix-list SUM_BB seq 5 deny 74.1.0.0/22 ge 24
ip prefix-list SUM_BB seq 6 deny 77.1.0.0/24
ip prefix-list SUM_BB seq 7 deny 77.1.1.0/24
ip prefix-list SUM_BB seq 8 deny 77.1.2.0/24
ip prefix-list SUM_BB seq 9 deny 77.1.3.0/24
ip prefix-list SUM_BB seq 10 permit 0.0.0.0/0 le 32
!
```

After Prefix list applied.



```
R87#show ip route 77.1.0.1
% Network not in table
R87#show ip route 77.1.1.1
% Network not in table
R87#show ip route 77.1.2.1
% Network not in table
R87#show ip route 77.1.3.1
% Network not in table
R87#show ip route | in 77.1
B        10.77.1.1 [20/0] via 192.168.87.2, 13:47:24
R87#
```

You can also achieve on R87 by mentioning **in**(since we learning routes from R74)

BGP Filtering with Extended Access-Lists

Extended access-lists add more functionality to BGP prefixes filtering. In addition to matching the subnet numbers, they also allow for subnet mask matching. A typical extended access-list entry in the format permit {proto} <src-subnet> <src-mask> <dstsubnet> <dst-mask> [options] is treated as follows. First, the protocol field and other options are ignored. Next, the <src-subnet> <src-mask> pair is used to build an expression for prefix subnet matching. The pair <dst-subnet> <dst-mask> is used as an expression to match prefixes subnet mask.

2.7.f Implement and troubleshoot multi-protocol BGP
2.7.f [i] IPv4, IPv6, VPN address-family

2.7.g Implement and troubleshoot AS path manipulations
2.7.g [i] Local AS, allow AS in, remove private AS

BGP Communities:

The Community attribute is a numerical value that cab ne attached to a given prefix and advertised to a specific neighbour, once the neighbour received the prefix, it will examine the community value and it will perform either filtering or use that value for route selection process.

By default, no community attribute is sent to any neighbour, To specify that a community attribute should be send to a BGP neighbour, the neighbor send-community command is configured.

Here are the 4 well known BGP communities:

- Internet: advertise the prefix to all BGP neighbors.
- No-Advertise: don’t advertise the prefix to any BGP neighbors.
- No-Export: don’t advertise the prefix to any eBGP neighbors.
- Local-AS: don’t advertise the prefix outside of the sub-AS (this one is used for BGP confederations).

BGP Communities - No-Advertise

The well-known NO_ADVERTISE BGP community signals the BGP speaker not to advertise the particular prefix to any BGP peer. This may be useful to limit the scope of routing information to just directly connected neighbors.

BGP Communities - No-Export

The well-known NO_EXPORT community instructs the BGP speaker to advertise the prefix only across iBGP peering links. This restricts the prefix to remain within the boundaries of the local AS. One good use of this feature is prefix aggregation.
Imagine that your local AS has multiple connections to some other AS. You advertise a summary of all your internal prefixes using the aggregate-address command out of all links. However, you want just the adjacent AS to select the best entry point based on the MED attribute. This could be accomplished by advertising the specific prefixes tagged with NO_EXPORT community along with the aggregates. The neighboring AS would be able to select the best path based on the specific information (MED) but will not advertise the specifics any further, thus containing the routing information.

BGP Communities - Local-AS

The local-AS feature allows a router to appear to be a member of a second autonomous system (AS), in addition to its real AS. This feature can only be used for true eBGP peers. You cannot use this feature for two peers that are members of different confederation sub-ASs.

Use case:

The local-AS feature is useful if ISP-A purchases ISP-B, but ISP-B's customers do not want to modify any peering arrangements or configurations. The local-AS feature allows routers in ISP-B to become members of ISP-A's AS. At the same time, these routers appear to their customers to retain their ISP-B AS number.

2.7.g [ii] Prepend

AS-Path prepending is a way to manipulate the AS-Path attribute of a BGP route. It allows prepending multiple entries of AS to a BGP route. This can come as a workaround if a specific path is required to be followed, and other means like Multi-Exit Discriminator (MED) is not supported.

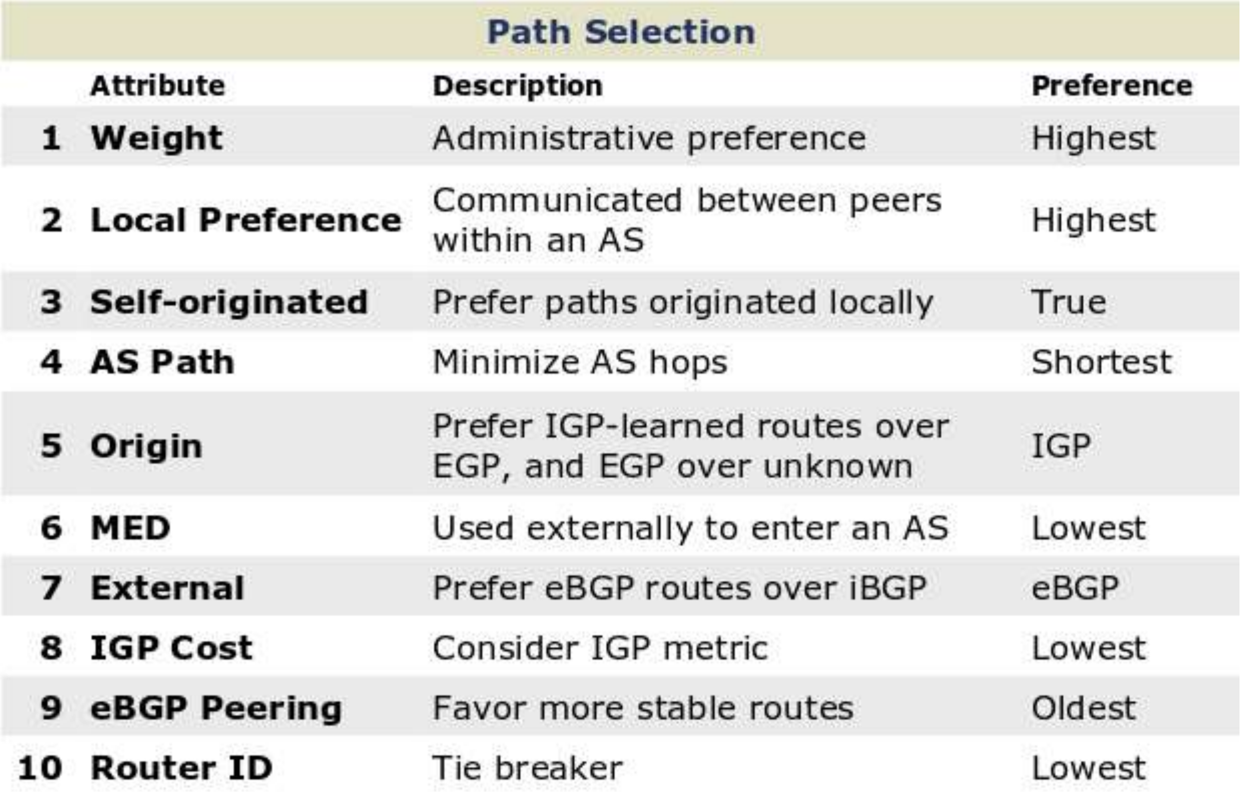
AS-Path prepending can be applied to inbound and outbound direction using route-maps.

2.7.g [iii] Regexp

BGP Attributes and Path Selection

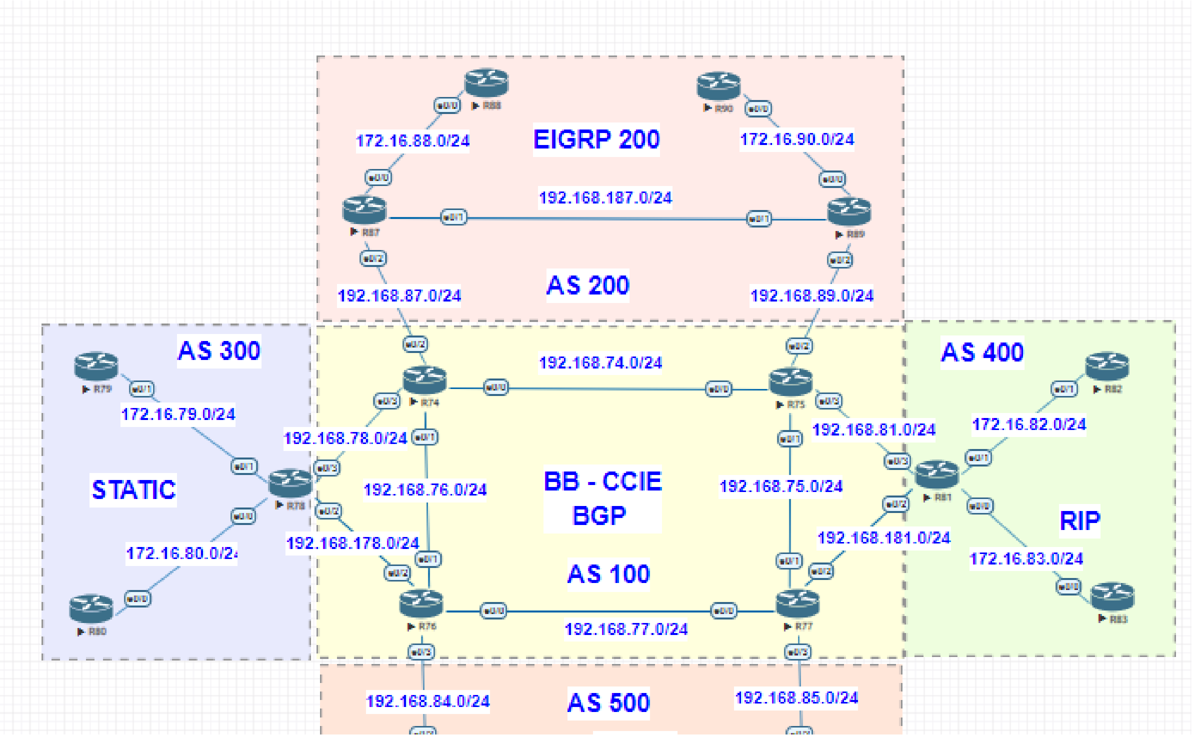
Priory	Attribute	Description
1	**Weight**	1. Prefer the path with the **highest weight**. 2. This is a value that is **local** to the router and it’s Cisco proprietary. 3. The default value is 32768 connected routers – Default 0 for all routes that are not originated by the local router. 4. Control outbound traffic. 5. Attribute that is NOT sent to anybody.
2	**Local Preference**	1. The local preference is used within an autonomous system and exchanged between iBGP routers. 2. We prefer the path with the **highest local preference**. 3. The default value is 100. 4. Controls only outbound traffic 5. It must be numeric value.
3	**Originate**	Prefer the path that the local router originated. In the BGP table, you will see **next hop 0.0.0.0**. You can get a path in the BGP table through the BGP network command, aggregation, or redistribution. A BGP router will prefer routes that it installed into BGP itself over a route that another router installed in BGP.
4	**AS Path**	1. Used for loop prevention 2. Prefer the path with the **shortest AS path length** 3. Control inbound or outbound traffic
5	**Origin Code**	Prefer the lowest origin code. There are three origin codes: • IGP

		• EGP • INCOMPLETE IGP is lower than EGP and EGP is lower than INCOMPLETE.
6	**MED**	1. Prefer the path with the **lowest MED**. 2. The MED is exchanged between autonomous systems (this means eBGP neighbours) 3. Control inbound traffic 4. Default value is 0
7	eBGP over iBGP Path	Prefer eBGP (external BGP) over iBGP (internal BGP) paths. (because of metric 20)
8	Shortest IGP path to BGP next hop	Prefer the path within the autonomous system with the **lowest IGP metric** to the BGP next hop.
9	Oldest Path	Prefer the path that **we received first,** in other words, the oldest path.
10	Router-ID	Prefer the path with the **lowest BGP neighbor router ID**. The router ID is based on the highest IP address. If you have a loopback interface, then the IP address on the loopback will be used. The router ID can also be manually configured.
11	Neighbor IP address	Prefer the path with the **lowest neighbor IP address**. If you have two eBGP routers and two links in between then the router ID will be the same. In this case, the neighbor IP address is the tiebreaker.



Path Selection

	Attribute	Description	Preference
1	**Weight**	Administrative preference	Highest
2	**Local Preference**	Communicated between peers within an AS	Highest
3	**Self-originated**	Prefer paths originated locally	True
4	**AS Path**	Minimize AS hops	Shortest
5	**Origin**	Prefer IGP-learned routes over EGP, and EGP over unknown	IGP
6	**MED**	Used externally to enter an AS	Lowest
7	**External**	Prefer eBGP routes over iBGP	eBGP
8	**IGP Cost**	Consider IGP metric	Lowest
9	**eBGP Peering**	Favor more stable routes	Oldest
10	**Router ID**	Tie breaker	Lowest

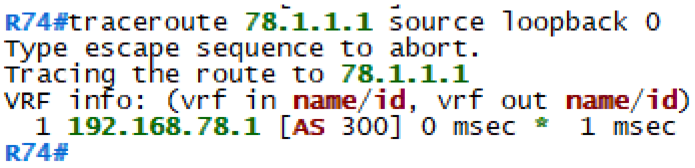


We use above topology to configure attributes.

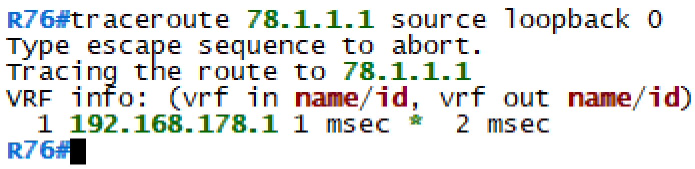
Local Preference.

We use R74, R75 and R78, for now all the routers have BGP peering relation.
R74 and R75 have tie breaker with all the attribute so they go by the default route.

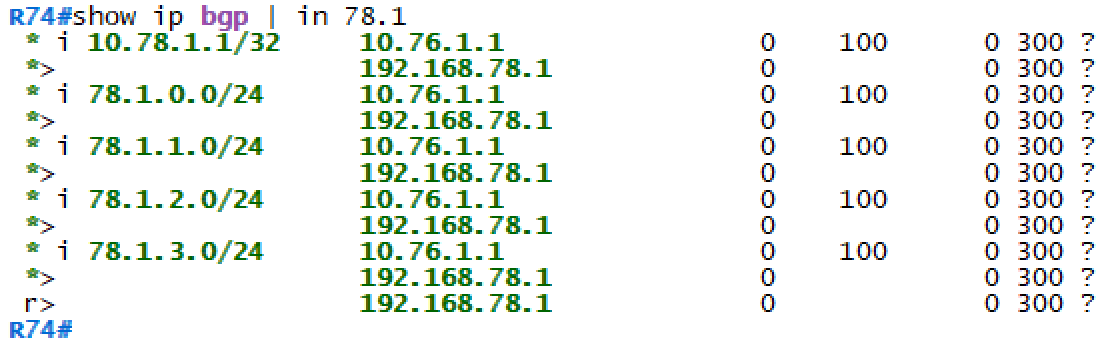
Verify before we implement local preference



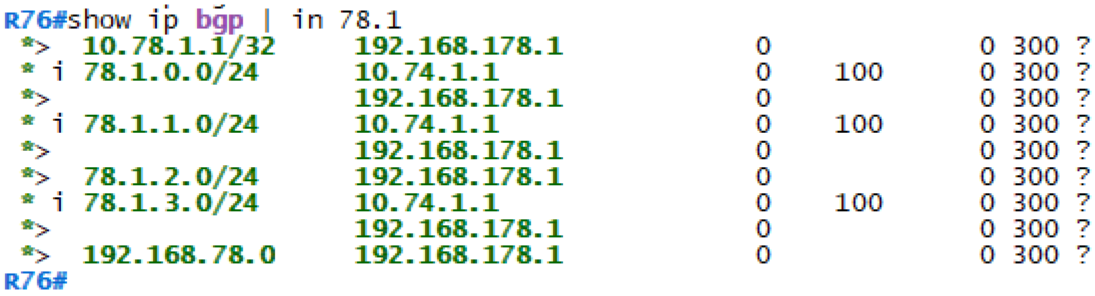
```
R74#traceroute 78.1.1.1 source loopback 0
Type escape sequence to abort.
Tracing the route to 78.1.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.78.1 [AS 300] 0 msec *  1 msec
R74#
```



```
R76#traceroute 78.1.1.1 source loopback 0
Type escape sequence to abort.
Tracing the route to 78.1.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.178.1 1 msec *  2 msec
R76#
```

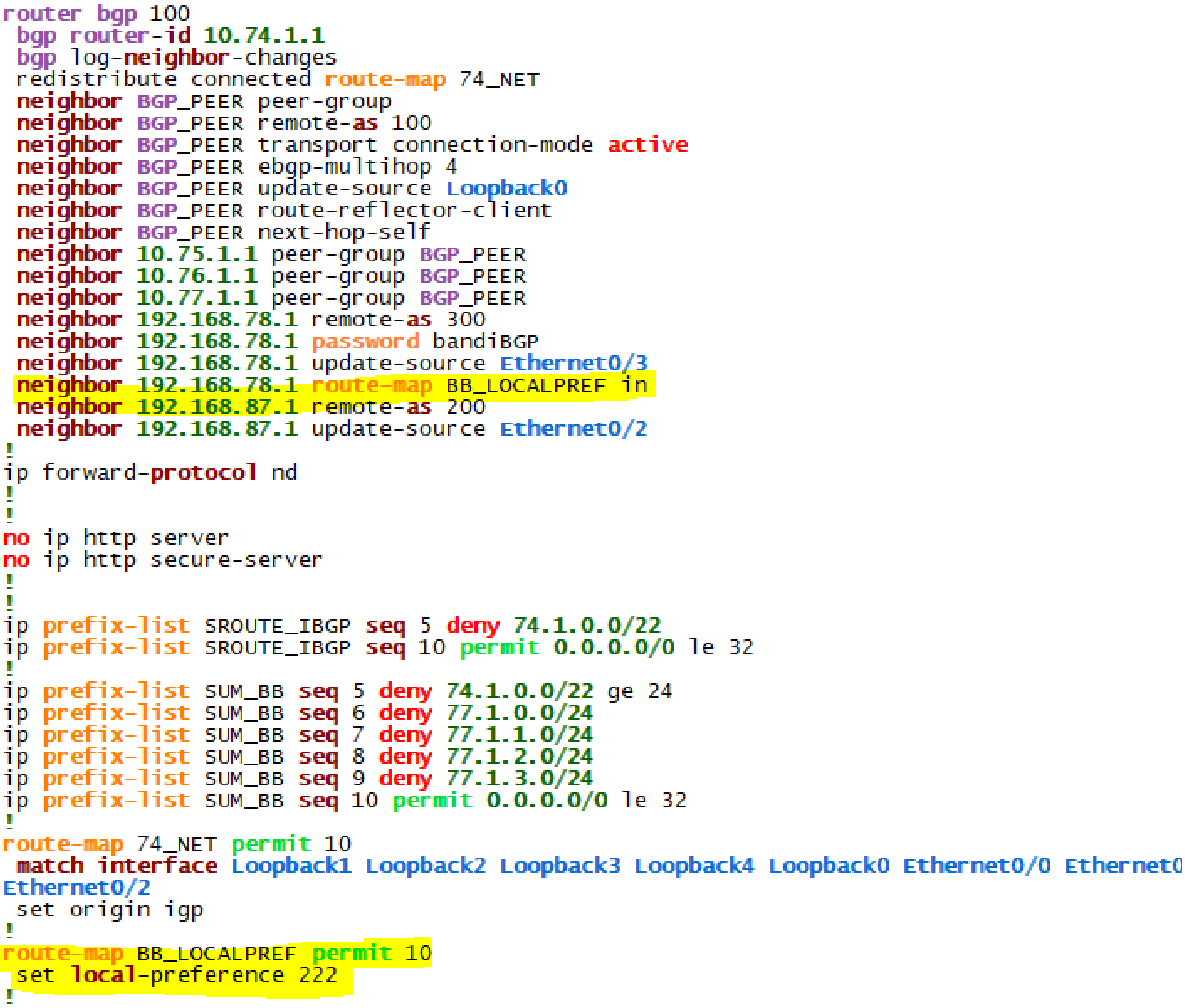


```
R74#show ip bgp | in 78.1
 * i 10.78.1.1/32     10.76.1.1                0    100      0 300 ?
 *>                   192.168.78.1             0             0 300 ?
 * i 78.1.0.0/24      10.76.1.1                0    100      0 300 ?
 *>                   192.168.78.1             0             0 300 ?
 * i 78.1.1.0/24      10.76.1.1                0    100      0 300 ?
 *>                   192.168.78.1             0             0 300 ?
 * i 78.1.2.0/24      10.76.1.1                0    100      0 300 ?
 *>                   192.168.78.1             0             0 300 ?
 * i 78.1.3.0/24      10.76.1.1                0    100      0 300 ?
 *>                   192.168.78.1             0             0 300 ?
 r>                   192.168.78.1             0             0 300 ?
R74#
```

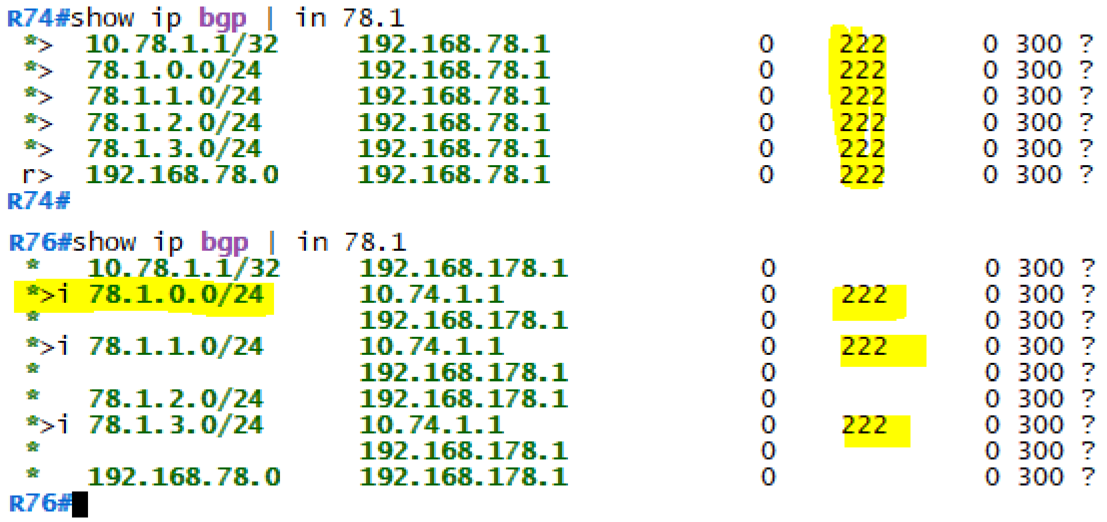


```
R76#show ip bgp | in 78.1
 *>  10.78.1.1/32     192.168.178.1            0             0 300 ?
 * i 78.1.0.0/24      10.74.1.1                0    100      0 300 ?
 *>                   192.168.178.1            0             0 300 ?
 * i 78.1.1.0/24      10.74.1.1                0    100      0 300 ?
 *>                   192.168.178.1            0             0 300 ?
 *>  78.1.2.0/24      192.168.178.1            0             0 300 ?
 * i 78.1.3.0/24      10.74.1.1                0    100      0 300 ?
 *>                   192.168.178.1            0             0 300 ?
 *>  192.168.78.0     192.168.178.1            0             0 300 ?
R76#
```

Now we implement Local preference on R74 so all the traffic use R74 path to reach 78.1.1.0/24 network.

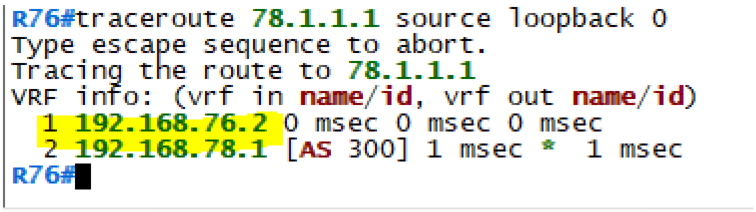


```
router bgp 100
 bgp router-id 10.74.1.1
 bgp log-neighbor-changes
 redistribute connected route-map 74_NET
 neighbor BGP_PEER peer-group
 neighbor BGP_PEER remote-as 100
 neighbor BGP_PEER transport connection-mode active
 neighbor BGP_PEER ebgp-multihop 4
 neighbor BGP_PEER update-source Loopback0
 neighbor BGP_PEER route-reflector-client
 neighbor BGP_PEER next-hop-self
 neighbor 10.75.1.1 peer-group BGP_PEER
 neighbor 10.76.1.1 peer-group BGP_PEER
 neighbor 10.77.1.1 peer-group BGP_PEER
 neighbor 192.168.78.1 remote-as 300
 neighbor 192.168.78.1 password bandiBGP
 neighbor 192.168.78.1 update-source Ethernet0/3
 neighbor 192.168.78.1 route-map BB_LOCALPREF in
 neighbor 192.168.87.1 remote-as 200
 neighbor 192.168.87.1 update-source Ethernet0/2
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
ip prefix-list SROUTE_IBGP seq 5 deny 74.1.0.0/22
ip prefix-list SROUTE_IBGP seq 10 permit 0.0.0.0/0 le 32
!
ip prefix-list SUM_BB seq 5 deny 74.1.0.0/22 ge 24
ip prefix-list SUM_BB seq 6 deny 77.1.0.0/24
ip prefix-list SUM_BB seq 7 deny 77.1.1.0/24
ip prefix-list SUM_BB seq 8 deny 77.1.2.0/24
ip prefix-list SUM_BB seq 9 deny 77.1.3.0/24
ip prefix-list SUM_BB seq 10 permit 0.0.0.0/0 le 32
!
route-map 74_NET permit 10
 match interface Loopback1 Loopback2 Loopback3 Loopback4 Loopback0 Ethernet0/0 Ethernet0
Ethernet0/2
 set origin igp
!
route-map BB_LOCALPREF permit 10
 set local-preference 222
!
```



```
R74#show ip bgp | in 78.1
 *>  10.78.1.1/32     192.168.78.1             0    222      0 300 ?
 *>  78.1.0.0/24      192.168.78.1             0    222      0 300 ?
 *>  78.1.1.0/24      192.168.78.1             0    222      0 300 ?
 *>  78.1.2.0/24      192.168.78.1             0    222      0 300 ?
 *>  78.1.3.0/24      192.168.78.1             0    222      0 300 ?
 r>  192.168.78.0     192.168.78.1             0    222      0 300 ?
R74#

R76#show ip bgp | in 78.1
 *   10.78.1.1/32     192.168.178.1            0             0 300 ?
 *>i 78.1.0.0/24      10.74.1.1                0    222      0 300 ?
 *                    192.168.178.1            0             0 300 ?
 *>i 78.1.1.0/24      10.74.1.1                0    222      0 300 ?
 *                    192.168.178.1            0             0 300 ?
 *   78.1.2.0/24      192.168.178.1            0             0 300 ?
 *>i 78.1.3.0/24      10.74.1.1                0    222      0 300 ?
 *                    192.168.178.1            0             0 300 ?
 *   192.168.78.0     192.168.178.1            0             0 300 ?
R76#
```

Verification: on R76, now it uses R74 as route path because we increased the Local Preference to 222



```
R76#traceroute 78.1.1.1 source loopback 0
Type escape sequence to abort.
Tracing the route to 78.1.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.76.2 0 msec 0 msec 0 msec
  2 192.168.78.1 [AS 300] 1 msec *  1 msec
R76#
```

2.7.h Implement and Troubleshoot Other Features

2.7.h [i] Multipath

2.7.h [ii] BGP synchronization

2.7.h [iii] Soft reconfiguration, route refresh

2.8 Troubleshooting layer 3 technologies

2.8.a Use IOS troubleshooting tools

2.8.a [i] debug, conditional debug

2.8.a [ii] ping, traceroute with extended options

2.8.a [iii] Embedded packet capture

2.8.b Apply troubleshooting methodologies

2.8.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]

2.8.b [ii] Design and implement valid solutions according to constraints

2.8.b [iii] Verify and monitor resolution

2.8.c Interpret packet capture

2.8.c [i] Using wireshark trace analyzer

2.8.c [ii] Using IOS embedded packet capture

3.0 VPN Technologies 20 %

3.1 Tunneling

3.1.a Implement and troubleshoot MPLS operations
3.1.a [i] Label stack, LSR, LSP
3.1.a [ii] LDP
3.1.a [iii] MPLS ping, MPLS traceroute

3.1.b Implement and troubleshoot basic MPLS L3VPN
3.1.b [i] L3VPN, CE, PE, P
3.1.b [ii] Extranet [route leaking]

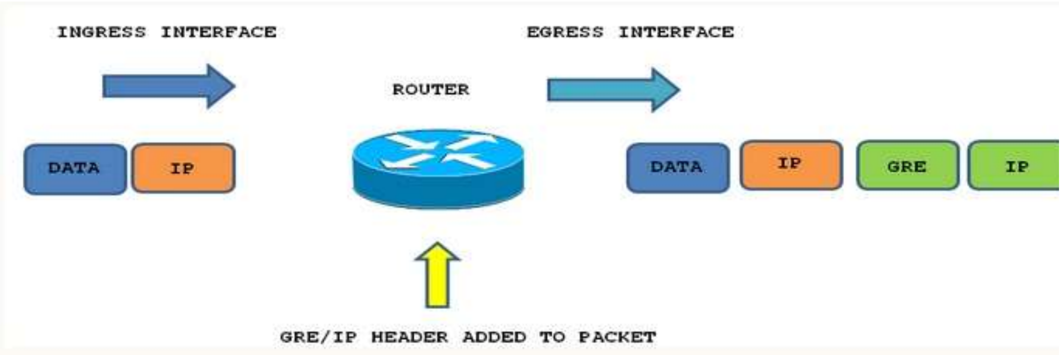
3.1.c Implement and troubleshoot encapsulation
3.1.c [i] GRE
3.1.c [ii] Dynamic GRE

https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/layer-3-vpns-l3vpn/116725-configure-mgre-00.html

GRE Tunnel

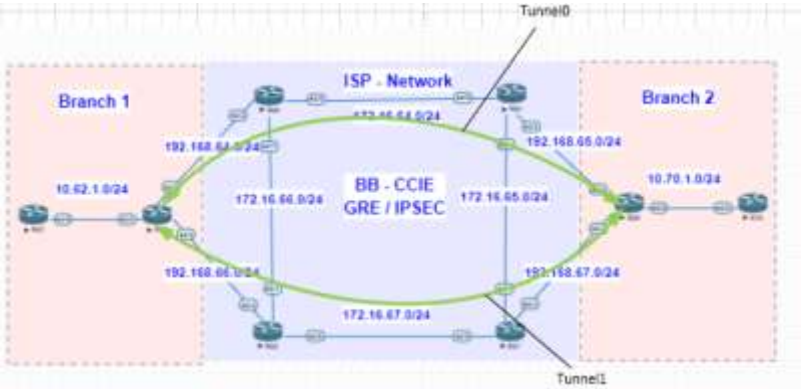
Tunneling provides a mechanism to transport packets of one protocol within another protocol. The protocol that is carried is called as the passenger protocol, and the protocol that is used for carrying the passenger protocol is called as the transport protocol.

Generic Routing Encapsulation (GRE) is one of the available tunneling mechanisms which uses IP as the transport protocol and can be used for carrying many different passenger protocols. The tunnels behave as virtual point-to-point links that have two endpoints identified by the tunnel source and tunnel destination addresses at each endpoint.



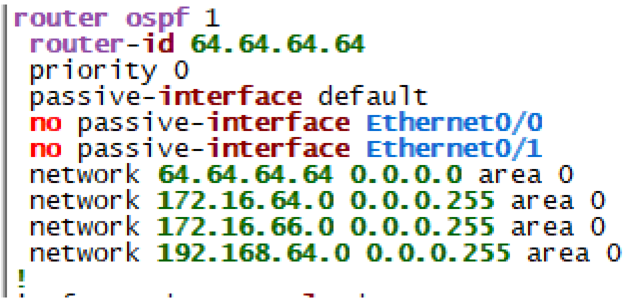
GRE is an encapsulation protocol designed to initiate point-to-point connections able to carry any OSI layer 3 protocol **over an IP network**. Since it is a tunneling protocol, it can be deployed to connect discontinuous sub-networks, or provide workarounds for networks with limited hops. GRE packets are identified within an IP packet by the protocol type 47.

GRE itself do not offer any Security Layer, need to add IPSEC or PPTP on top of GRE to secure the data.

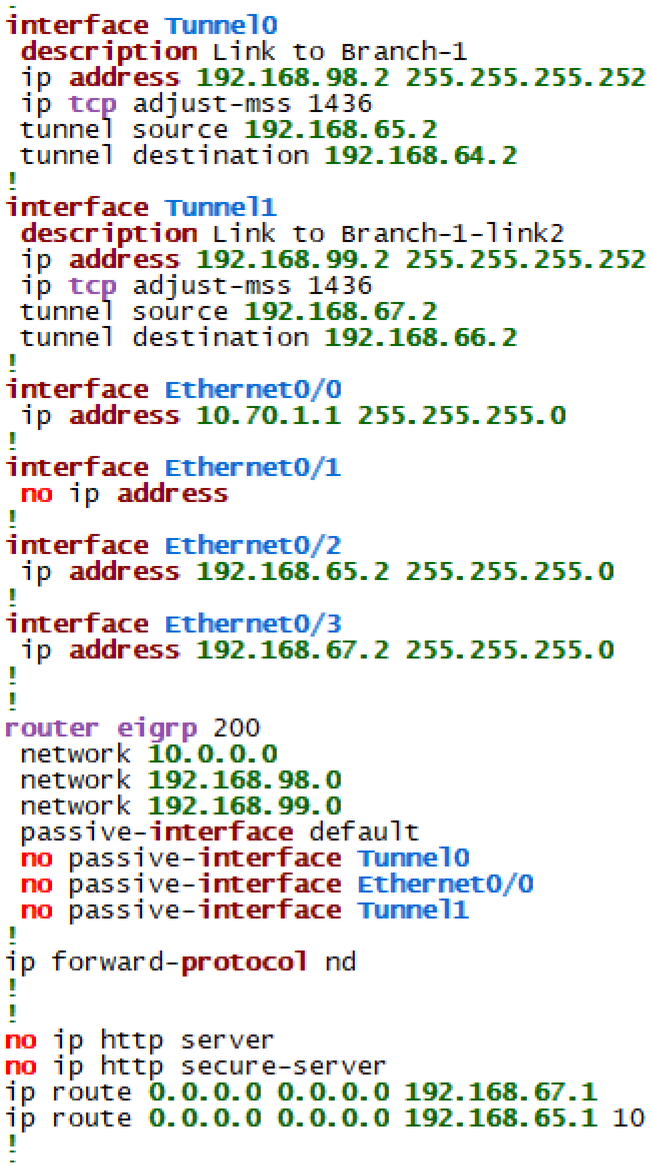


We run ISP side OSPF config – we run EIGRP over tunnel Brach office 1 and 2

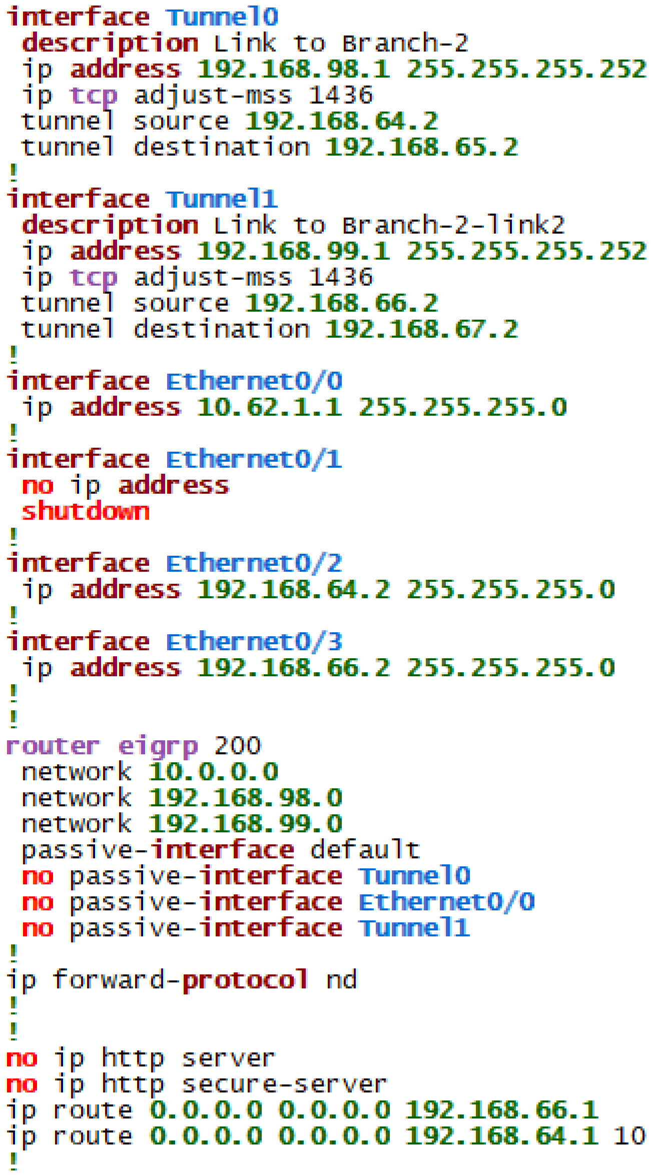
OSPF config to form network R64, R65, R66, R67 as below


```
router ospf 1
 router-id 64.64.64.64
 priority 0
 passive-interface default
 no passive-interface Ethernet0/0
 no passive-interface Ethernet0/1
 network 64.64.64.64 0.0.0.0 area 0
 network 172.16.64.0 0.0.0.255 area 0
 network 172.16.66.0 0.0.0.255 area 0
 network 192.168.64.0 0.0.0.255 area 0
!
```

R68, R69 GRE/Eigrip config as below :

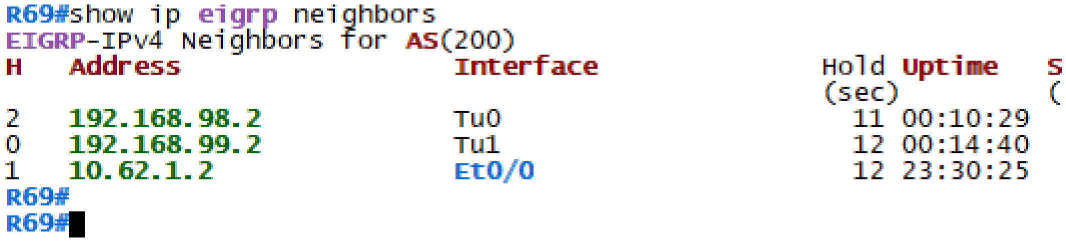


```
!
interface Tunnel0
 description Link to Branch-1
 ip address 192.168.98.2 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.65.2
 tunnel destination 192.168.64.2
!
interface Tunnel1
 description Link to Branch-1-link2
 ip address 192.168.99.2 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.67.2
 tunnel destination 192.168.66.2
!
interface Ethernet0/0
 ip address 10.70.1.1 255.255.255.0
!
interface Ethernet0/1
 no ip address
!
interface Ethernet0/2
 ip address 192.168.65.2 255.255.255.0
!
interface Ethernet0/3
 ip address 192.168.67.2 255.255.255.0
!
!
router eigrp 200
 network 10.0.0.0
 network 192.168.98.0
 network 192.168.99.0
 passive-interface default
 no passive-interface Tunnel0
 no passive-interface Ethernet0/0
 no passive-interface Tunnel1
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
ip route 0.0.0.0 0.0.0.0 192.168.67.1
ip route 0.0.0.0 0.0.0.0 192.168.65.1 10
!
```

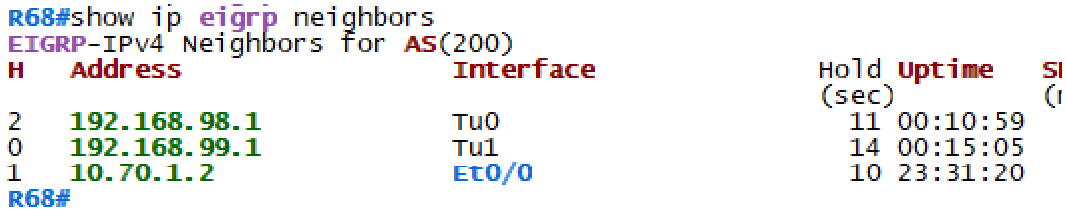


```
interface Tunnel0
 description Link to Branch-2
 ip address 192.168.98.1 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.64.2
 tunnel destination 192.168.65.2
!
interface Tunnel1
 description Link to Branch-2-link2
 ip address 192.168.99.1 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.66.2
 tunnel destination 192.168.67.2
!
interface Ethernet0/0
 ip address 10.62.1.1 255.255.255.0
!
interface Ethernet0/1
 no ip address
 shutdown
!
interface Ethernet0/2
 ip address 192.168.64.2 255.255.255.0
!
interface Ethernet0/3
 ip address 192.168.66.2 255.255.255.0
!
!
router eigrp 200
 network 10.0.0.0
 network 192.168.98.0
 network 192.168.99.0
 passive-interface default
 no passive-interface Tunnel0
 no passive-interface Ethernet0/0
 no passive-interface Tunnel1
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
ip route 0.0.0.0 0.0.0.0 192.168.66.1
ip route 0.0.0.0 0.0.0.0 192.168.64.1 10
!
```

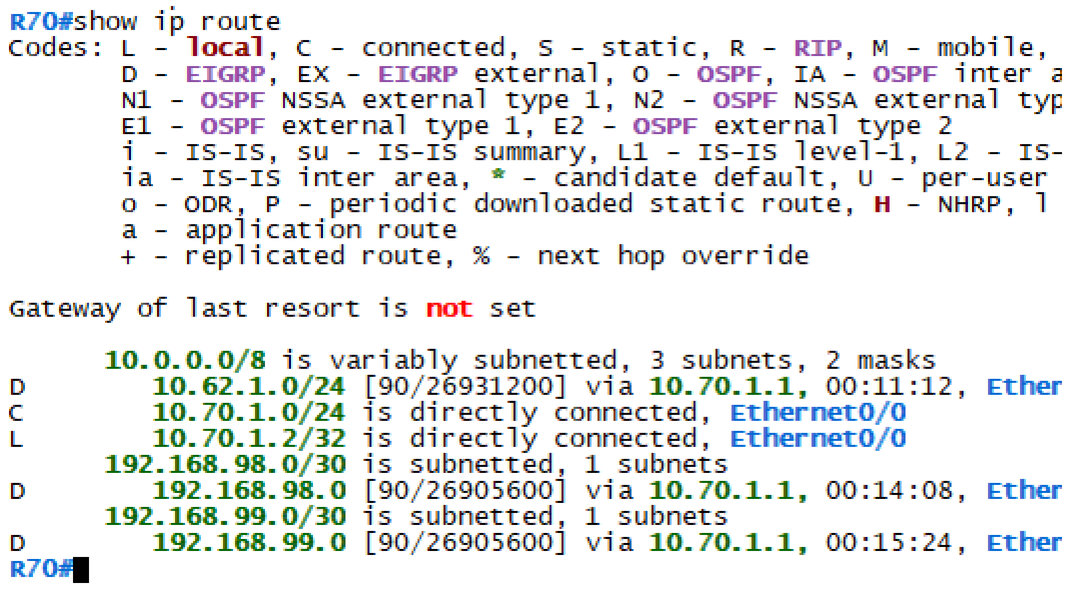
Check Tunnel and EIGRP verification and reachability between R62 and R70 networks.



```
R69#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   S
                                                   (sec)         (
2   192.168.98.2            Tu0                      11 00:10:29
0   192.168.99.2            Tu1                      12 00:14:40
1   10.62.1.2               Et0/0                    12 23:30:25
R69#
R69#
```

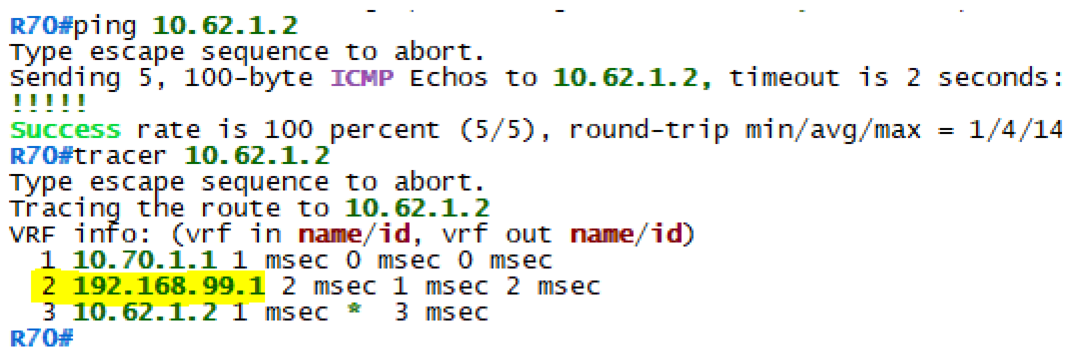


```
R68#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SI
                                                   (sec)         (
2   192.168.98.1            Tu0                      11 00:10:59
0   192.168.99.1            Tu1                      14 00:15:05
1   10.70.1.2               Et0/0                    10 23:31:20
R68#
```



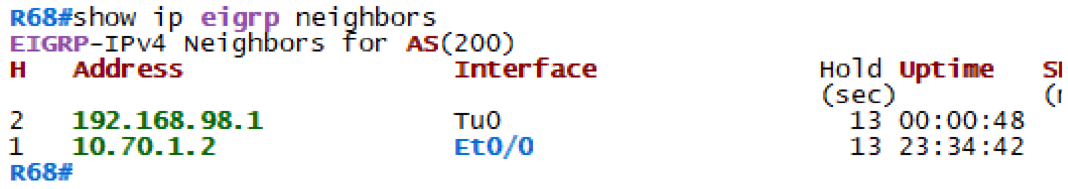
```
R70#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile,
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter a
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external typ
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-
       ia - IS-IS inter area, * - candidate default, U - per-user
       o - ODR, P - periodic downloaded static route, H - NHRP, l
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
D        10.62.1.0/24 [90/26931200] via 10.70.1.1, 00:11:12, Ether
C        10.70.1.0/24 is directly connected, Ethernet0/0
L        10.70.1.2/32 is directly connected, Ethernet0/0
      192.168.98.0/30 is subnetted, 1 subnets
D        192.168.98.0 [90/26905600] via 10.70.1.1, 00:14:08, Ether
      192.168.99.0/30 is subnetted, 1 subnets
D        192.168.99.0 [90/26905600] via 10.70.1.1, 00:15:24, Ether
R70#
```

Verification we can see the traffic going via Tunnel1

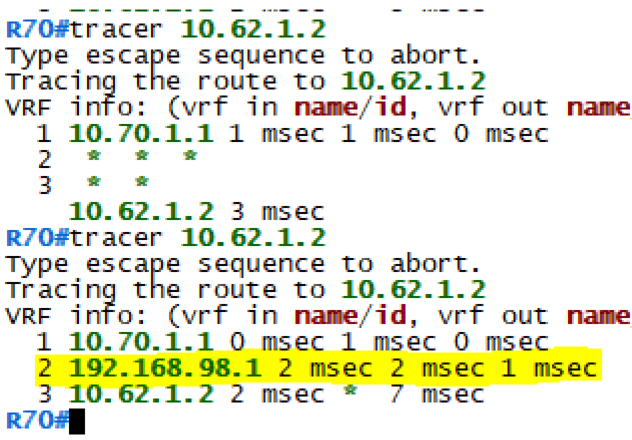


```
R70#ping 10.62.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.62.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/14
R70#tracer 10.62.1.2
Type escape sequence to abort.
Tracing the route to 10.62.1.2
VRF info: (vrf in name/id, vrf out name/id)
  1 10.70.1.1 1 msec 0 msec 0 msec
  2 192.168.99.1 2 msec 1 msec 2 msec
  3 10.62.1.2 1 msec *  3 msec
R70#
```

Lets bring down the Tunnel 1 and see if the traffic reach over Tunnel 0



```
R68#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SR
                                                   (sec)         (r
2   192.168.98.1            Tu0                      13 00:00:48
1   10.70.1.2               Et0/0                    13 23:34:42
R68#
```

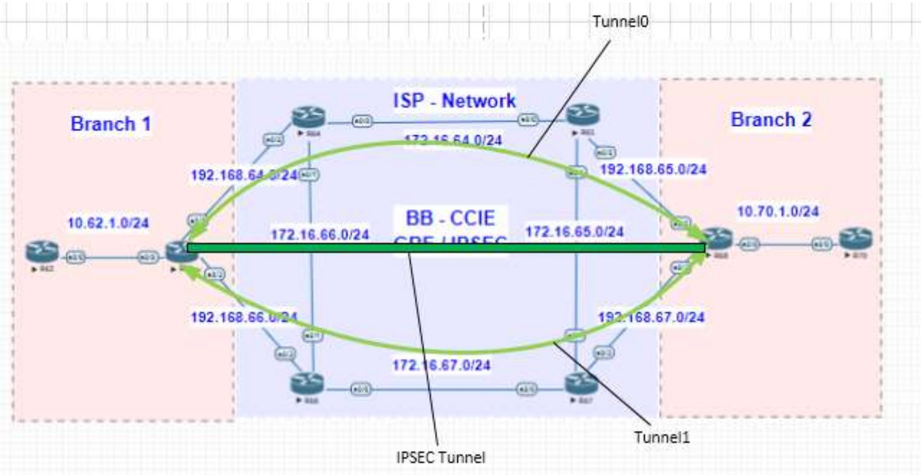


```
R70#tracer 10.62.1.2
Type escape sequence to abort.
Tracing the route to 10.62.1.2
VRF info: (vrf in name/id, vrf out name/id)
  1 10.70.1.1 1 msec 1 msec 0 msec
  2  *  *  *
  3  *  *
    10.62.1.2 3 msec
R70#tracer 10.62.1.2
Type escape sequence to abort.
Tracing the route to 10.62.1.2
VRF info: (vrf in name/id, vrf out name/id)
  1 10.70.1.1 0 msec 1 msec 0 msec
  2 192.168.98.1 2 msec 2 msec 1 msec
  3 10.62.1.2 2 msec *  7 msec
R70#
```

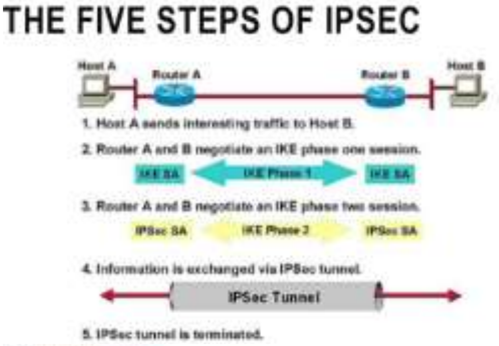
GRE / IPSEC

GRE/IPSEC has several benefits of using GRE tunnels alongside IPSec, primarily because IPSec does not support traffic other than **unicast**. This can lead to issues if you plan to use services that require such type of traffic, such as routing protocols like OSPF or EIGRP.
GRE encapsulation process, broadcast and multicast traffic are encapsulated into a unicast packet that can be treated by IPSec, making dynamic routing possible between peers separated by an insecure network area.

Implementing IPsec benefit from GRE strengths and are concerned with privacy (remember that GRE **does not** encrypt traffic). Also, GRE tunnels provide a higher level of resiliency than IKE keepalives actually do.

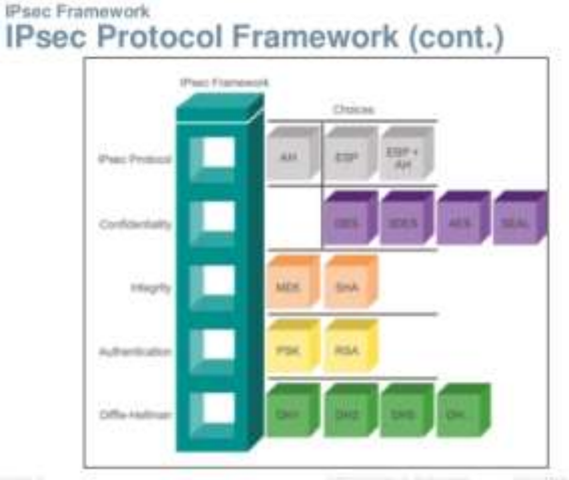


IPSEC Protocol



THE FIVE STEPS OF IPSEC

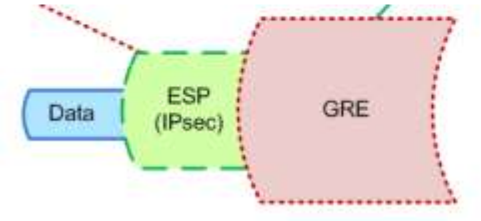
1. Host A sends interesting traffic to Host B.
2. Router A and B negotiate an IKE phase one session.
3. Router A and B negotiate an IKE phase two session.
4. Information is exchanged via IPSec tunnel.
5. IPSec tunnel is terminated.



IPsec Framework

IPsec Protocol Framework (cont.)

2 Phases to get IPSEC working.



Phase 1 (ISAKMP) also called IKE - This will be used to exchange the Key – UDP – 500 / NAT-T 4500

Main components (K E H)

Key (K) - PSK (its pre-shared key)
Encryption(E) - DES / 3DES / AES
Hash(H) – MD5 / SHA

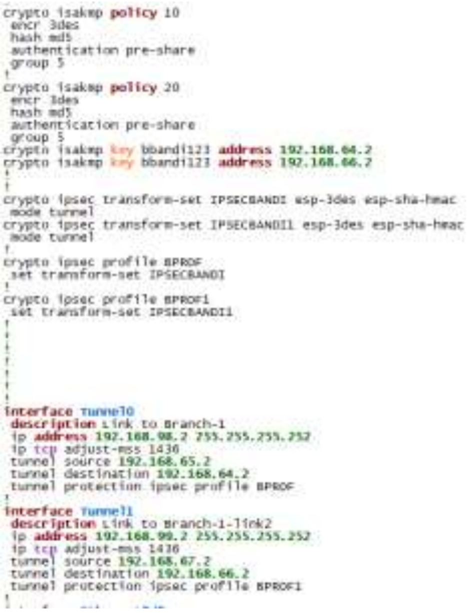
Session Key Generated by DH (Diffie Helmen) Option 1 3 5 ..

Phase 2 – This will be used to Exchange the DATA ESP Protocol 50

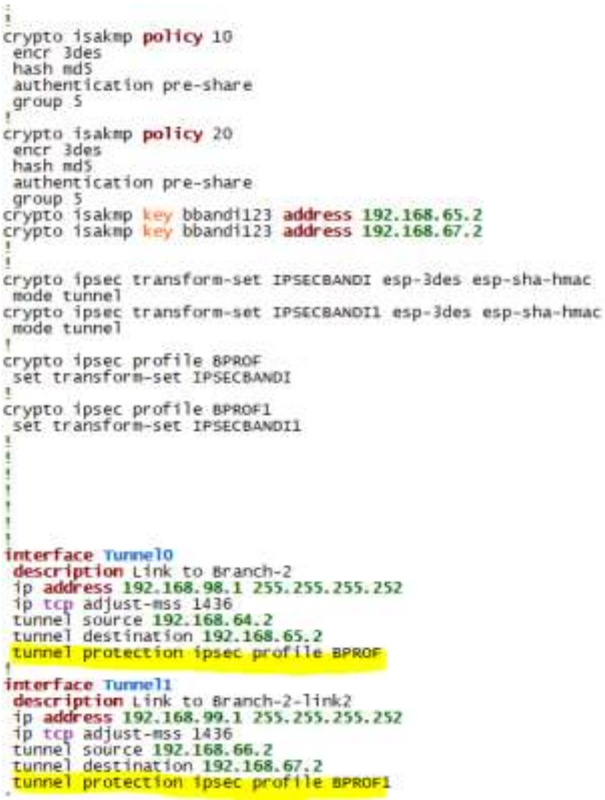
Main component (E H)

Encryption(E) - DES / 3DES / AES
Hash(H) – MD5 / SHA

We configure IPSEC config on R68 and R69

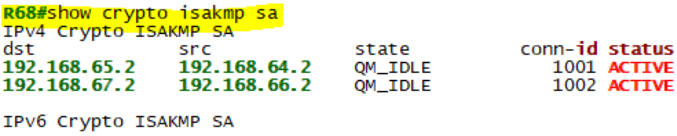


```
crypto isakmp policy 10
 encr 3des
 hash md5
 authentication pre-share
 group 5
!
crypto isakmp policy 20
 encr 3des
 hash md5
 authentication pre-share
 group 5
crypto isakmp key bbandi123 address 192.168.64.2
crypto isakmp key bbandi123 address 192.168.66.2
!
!
crypto ipsec transform-set IPSECBANDI esp-3des esp-sha-hmac
 mode tunnel
crypto ipsec transform-set IPSECBANDI1 esp-3des esp-sha-hmac
 mode tunnel
!
crypto ipsec profile BPROF
 set transform-set IPSECBANDI
!
crypto ipsec profile BPROF1
 set transform-set IPSECBANDI1
!
!
!
!
!
!
!
interface Tunnel0
 description Link to Branch-1
 ip address 192.168.98.2 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.65.2
 tunnel destination 192.168.64.2
 tunnel protection ipsec profile BPROF
!
interface Tunnel1
 description Link to Branch-1-link2
 ip address 192.168.99.2 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.67.2
 tunnel destination 192.168.66.2
 tunnel protection ipsec profile BPROF1
!
```

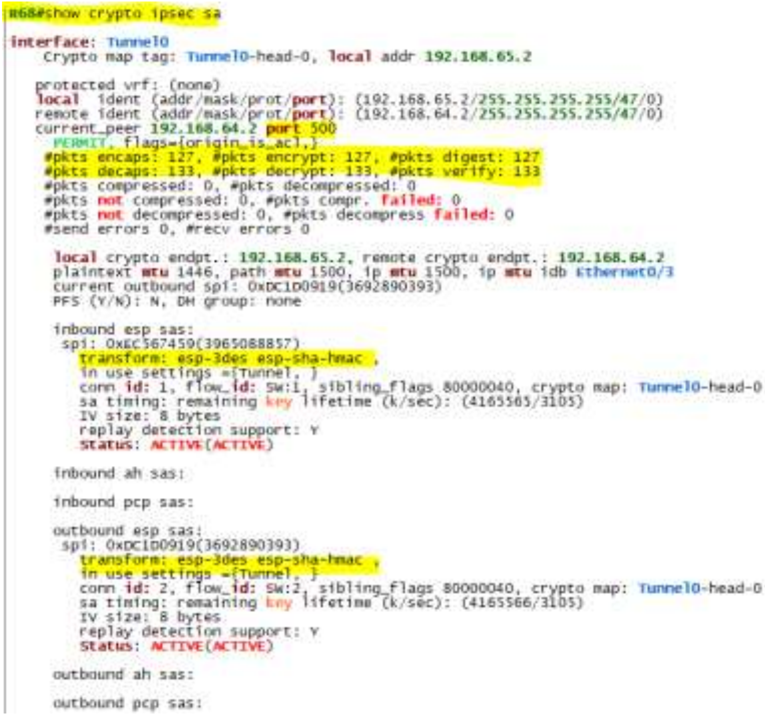


```
!
crypto isakmp policy 10
 encr 3des
 hash md5
 authentication pre-share
 group 5
!
crypto isakmp policy 20
 encr 3des
 hash md5
 authentication pre-share
 group 5
crypto isakmp key bbandi123 address 192.168.65.2
crypto isakmp key bbandi123 address 192.168.67.2
!
!
crypto ipsec transform-set IPSECBANDI esp-3des esp-sha-hmac
 mode tunnel
crypto ipsec transform-set IPSECBANDI1 esp-3des esp-sha-hmac
 mode tunnel
!
crypto ipsec profile BPROF
 set transform-set IPSECBANDI
!
crypto ipsec profile BPROF1
 set transform-set IPSECBANDI1
!
!
!
!
!
!
!
interface Tunnel0
 description Link to Branch-2
 ip address 192.168.98.1 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.64.2
 tunnel destination 192.168.65.2
 tunnel protection ipsec profile BPROF
!
interface Tunnel1
 description Link to Branch-2-link2
 ip address 192.168.99.1 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.66.2
 tunnel destination 192.168.67.2
 tunnel protection ipsec profile BPROF1
```

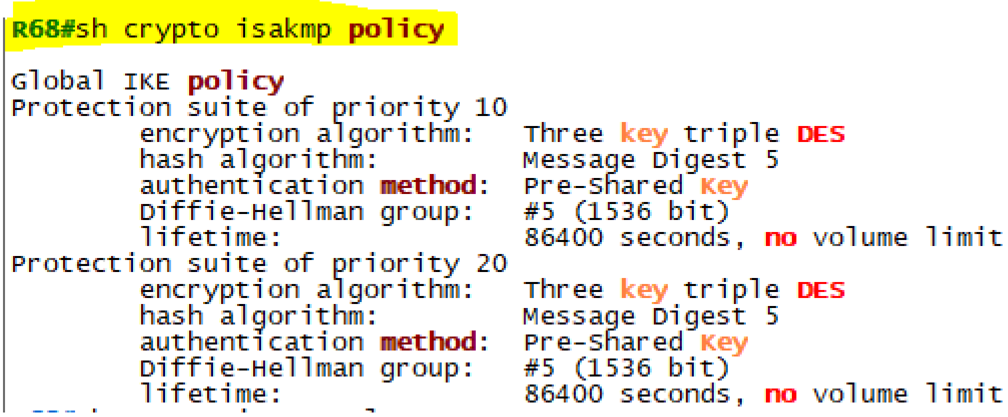
Now we check Crypto



```
R68#show crypto isakmp sa
IPv4 Crypto ISAKMP SA
dst             src             state          conn-id status
192.168.65.2    192.168.64.2    QM_IDLE           1001 ACTIVE
192.168.67.2    192.168.66.2    QM_IDLE           1002 ACTIVE

IPv6 Crypto ISAKMP SA
```



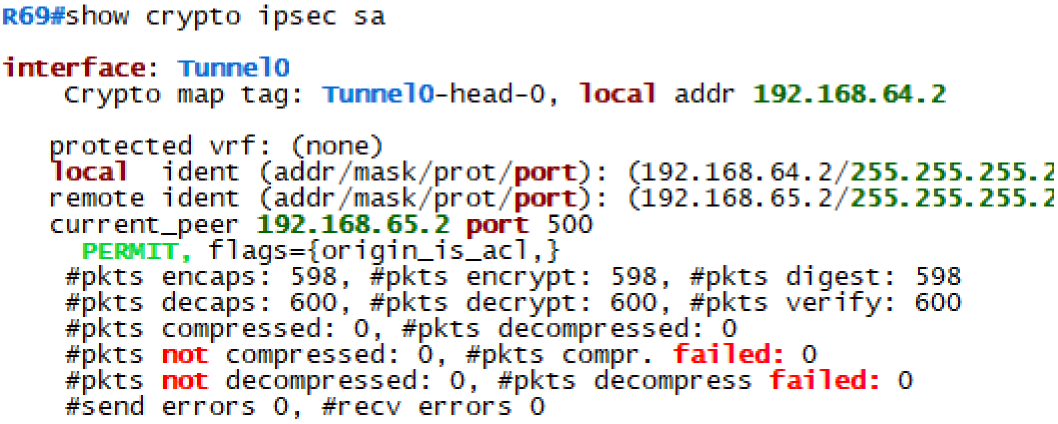
```
R68#show crypto ipsec sa

interface: Tunnel0
    Crypto map tag: Tunnel0-head-0, local addr 192.168.65.2

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (192.168.65.2/255.255.255.255/47/0)
   remote ident (addr/mask/prot/port): (192.168.64.2/255.255.255.255/47/0)
   current_peer 192.168.64.2 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 127, #pkts encrypt: 127, #pkts digest: 127
    #pkts decaps: 133, #pkts decrypt: 133, #pkts verify: 133
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 192.168.65.2, remote crypto endpt.: 192.168.64.2
     plaintext mtu 1446, path mtu 1500, ip mtu 1500, ip mtu idb Ethernet0/3
     current outbound spi: 0xDC1D0919(3692890393)
     PFS (Y/N): N, DH group: none

     inbound esp sas:
      spi: 0xEC567459(3965088857)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 1, flow_id: SW:1, sibling_flags 80000040, crypto map: Tunnel0-head-0
        sa timing: remaining key lifetime (k/sec): (4165565/3105)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0xDC1D0919(3692890393)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2, flow_id: SW:2, sibling_flags 80000040, crypto map: Tunnel0-head-0
        sa timing: remaining key lifetime (k/sec): (4165566/3105)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     outbound ah sas:

     outbound pcp sas:
```



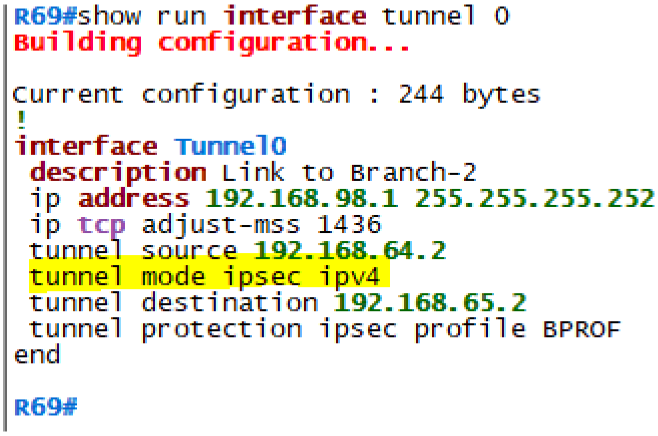
```
R68#sh crypto isakmp policy

Global IKE policy
Protection suite of priority 10
        encryption algorithm:   Three key triple DES
        hash algorithm:         Message Digest 5
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #5 (1536 bit)
        lifetime:               86400 seconds, no volume limit
Protection suite of priority 20
        encryption algorithm:   Three key triple DES
        hash algorithm:         Message Digest 5
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #5 (1536 bit)
        lifetime:               86400 seconds, no volume limit
```

IPsec Tunnel Interfaces Using Static VTI

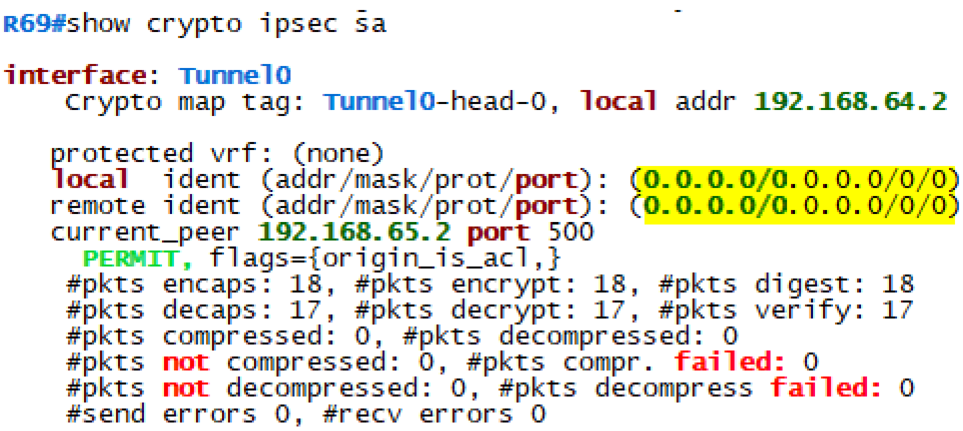
- By default, the IPsec SA will be created for the tunnel source and destination IP using 'GRE' (protocol 47) to carry multicast traffic.



```
R69#show crypto ipsec sa

interface: Tunnel0
    Crypto map tag: Tunnel0-head-0, local addr 192.168.64.2

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (192.168.64.2/255.255.255.2
   remote ident (addr/mask/prot/port): (192.168.65.2/255.255.255.2
   current_peer 192.168.65.2 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 598, #pkts encrypt: 598, #pkts digest: 598
    #pkts decaps: 600, #pkts decrypt: 600, #pkts verify: 600
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0
```



```
R69#show run interface tunnel 0
Building configuration...

Current configuration : 244 bytes
!
interface Tunnel0
 description Link to Branch-2
 ip address 192.168.98.1 255.255.255.252
 ip tcp adjust-mss 1436
 tunnel source 192.168.64.2
 tunnel mode ipsec ipv4
 tunnel destination 192.168.65.2
 tunnel protection ipsec profile BPROF
end

R69#
```



```
R69#show crypto ipsec sa

interface: Tunnel0
    Crypto map tag: Tunnel0-head-0, local addr 192.168.64.2

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (0.0.0.0/0.0.0.0/0/0)
   remote ident (addr/mask/prot/port): (0.0.0.0/0.0.0.0/0/0)
   current_peer 192.168.65.2 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 18, #pkts encrypt: 18, #pkts digest: 18
    #pkts decaps: 17, #pkts decrypt: 17, #pkts verify: 17
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0
```

3.1.d Implement and troubleshoot DMVPN [single hub]

DMVPN stands for Dynamic Multipoint VPN and it is an effective solution for dynamic secure overlay networks. In short, DMVPN is combination of the following technologies:

- Multipoint GRE (mGRE)
- Next-Hop Resolution Protocol (NHRP)
- Dynamic Routing Protocol (EIGRP, RIP, OSPF, BGP)
- Dynamic IPsec encryption
- Cisco Express Forwarding (CEF)

DMVPN Terminology

- NBMA IP Address – Typically a public IP address on internet facing interface
- Tunnel IP Address – GRE Tunnel interface IP address
- NHS (Next Hop Server) – DMVPN Hub Router(s)
- NHC (Next Hop Client) – DMVPN Spoke Router(s)

Type of IP Addresses in GRE or mGRE with DMVPN

- Hub & Spokes Public IP address are called Infrastructure IP Address, Outside IP Address, Service Provider Address or NBMA IP Address, all names has same meaning.
- Hub & Spokes Private IP Addresses (Tunnel IP Address) are also called Enterprise Addressing space, Inside address.

3.1.d [i] NHRP

NHRP (Next Hop Resolution Protocol)

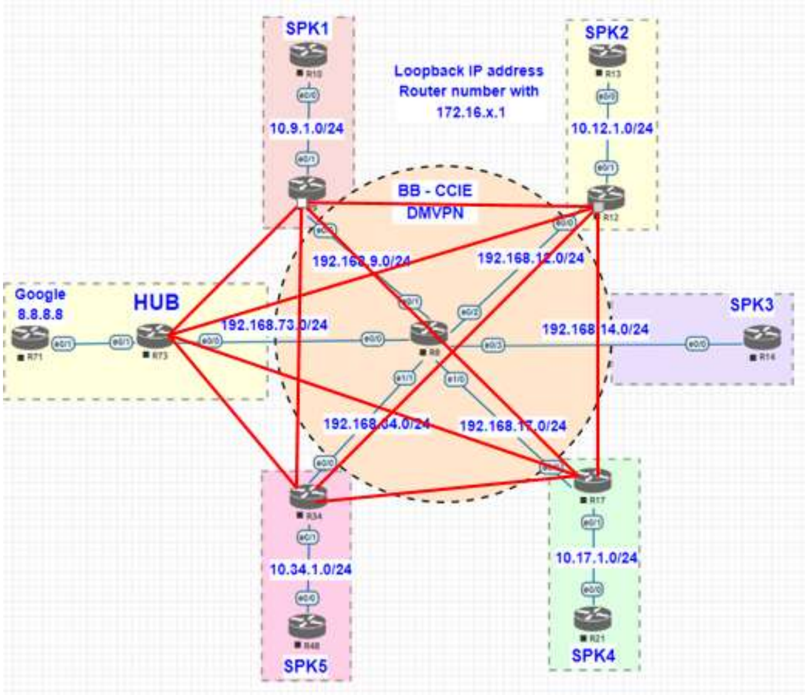
NHRP required mappings for the Spoke (SPK1) figure out what the public IP address is of the SPK2 router and rest of the Spokes, we do this with a protocol called NHRP (Next Hop Resolution Protocol).

One router will be the NHRP server. All other routers will be NHRP clients.

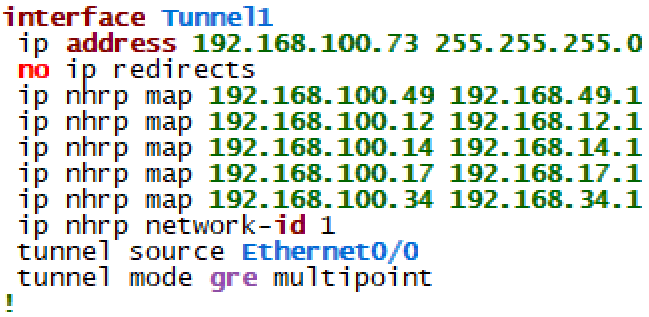
NHRP clients register themselves with the NHRP server and report their public IP address.

The NHRP server keeps track of all public IP addresses in its cache.
When one router wants to tunnel something to another router, it will request the NHRP server for the public IP address of the other router.

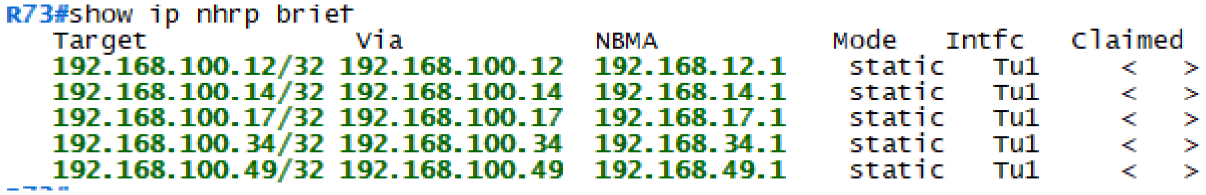
NHRP uses this server and clients model



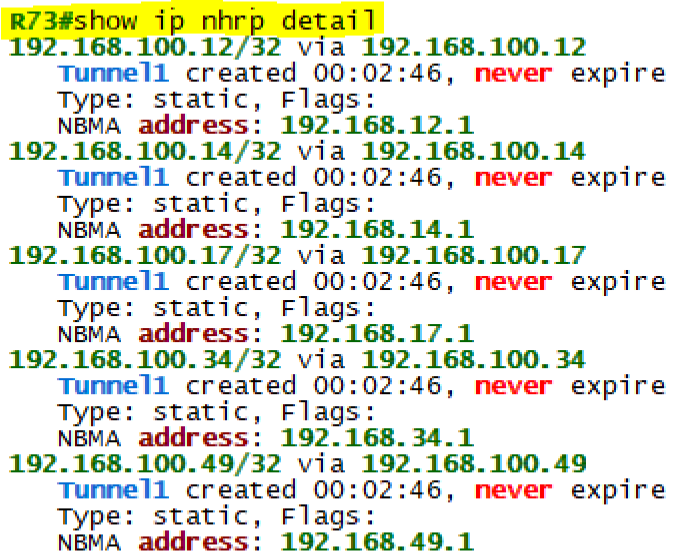
Required mapping on all the spokes to talk each other below config on R73, R49,R12,R14,R17,R34



```
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 ip nhrp map 192.168.100.49 192.168.49.1
 ip nhrp map 192.168.100.12 192.168.12.1
 ip nhrp map 192.168.100.14 192.168.14.1
 ip nhrp map 192.168.100.17 192.168.17.1
 ip nhrp map 192.168.100.34 192.168.34.1
 ip nhrp network-id 1
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
```

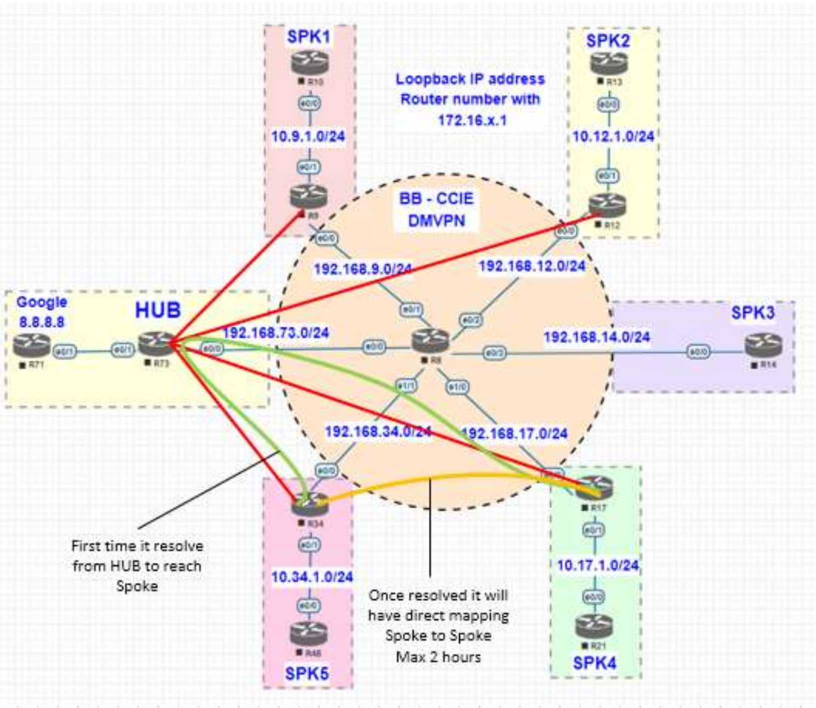


```
R73#show ip nhrp brief
   Target             Via            NBMA           Mode   Intfc   Claimed
   192.168.100.12/32 192.168.100.12 192.168.12.1    static   Tu1     <   >
   192.168.100.14/32 192.168.100.14 192.168.14.1    static   Tu1     <   >
   192.168.100.17/32 192.168.100.17 192.168.17.1    static   Tu1     <   >
   192.168.100.34/32 192.168.100.34 192.168.34.1    static   Tu1     <   >
   192.168.100.49/32 192.168.100.49 192.168.49.1    static   Tu1     <   >
```



```
R73#show ip nhrp detail
192.168.100.12/32 via 192.168.100.12
   Tunnel1 created 00:02:46, never expire
   Type: static, Flags:
   NBMA address: 192.168.12.1
192.168.100.14/32 via 192.168.100.14
   Tunnel1 created 00:02:46, never expire
   Type: static, Flags:
   NBMA address: 192.168.14.1
192.168.100.17/32 via 192.168.100.17
   Tunnel1 created 00:02:46, never expire
   Type: static, Flags:
   NBMA address: 192.168.17.1
192.168.100.34/32 via 192.168.100.34
   Tunnel1 created 00:02:46, never expire
   Type: static, Flags:
   NBMA address: 192.168.34.1
192.168.100.49/32 via 192.168.100.49
   Tunnel1 created 00:02:46, never expire
   Type: static, Flags:
   NBMA address: 192.168.49.1
```

3.1.d [ii] DMVPN with IPsec using preshared key

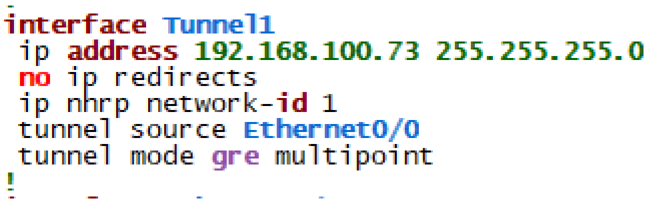


Loopback IP address Router number with 172.16.x.1

First time it resolve from HUB to reach Spoke

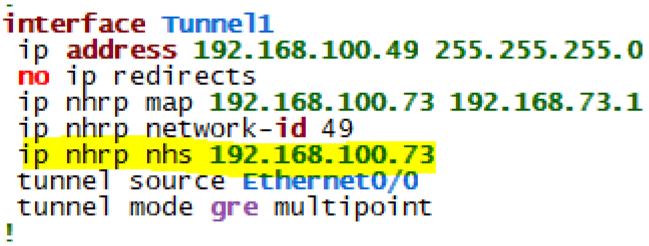
Once resolved it will have direct mapping Spoke to Spoke Max 2 hours

We Configure R73 asDMVPN HUB and rest all will be SPOKE



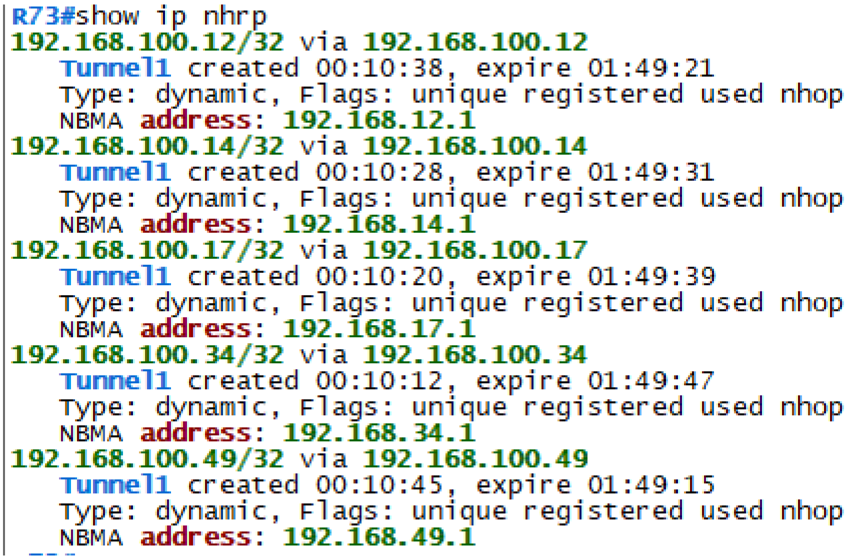
```
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 ip nhrp network-id 1
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
```

R49 Spoke config :



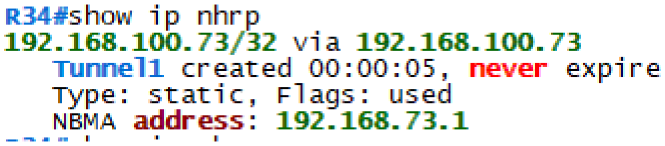
```
interface Tunnel1
 ip address 192.168.100.49 255.255.255.0
 no ip redirects
 ip nhrp map 192.168.100.73 192.168.73.1
 ip nhrp network-id 49
 ip nhrp nhs 192.168.100.73
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
```

R73 Mappings

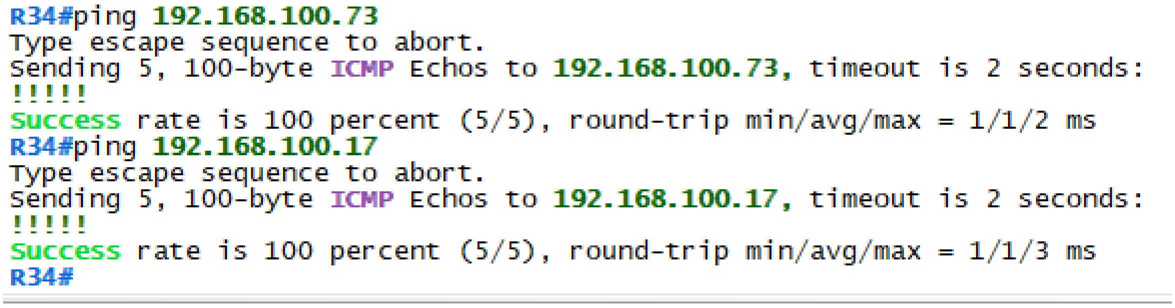


```
R73#show ip nhrp
192.168.100.12/32 via 192.168.100.12
   Tunnel1 created 00:10:38, expire 01:49:21
   Type: dynamic, Flags: unique registered used nhop
   NBMA address: 192.168.12.1
192.168.100.14/32 via 192.168.100.14
   Tunnel1 created 00:10:28, expire 01:49:31
   Type: dynamic, Flags: unique registered used nhop
   NBMA address: 192.168.14.1
192.168.100.17/32 via 192.168.100.17
   Tunnel1 created 00:10:20, expire 01:49:39
   Type: dynamic, Flags: unique registered used nhop
   NBMA address: 192.168.17.1
192.168.100.34/32 via 192.168.100.34
   Tunnel1 created 00:10:12, expire 01:49:47
   Type: dynamic, Flags: unique registered used nhop
   NBMA address: 192.168.34.1
192.168.100.49/32 via 192.168.100.49
   Tunnel1 created 00:10:45, expire 01:49:15
   Type: dynamic, Flags: unique registered used nhop
   NBMA address: 192.168.49.1
```

Now we verify from R34 reaching R17

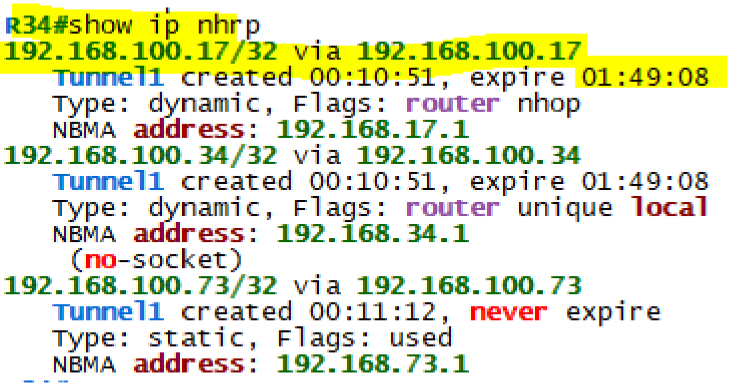


```
R34#show ip nhrp
192.168.100.73/32 via 192.168.100.73
   Tunnel1 created 00:00:05, never expire
   Type: static, Flags: used
   NBMA address: 192.168.73.1
```



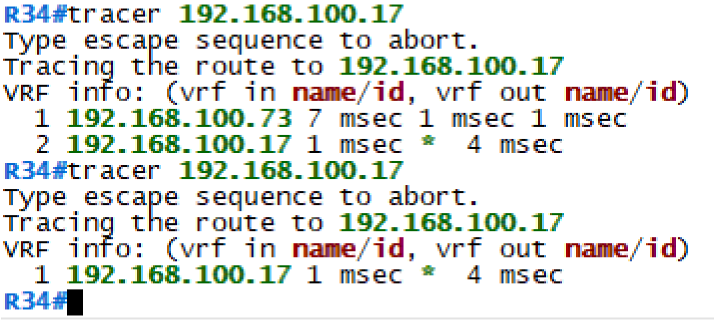
```
R34#ping 192.168.100.73
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.73, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
R34#ping 192.168.100.17
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.17, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/3 ms
R34#
```

Now we can see Mappings for R17



```
R34#show ip nhrp
192.168.100.17/32 via 192.168.100.17
   Tunnel1 created 00:10:51, expire 01:49:08
   Type: dynamic, Flags: router nhop
   NBMA address: 192.168.17.1
192.168.100.34/32 via 192.168.100.34
   Tunnel1 created 00:10:51, expire 01:49:08
   Type: dynamic, Flags: router unique local
   NBMA address: 192.168.34.1
    (no-socket)
192.168.100.73/32 via 192.168.100.73
   Tunnel1 created 00:11:12, never expire
   Type: static, Flags: used
   NBMA address: 192.168.73.1
```

Lets we traceroute and verify, first time it will got to Hub R73 to resolve, then onwards it has direct connection with R17

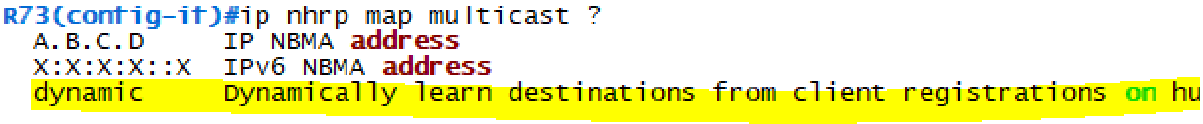


```
R34#tracer 192.168.100.17
Type escape sequence to abort.
Tracing the route to 192.168.100.17
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.100.73 7 msec 1 msec 1 msec
  2 192.168.100.17 1 msec *  4 msec
R34#tracer 192.168.100.17
Type escape sequence to abort.
Tracing the route to 192.168.100.17
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.100.17 1 msec *  4 msec
R34#
```

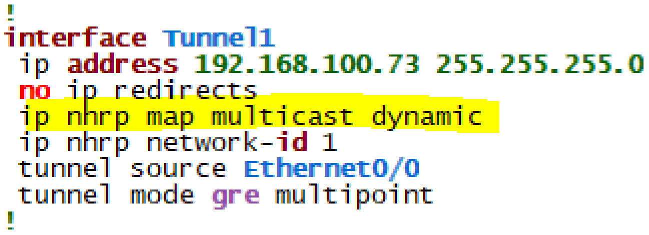
DMVPN with Routing protocol :

By Default Multicast will be dropped, since Hub and Spoke do not know where to send multicast traffic.

So we need to map at spoke – that all the multicast traffic will be forward to Hub
On Hub it will be mapped as Dynamic.

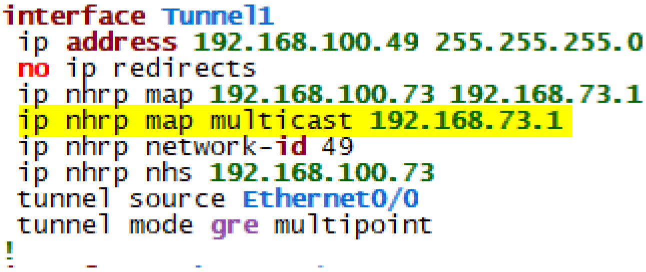


```
R73(config-if)#ip nhrp map multicast ?
  A.B.C.D      IP NBMA address
  X:X:X:X::X   IPv6 NBMA address
  dynamic      Dynamically learn destinations from client registrations on hu
```



```
!
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 ip nhrp map multicast dynamic
 ip nhrp network-id 1
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
```

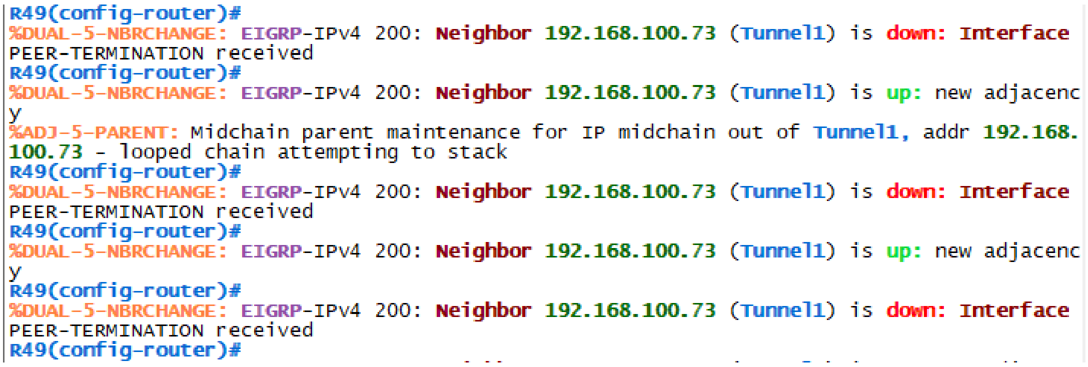
Spoke (we need to mentioned HUB Public IP not the Tunnel IP Address)



```
interface Tunnel1
 ip address 192.168.100.49 255.255.255.0
 no ip redirects
 ip nhrp map 192.168.100.73 192.168.73.1
 ip nhrp map multicast 192.168.73.1
 ip nhrp network-id 49
 ip nhrp nhs 192.168.100.73
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
```

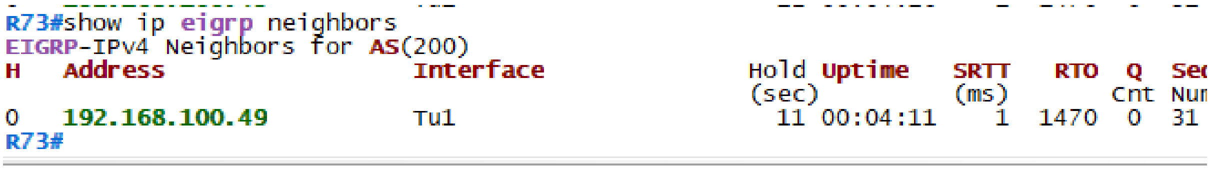
Now we will run EIGRP on DMVPN and Verify

Note : On EIGRP network statement, do not Announce WAN IP address, this will have different issue as per below logs

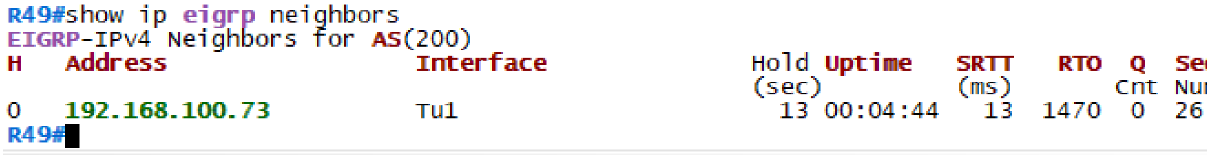


```
R49(config-router)#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is down: Interface
PEER-TERMINATION received
R49(config-router)#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is up: new adjacenc
y
%ADJ-5-PARENT: Midchain parent maintenance for IP midchain out of Tunnel1, addr 192.168.
100.73 - looped chain attempting to stack
R49(config-router)#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is down: Interface
PEER-TERMINATION received
R49(config-router)#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is up: new adjacenc
y
R49(config-router)#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is down: Interface
PEER-TERMINATION received
R49(config-router)#
```

Lets check R73 and R49 Eigrp neighbours



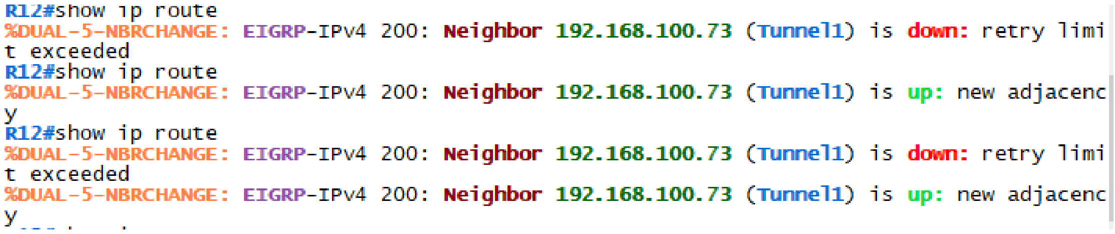
```
R73#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Se
                                                   (sec)         (ms)       Cnt Nu
0   192.168.100.49          Tu1                      11 00:04:11    1  1470  0  31
R73#
```



```
R49#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Se
                                                   (sec)         (ms)       Cnt Nu
0   192.168.100.73          Tu1                      13 00:04:44   13  1470  0  26
R49#
```

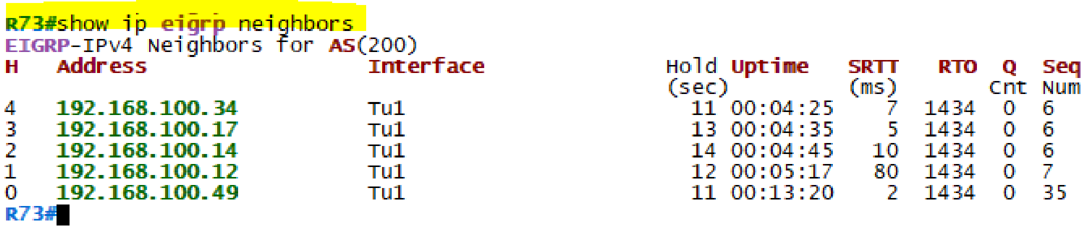
Since Rest of the spokes we do not configured Multicast so Eigrp not form the neighbourship we see the logs like below :

R12



```
R12#show ip route
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is down: retry limi
t exceeded
R12#show ip route
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is up: new adjacenc
y
R12#show ip route
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is down: retry limi
t exceeded
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is up: new adjacenc
y
```

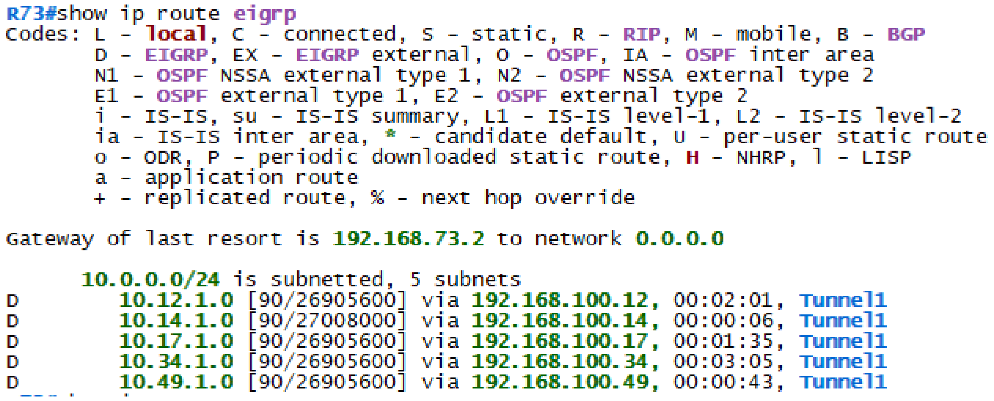
Once we enable the Multicast mapping we see the 2 neighbours on R73



```
R73#show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(200)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
4   192.168.100.34          Tu1                      11 00:04:25    7  1434  0  6
3   192.168.100.17          Tu1                      13 00:04:35    5  1434  0  6
2   192.168.100.14          Tu1                      14 00:04:45   10  1434  0  6
1   192.168.100.12          Tu1                      12 00:05:17   80  1434  0  7
0   192.168.100.49          Tu1                      11 00:13:20    2  1434  0  35
R73#
```

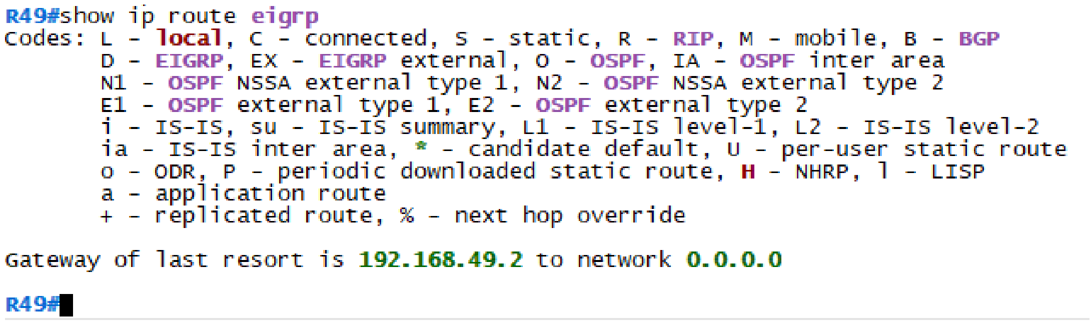
Now we will have another issue, now we can only see the routes on Hub, and same Routes not available on spoke, due to EIGRP rule, the route learn from same interface cannot announce back.

R73



```
R73#show ip route eigrp
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.73.2 to network 0.0.0.0

      10.0.0.0/24 is subnetted, 5 subnets
D        10.12.1.0 [90/26905600] via 192.168.100.12, 00:02:01, Tunnel1
D        10.14.1.0 [90/27008000] via 192.168.100.14, 00:00:06, Tunnel1
D        10.17.1.0 [90/26905600] via 192.168.100.17, 00:01:35, Tunnel1
D        10.34.1.0 [90/26905600] via 192.168.100.34, 00:03:05, Tunnel1
D        10.49.1.0 [90/26905600] via 192.168.100.49, 00:00:43, Tunnel1
```

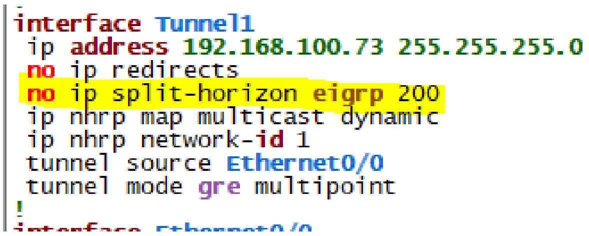
R49 there is no routes in related to DMVPN



```
R49#show ip route eigrp
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.49.2 to network 0.0.0.0

R49#
```

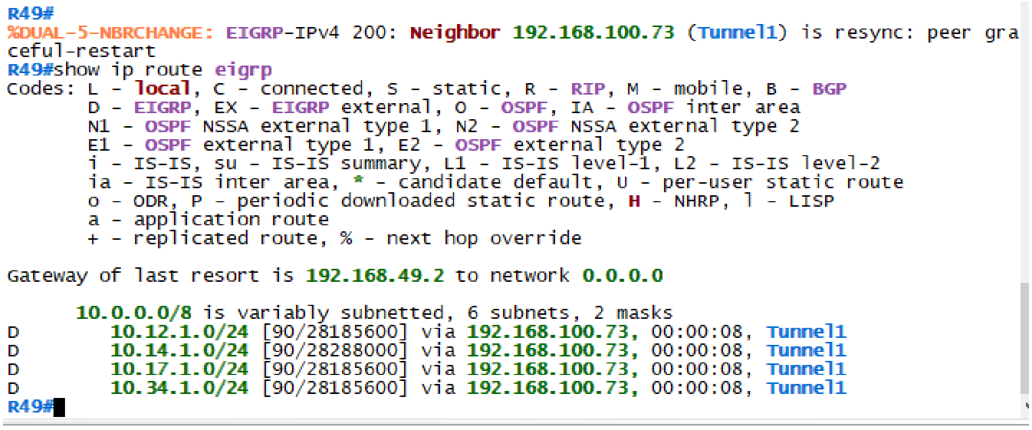
Now we need to enable split-horizon on Hub Tunnel interface to fix this issue.

R73



```
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 no ip split-horizon eigrp 200
 ip nhrp map multicast dynamic
 ip nhrp network-id 1
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
interface Ethernet0/0
```

Verify the routes on R49, now we have updated route from other spokes.

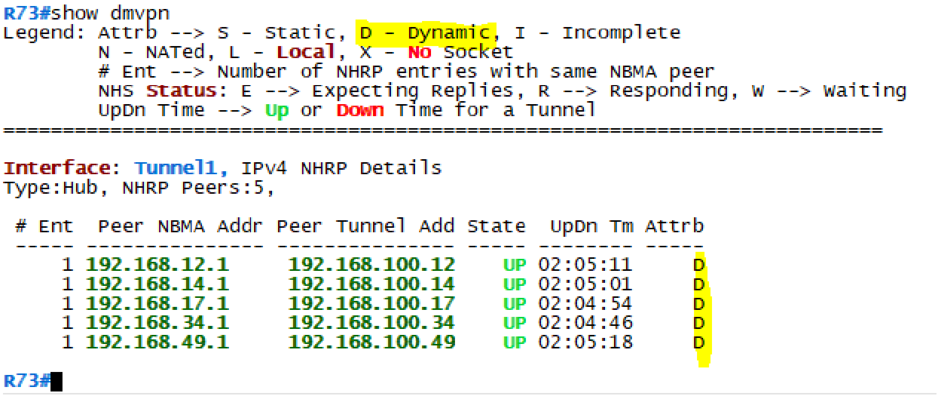


```
R49#
%DUAL-5-NBRCHANGE: EIGRP-IPv4 200: Neighbor 192.168.100.73 (Tunnel1) is resync: peer gra
ceful-restart
R49#show ip route eigrp
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.49.2 to network 0.0.0.0

      10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
D        10.12.1.0/24 [90/28185600] via 192.168.100.73, 00:00:08, Tunnel1
D        10.14.1.0/24 [90/28288000] via 192.168.100.73, 00:00:08, Tunnel1
D        10.17.1.0/24 [90/28185600] via 192.168.100.73, 00:00:08, Tunnel1
D        10.34.1.0/24 [90/28185600] via 192.168.100.73, 00:00:08, Tunnel1
R49#
```

DMVPN Phase 1 -

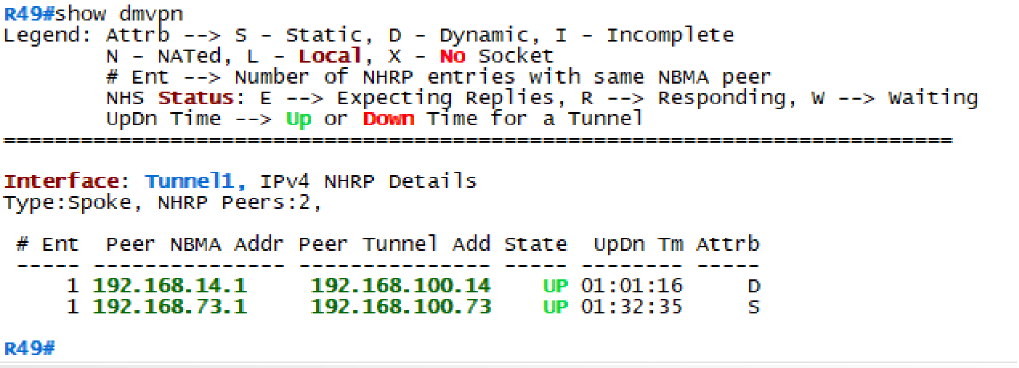
- Hub is configured with mGRE tunnel and Spokes are configured with point-to-point GRE tunnel with the physical IP address of the HUB as the tunnel destination.
- Spoke-to-spoke communication has to go through the hub.
- Benefit is simplified hub router configuration, which does not require static NHRP mapping for every new spoke.
- Recommended routing:
  – The hub advertises a default route to the spokes.
  – Spokes advertise their subnets to the hub.

With the above config Phase1 configure we can check DMVPN as below



```
R73#show dmvpn
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel1, IPv4 NHRP Details
Type:Hub, NHRP Peers:5,

 # Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
 ----- --------------- --------------- ----- -------- -----
     1 192.168.12.1     192.168.100.12    UP 02:05:11     D
     1 192.168.14.1     192.168.100.14    UP 02:05:01     D
     1 192.168.17.1     192.168.100.17    UP 02:04:54     D
     1 192.168.34.1     192.168.100.34    UP 02:04:46     D
     1 192.168.49.1     192.168.100.49    UP 02:05:18     D

R73#
```

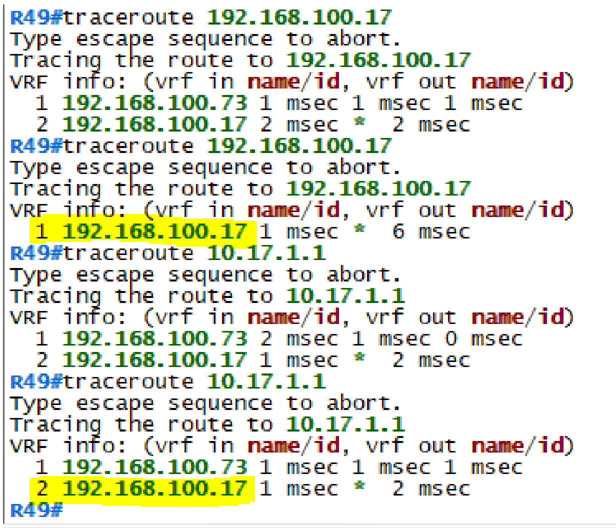
Directly Connected neworks will go directly, indirectly connected routes always go via hub, now we verify from R49



```
R49#show dmvpn
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel1, IPv4 NHRP Details
Type:Spoke, NHRP Peers:2,

 # Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
 ----- --------------- --------------- ----- -------- -----
     1 192.168.14.1     192.168.100.14    UP 01:01:16     D
     1 192.168.73.1     192.168.100.73    UP 01:32:35     S

R49#
```

We can observe that 192.168.100.17 directly connected to go directly

10.17.1.1 not directly connected always go via Hub



```
R49#traceroute 192.168.100.17
Type escape sequence to abort.
Tracing the route to 192.168.100.17
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.100.73 1 msec 1 msec 1 msec
  2 192.168.100.17 2 msec *  2 msec
R49#traceroute 192.168.100.17
Type escape sequence to abort.
Tracing the route to 192.168.100.17
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.100.17 1 msec *  6 msec
R49#traceroute 10.17.1.1
Type escape sequence to abort.
Tracing the route to 10.17.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.100.73 2 msec 1 msec 0 msec
  2 192.168.100.17 1 msec *  2 msec
R49#traceroute 10.17.1.1
Type escape sequence to abort.
Tracing the route to 10.17.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.100.73 1 msec 1 msec 1 msec
  2 192.168.100.17 1 msec *  2 msec
R49#
```

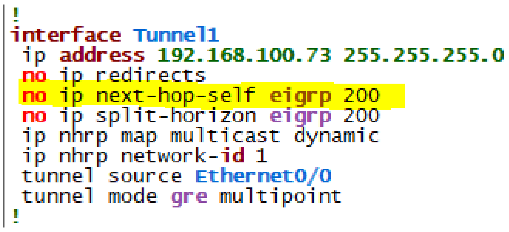
DMVPN Phase 2 –

- The only difference in this phase is that the spokes can form an IPsec tunnel directly with the other spokes instead of forcing the traffic to go through the hub as in the case of Phase 1.
- The tunnel interfaces on the hub and spoke are all mGRE encapsulated.
- There is a CEF entry for a prefix learnt via the overlay routing protocol with the next-hop set to the tunnel IP address of the router from where it originated. At this point the CEF entry for the next-hop is marked as "glean", meaning it needs L3 to L2 lookup to be performed. This L3 to L2 lookup is performed by NHRP, when an initial packet is being sent to the destination prefix.
- The spokes are also the NHCs so they register themselves with the NHS and also request the NHS for the IP-to-NBMA mapping information of the spoke they want to peer with.
- Using that mapping information they form IPsec tunnels with the spokes for which you have to use either a wildcard pre-shared key or specific keys for each of the other spoke they want to peer with.

One additional command should change from Phase1 to Phase2 on the HUB

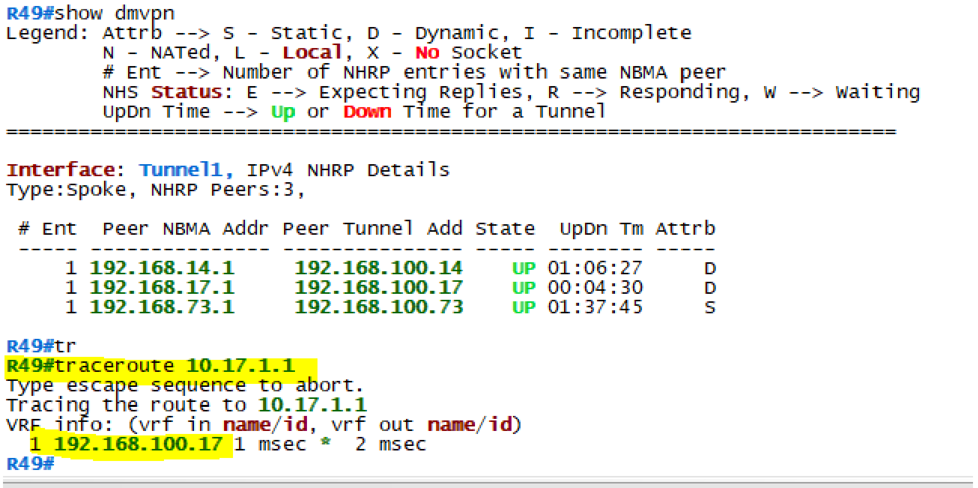
no ip next-hop-self eigrp 200

Now we configured and verify the same.

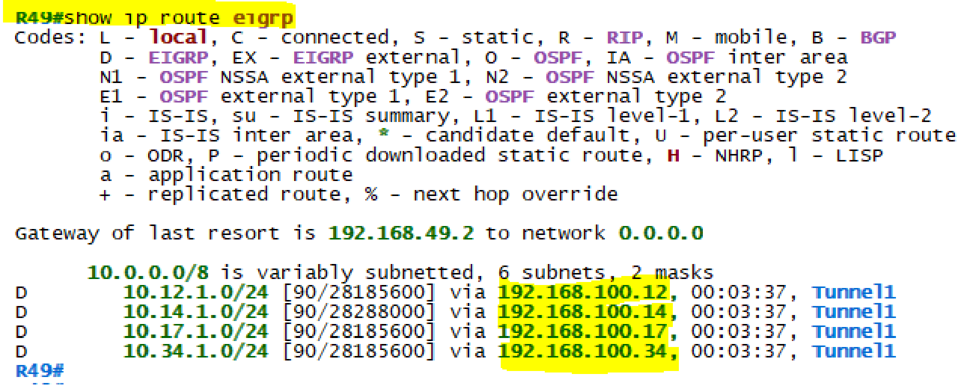


```
!
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 no ip next-hop-self eigrp 200
 no ip split-horizon eigrp 200
 ip nhrp map multicast dynamic
 ip nhrp network-id 1
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
```

We verify the same from R49



```
R49#show dmvpn
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel1, IPv4 NHRP Details
Type:Spoke, NHRP Peers:3,

 # Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
 ----- --------------- --------------- ----- -------- -----
     1 192.168.14.1     192.168.100.14    UP 01:06:27     D
     1 192.168.17.1     192.168.100.17    UP 00:04:30     D
     1 192.168.73.1     192.168.100.73    UP 01:37:45     S

R49#tr
R49#traceroute 10.17.1.1
Type escape sequence to abort.
Tracing the route to 10.17.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.100.17 1 msec *  2 msec
R49#
```



```
 R49#show ip route eigrp
 Codes: L - local, 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, H - NHRP, l - LISP
        a - application route
        + - replicated route, % - next hop override

 Gateway of last resort is 192.168.49.2 to network 0.0.0.0

       10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
 D        10.12.1.0/24 [90/28185600] via 192.168.100.12, 00:03:37, Tunnel1
 D        10.14.1.0/24 [90/28288000] via 192.168.100.14, 00:03:37, Tunnel1
 D        10.17.1.0/24 [90/28185600] via 192.168.100.17, 00:03:37, Tunnel1
 D        10.34.1.0/24 [90/28185600] via 192.168.100.34, 00:03:37, Tunnel1
 R49#
```

DMVPN Phase 3 -

- It borrows the benefit of summarization from Phase 1 and spoke-to-spoke tunnels from Phase 2.
- NHRP redirect configured on the hub tells the initiator spoke to look for a better path to the destination spoke. Upon receiving the NHRP redirect message the spokes communicate with each other over the hub and they have their NHRP replies for the NHRP Resolution Requests that they sent out.
- NHRP Shortcut configured on the spoke updates the CEF table. It basically overrides the next-hop value for a remote spoke network from the default initial hub tunnel IP address to the NHRP resolved remote spoke tunnel IP address.
- NHRP Resolution Requests in DMVPN Phase 2 is done for the next-hop IP address, which is identified from the routing table; this is similar to how Ethernet ARP works for a static route configured with a next-hop IP address.

- NHRP Resolution Requests in DMVPN Phase3 is done for the destination network being accessed; this is similar to how Ethernet ARP works for a static route configured with a multipoint exit.

Required config on HUB

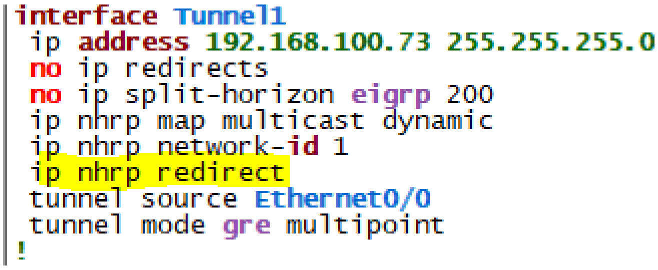
Ip nhrp redirect

Required config on SPOKE

Ip nhrp shortcut

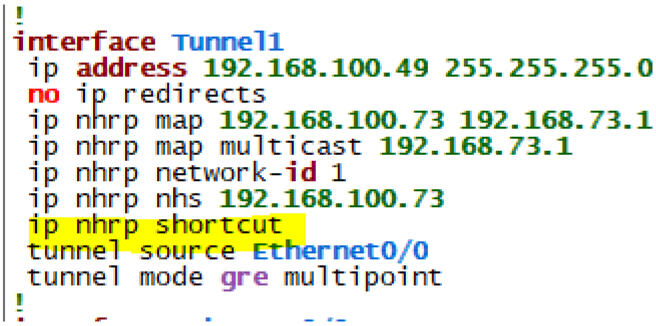
We will configure and verify the same

HUB R73



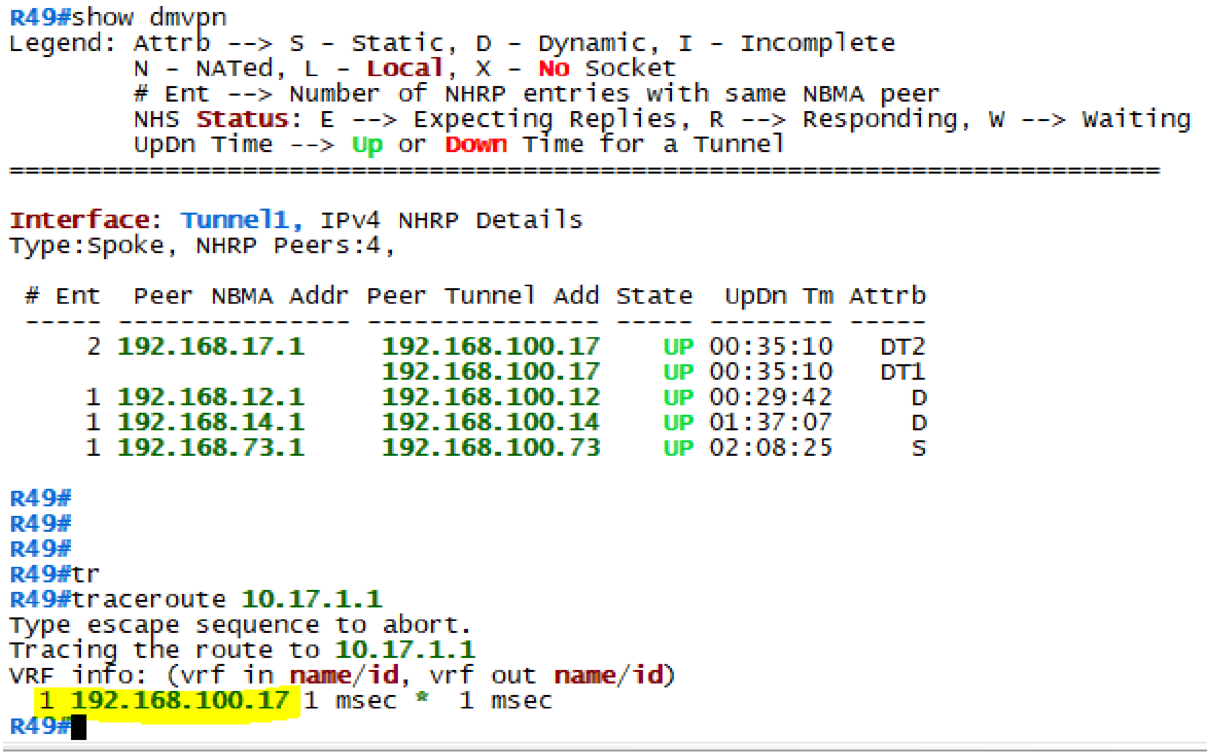
```
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 no ip split-horizon eigrp 200
 ip nhrp map multicast dynamic
 ip nhrp network-id 1
 ip nhrp redirect
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
```

R49 SPOKE



```
!
interface Tunnel1
 ip address 192.168.100.49 255.255.255.0
 no ip redirects
 ip nhrp map 192.168.100.73 192.168.73.1
 ip nhrp map multicast 192.168.73.1
 ip nhrp network-id 1
 ip nhrp nhs 192.168.100.73
 ip nhrp shortcut
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
!
```

- Initially there is no spoke-to-spoke communication
- Traffic between Spoke 1 and Spoke 2
- First packet would traverse via Hub and it will trigger NHRP shortcut switching process then traffic will start using direct spoke-to-spoke tunnel

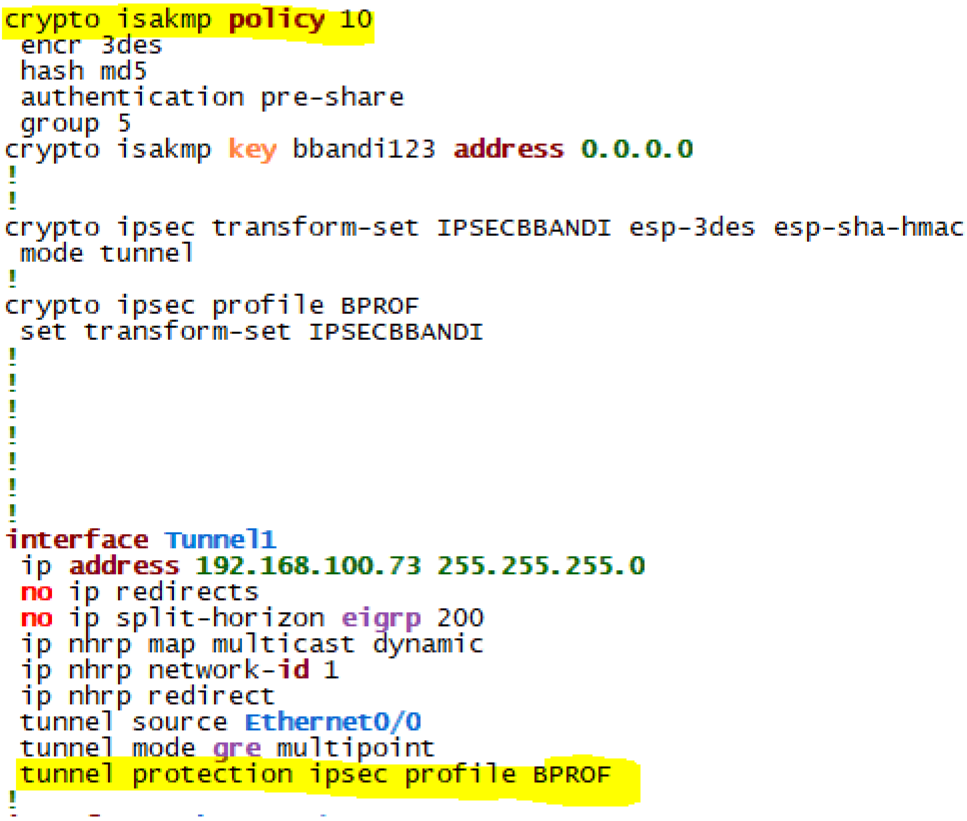


```
R49#show dmvpn
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel1, IPv4 NHRP Details
Type:Spoke, NHRP Peers:4,

 # Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
 ----- --------------- --------------- ----- -------- -----
     2 192.168.17.1     192.168.100.17    UP 00:35:10   DT2
                        192.168.100.17    UP 00:35:10   DT1
     1 192.168.12.1     192.168.100.12    UP 00:29:42     D
     1 192.168.14.1     192.168.100.14    UP 01:37:07     D
     1 192.168.73.1     192.168.100.73    UP 02:08:25     S

R49#
R49#
R49#
R49#tr
R49#traceroute 10.17.1.1
Type escape sequence to abort.
Tracing the route to 10.17.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.100.17 1 msec *  1 msec
R49#
```

```
*T1 - Route Installed
*T2 - Nexthop-override
```

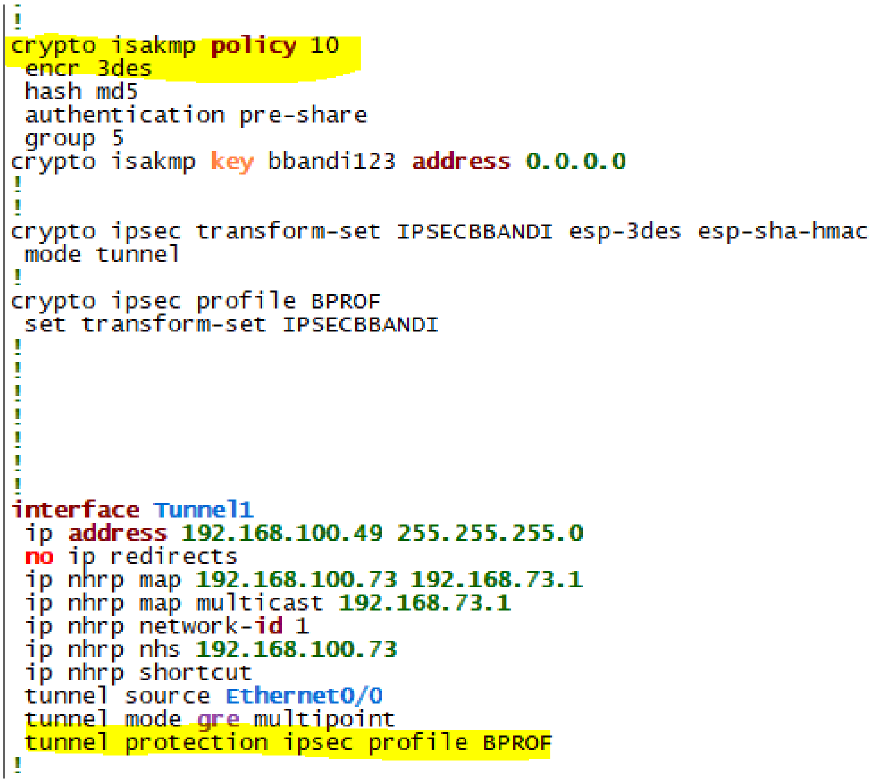
DMVPN with IPSEC

R73 HUB



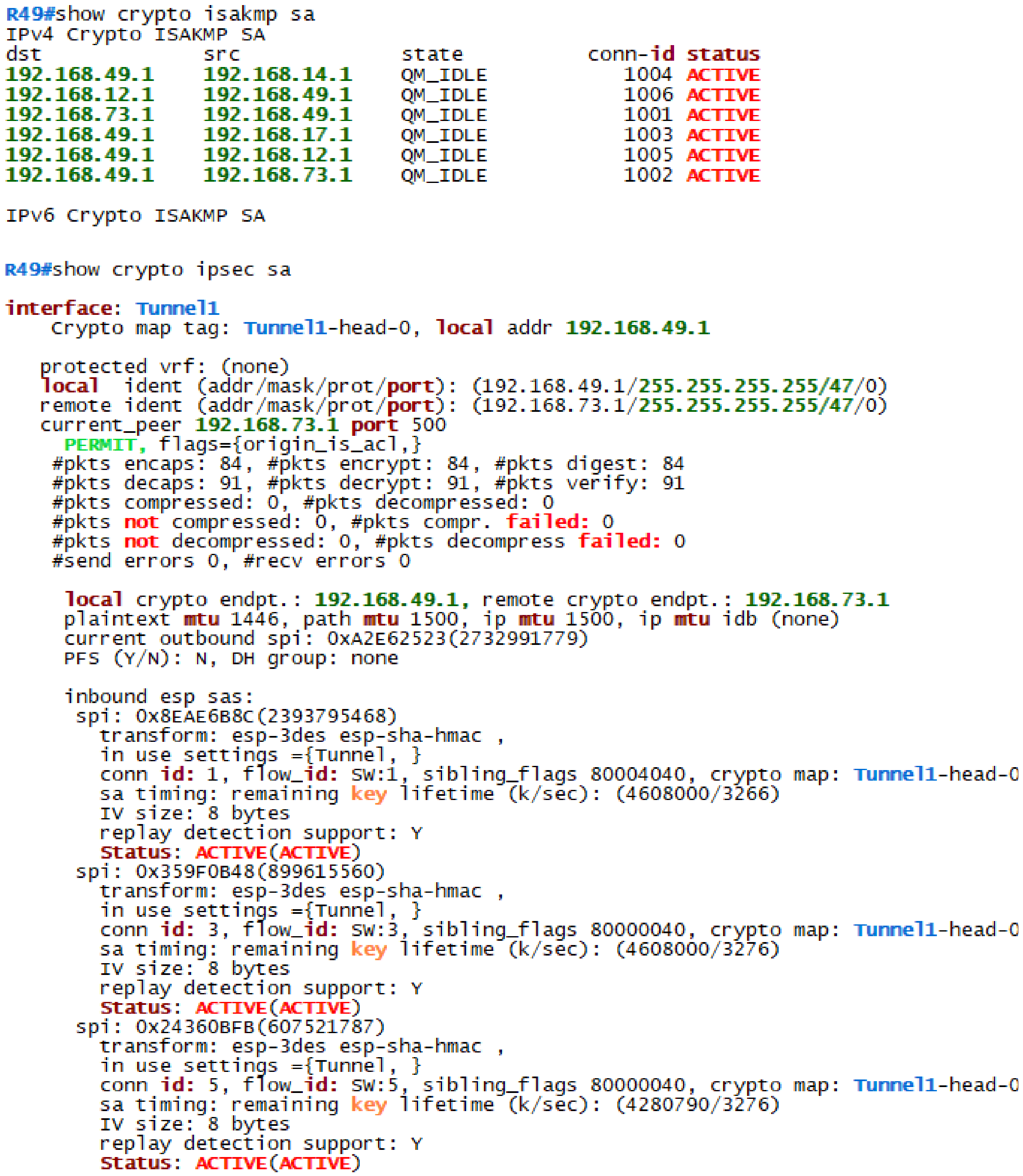
```
crypto isakmp policy 10
 encr 3des
 hash md5
 authentication pre-share
 group 5
crypto isakmp key bbandi123 address 0.0.0.0
!
!
crypto ipsec transform-set IPSECBBANDI esp-3des esp-sha-hmac
 mode tunnel
!
crypto ipsec profile BPROF
 set transform-set IPSECBBANDI
!
!
!
!
!
!
!
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 no ip split-horizon eigrp 200
 ip nhrp map multicast dynamic
 ip nhrp network-id 1
 ip nhrp redirect
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
 tunnel protection ipsec profile BPROF
!
```

Spoke config



```
!
crypto isakmp policy 10
 encr 3des
 hash md5
 authentication pre-share
 group 5
crypto isakmp key bbandi123 address 0.0.0.0
!
!
crypto ipsec transform-set IPSECBBANDI esp-3des esp-sha-hmac
 mode tunnel
!
crypto ipsec profile BPROF
 set transform-set IPSECBBANDI
!
!
!
!
!
!
!
interface Tunnel1
 ip address 192.168.100.49 255.255.255.0
 no ip redirects
 ip nhrp map 192.168.100.73 192.168.73.1
 ip nhrp map multicast 192.168.73.1
 ip nhrp network-id 1
 ip nhrp nhs 192.168.100.73
 ip nhrp shortcut
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
 tunnel protection ipsec profile BPROF
!
```

R49

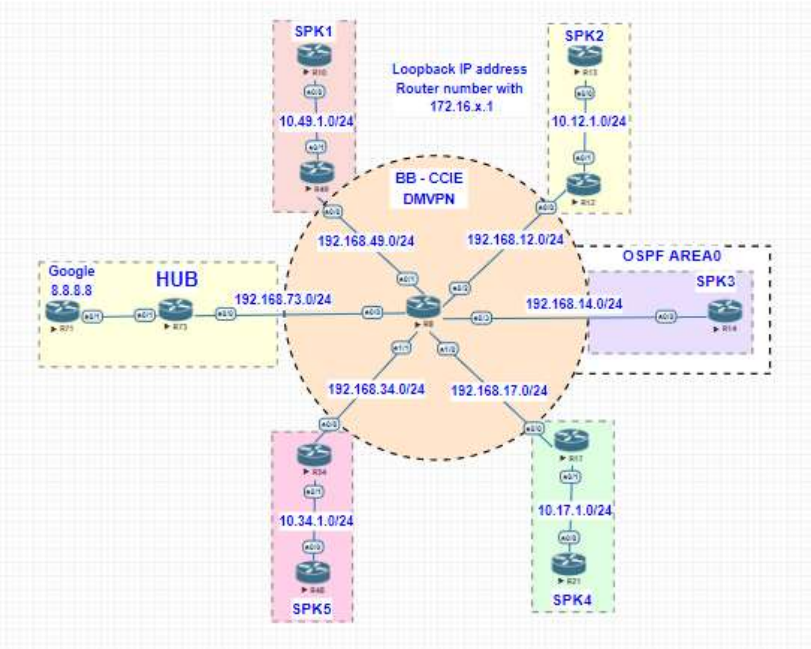


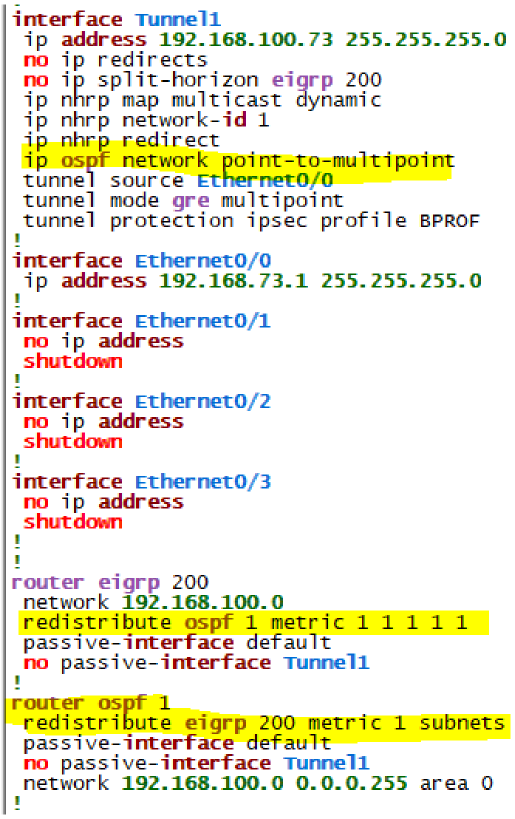
```
R49#show crypto isakmp sa
IPv4 Crypto ISAKMP SA
dst             src             state          conn-id status
192.168.49.1    192.168.14.1    QM_IDLE           1004 ACTIVE
192.168.12.1    192.168.49.1    QM_IDLE           1006 ACTIVE
192.168.73.1    192.168.49.1    QM_IDLE           1001 ACTIVE
192.168.49.1    192.168.17.1    QM_IDLE           1003 ACTIVE
192.168.49.1    192.168.12.1    QM_IDLE           1005 ACTIVE
192.168.49.1    192.168.73.1    QM_IDLE           1002 ACTIVE

IPv6 Crypto ISAKMP SA

R49#show crypto ipsec sa

interface: Tunnel1
    Crypto map tag: Tunnel1-head-0, local addr 192.168.49.1

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (192.168.49.1/255.255.255.255/47/0)
   remote ident (addr/mask/prot/port): (192.168.73.1/255.255.255.255/47/0)
   current_peer 192.168.73.1 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 84, #pkts encrypt: 84, #pkts digest: 84
    #pkts decaps: 91, #pkts decrypt: 91, #pkts verify: 91
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 192.168.49.1, remote crypto endpt.: 192.168.73.1
     plaintext mtu 1446, path mtu 1500, ip mtu 1500, ip mtu idb (none)
     current outbound spi: 0xA2E62523(2732991779)
     PFS (Y/N): N, DH group: none

     inbound esp sas:
      spi: 0x8EAE6B8C(2393795468)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 1, flow_id: SW:1, sibling_flags 80004040, crypto map: Tunnel1-head-0
        sa timing: remaining key lifetime (k/sec): (4608000/3266)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)
      spi: 0x359F0B48(899615560)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 3, flow_id: SW:3, sibling_flags 80000040, crypto map: Tunnel1-head-0
        sa timing: remaining key lifetime (k/sec): (4608000/3276)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)
      spi: 0x24360BFB(607521787)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 5, flow_id: SW:5, sibling_flags 80000040, crypto map: Tunnel1-head-0
        sa timing: remaining key lifetime (k/sec): (4280790/3276)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)
```

For testing, I have mixed OSPF and EIGRP to test (I have not see mixing in real world, but since I got apportunity test)

HUB R73 have OSPF EIGRP

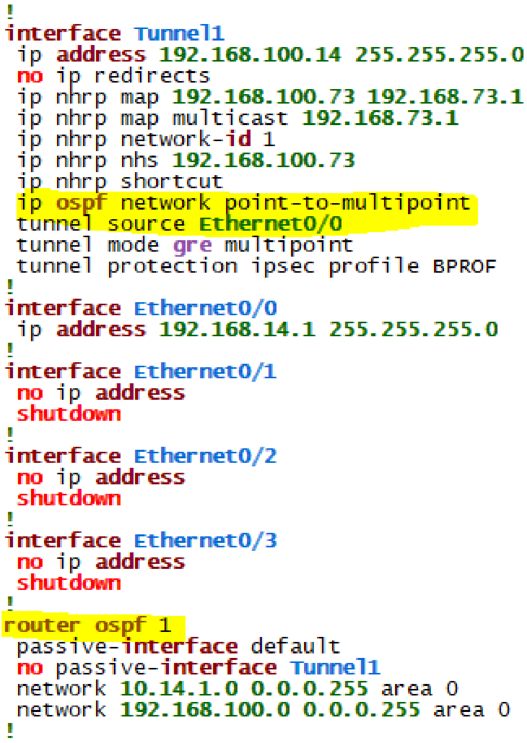
All SPOKE EIGRP and R14 only have OSPF Area 0 (R73 does the redistribution)



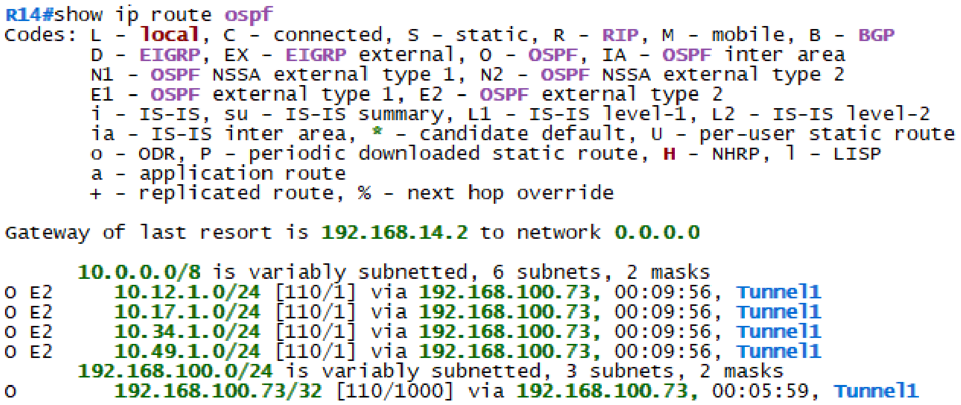


```
!
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 no ip split-horizon eigrp 200
 ip nhrp map multicast dynamic
 ip nhrp network-id 1
 ip nhrp redirect
 ip ospf network point-to-multipoint
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
 tunnel protection ipsec profile BPROF
!
interface Ethernet0/0
 ip address 192.168.73.1 255.255.255.0
!
interface Ethernet0/1
 no ip address
 shutdown
!
interface Ethernet0/2
 no ip address
 shutdown
!
interface Ethernet0/3
 no ip address
 shutdown
!
!
router eigrp 200
 network 192.168.100.0
 redistribute ospf 1 metric 1 1 1 1 1
 passive-interface default
 no passive-interface Tunnel1
!
router ospf 1
 redistribute eigrp 200 metric 1 subnets
 passive-interface default
 no passive-interface Tunnel1
 network 192.168.100.0 0.0.0.255 area 0
!
```

R14



```
!
interface Tunnel1
 ip address 192.168.100.14 255.255.255.0
 no ip redirects
 ip nhrp map 192.168.100.73 192.168.73.1
 ip nhrp map multicast 192.168.73.1
 ip nhrp network-id 1
 ip nhrp nhs 192.168.100.73
 ip nhrp shortcut
 ip ospf network point-to-multipoint
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
 tunnel protection ipsec profile BPROF
!
interface Ethernet0/0
 ip address 192.168.14.1 255.255.255.0
!
interface Ethernet0/1
 no ip address
 shutdown
!
interface Ethernet0/2
 no ip address
 shutdown
!
interface Ethernet0/3
 no ip address
 shutdown
!
router ospf 1
 passive-interface default
 no passive-interface Tunnel1
 network 10.14.1.0 0.0.0.255 area 0
 network 192.168.100.0 0.0.0.255 area 0
!
```



```
R14#show ip route ospf
Codes: L - local, 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, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.14.2 to network 0.0.0.0

      10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
O E2     10.12.1.0/24 [110/1] via 192.168.100.73, 00:09:56, Tunnel1
O E2     10.17.1.0/24 [110/1] via 192.168.100.73, 00:09:56, Tunnel1
O E2     10.34.1.0/24 [110/1] via 192.168.100.73, 00:09:56, Tunnel1
O E2     10.49.1.0/24 [110/1] via 192.168.100.73, 00:09:56, Tunnel1
      192.168.100.0/24 is variably subnetted, 3 subnets, 2 masks
O        192.168.100.73/32 [110/1000] via 192.168.100.73, 00:05:59, Tunnel1
```

3.1.d [iii] QoS profile

3.1.d [iv] Pre-classify

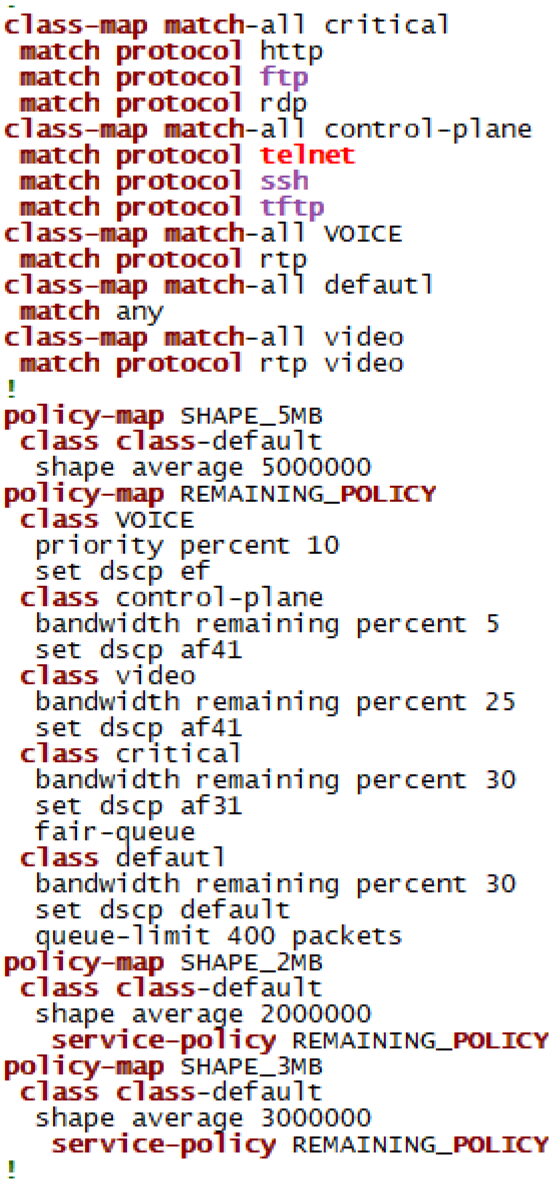
The Per-Tunnel QoS for DMVPN feature lets you apply a quality of service (QoS) policy on a Dynamic Multipoint VPN (DMVPN) hub on a per-tunnel instance (per-spoke basis) in the egress direction for DMVPN hub-to-spoke tunnels. The QoS policy on a DMVPN hub on a per-tunnel instance lets you shape tunnel traffic to individual spokes (a parent policy) and differentiate individual data flows going through the tunnel for policing (a child policy). The QoS policy that the hub uses for a specific spoke is selected according to the specific Next Hop Resolution Protocol (NHRP) group into which that spoke is configured. Although you can configure many spokes into the same NHRP group, the tunnel traffic for each spoke is measured individually for shaping and policing.

You can use this feature with DMVPN with or without Internet Protocol Security (IPsec).

When the Per-Tunnel QoS for DMVPN feature is enabled, queuing and shaping are performed at the outbound physical interface for generic routing encapsulation (GRE)/IPsec tunnel packets. The Per-Tunnel QoS for DMVPN feature ensures that the GRE header, the IPsec header, and the Layer 2 (for the physical interface) header are included in the packet-size calculations for shaping and bandwidth queuing of packets under QoS.

We configure 3 Policies 2MB / 3MB / 5MB for the Different Connection.

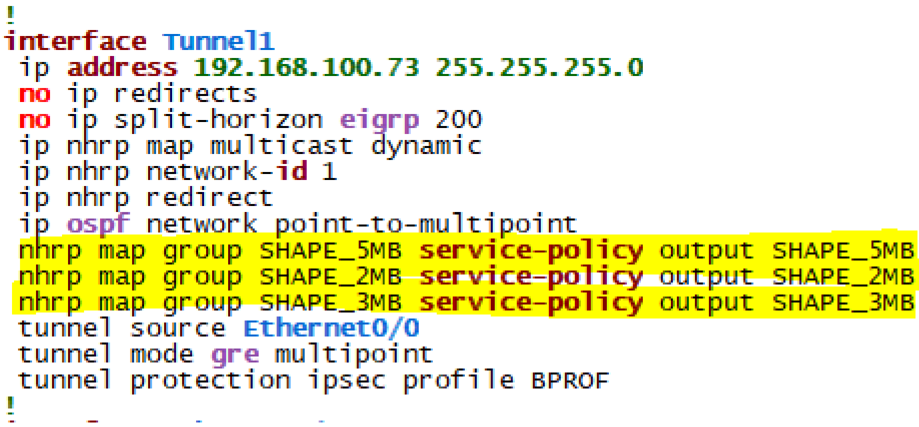
HUB we configure QOS on R73



```
class-map match-all critical
 match protocol http
 match protocol ftp
 match protocol rdp
class-map match-all control-plane
 match protocol telnet
 match protocol ssh
 match protocol tftp
class-map match-all VOICE
 match protocol rtp
class-map match-all defautl
 match any
class-map match-all video
 match protocol rtp video
!
policy-map SHAPE_5MB
 class class-default
  shape average 5000000
policy-map REMAINING_POLICY
 class VOICE
  priority percent 10
  set dscp ef
 class control-plane
  bandwidth remaining percent 5
  set dscp af41
 class video
  bandwidth remaining percent 25
  set dscp af41
 class critical
  bandwidth remaining percent 30
  set dscp af31
  fair-queue
 class defautl
  bandwidth remaining percent 30
  set dscp default
  queue-limit 400 packets
policy-map SHAPE_2MB
 class class-default
  shape average 2000000
   service-policy REMAINING_POLICY
policy-map SHAPE_3MB
 class class-default
  shape average 3000000
   service-policy REMAINING_POLICY
!
```

We apply now on Tunnel interface both HUB and SPOKE, we will test and verify.

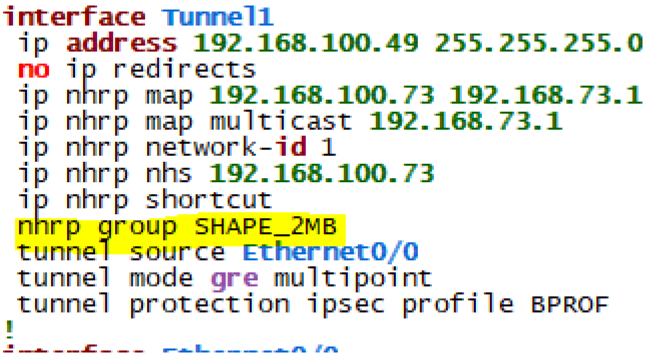
HUB R73



```
!
interface Tunnel1
 ip address 192.168.100.73 255.255.255.0
 no ip redirects
 no ip split-horizon eigrp 200
 ip nhrp map multicast dynamic
 ip nhrp network-id 1
 ip nhrp redirect
 ip ospf network point-to-multipoint
 nhrp map group SHAPE_5MB service-policy output SHAPE_5MB
 nhrp map group SHAPE_2MB service-policy output SHAPE_2MB
 nhrp map group SHAPE_3MB service-policy output SHAPE_3MB
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
 tunnel protection ipsec profile BPROF
!
```

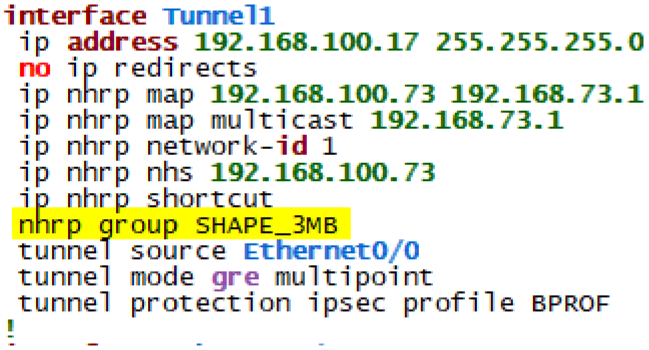
On SPOKE R49 and R17 apply and test

R49 for 2MB



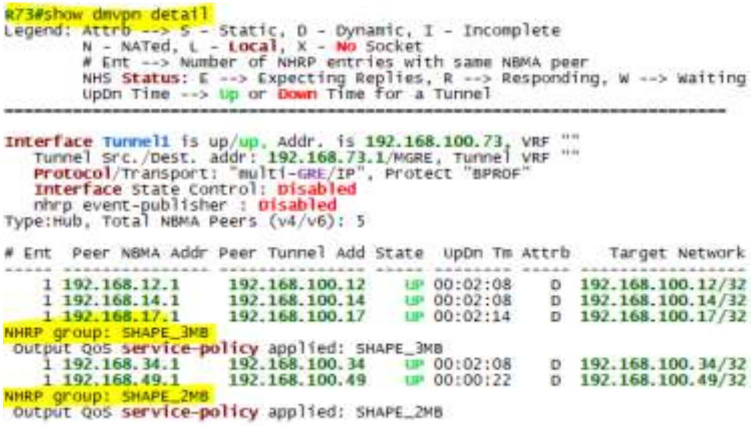
```
interface Tunnel1
 ip address 192.168.100.49 255.255.255.0
 no ip redirects
 ip nhrp map 192.168.100.73 192.168.73.1
 ip nhrp map multicast 192.168.73.1
 ip nhrp network-id 1
 ip nhrp nhs 192.168.100.73
 ip nhrp shortcut
 nhrp group SHAPE_2MB
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
 tunnel protection ipsec profile BPROF
!
interface Ethernet0/0
```

R17 for 3MB



```
interface Tunnel1
 ip address 192.168.100.17 255.255.255.0
 no ip redirects
 ip nhrp map 192.168.100.73 192.168.73.1
 ip nhrp map multicast 192.168.73.1
 ip nhrp network-id 1
 ip nhrp nhs 192.168.100.73
 ip nhrp shortcut
 nhrp group SHAPE_3MB
 tunnel source Ethernet0/0
 tunnel mode gre multipoint
 tunnel protection ipsec profile BPROF
!
```

Verify on Hub tag:



```
R73#show dmvpn detail
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface Tunnel1 is up/up, Addr. is 192.168.100.73, VRF ""
   Tunnel Src./Dest. addr: 192.168.73.1/MGRE, Tunnel VRF ""
   Protocol/Transport: "multi-GRE/IP", Protect "BPROF"
   Interface State Control: Disabled
   nhrp event-publisher : Disabled
Type:Hub, Total NBMA Peers (v4/v6): 5

# Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb    Target Network
----- --------------- --------------- ----- -------- ----- -----------------
    1 192.168.12.1     192.168.100.12    UP 00:02:08     D  192.168.100.12/32
    1 192.168.14.1     192.168.100.14    UP 00:02:08     D  192.168.100.14/32
    1 192.168.17.1     192.168.100.17    UP 00:02:14     D  192.168.100.17/32
NHRP group: SHAPE_3MB
 Output QoS service-policy applied: SHAPE_3MB
    1 192.168.34.1     192.168.100.34    UP 00:02:08     D  192.168.100.34/32
    1 192.168.49.1     192.168.100.49    UP 00:00:22     D  192.168.100.49/32
NHRP group: SHAPE_2MB
 Output QoS service-policy applied: SHAPE_2MB
```

3.2 Encryption

3.2.a Implement and troubleshoot IPsec with preshared key

3.2.a [i] IPv4 site to IPv4 site

3.2.a [ii] IPv6 in IPv4 tunnels
3.2.a [iii] Virtual tunneling interface [VTI]

3.3 Troubleshooting VPN technologies

3.3.a Use IOS troubleshooting tools
3.3.a [i] debug, conditional debug
3.3.a [ii] ping, traceroute with extended options
3.3.a [iii] Embedded packet capture

3.3.b Apply troubleshooting methodologies
3.3.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]
3.3.b [ii] Design and implement valid solutions according to constraints
3.3.b [iii] Verify and monitor resolution

3.3.c Interpret packet capture
3.3.c [i] Using wireshark trace analyzer
3.3.c [ii] Using IOS embedded packet capture

4.0 Infrastructure Security

4.1 Device security

4.1.a Implement and troubleshoot IOS AAA using local database
4.1.b Implement and troubleshoot device access control
4.1.b [i] Lines [VTY, AUX, console]
4.1.b [ii] SNMP
4.1.b [iii] Management plane protection
4.1.b [iv] Password encryption

4.1.c Implement and troubleshoot control plane policing

4.2 Network security

4.2.a Implement and troubleshoot switch security features
4.2.a [i] VACL, PACL
4.2.a [ii] Stormcontrol
4.2.a [iii] DHCP snooping
4.2.a [iv] IP source-guard
4.2.a [v] Dynamic ARP inspection
4.2.a [vi] Port-security
4.2.a [vii] Private VLAN

4.2.b Implement and troubleshoot router security features
4.2.b [i] IPv4 access control lists [standard, extended, time-based]

4.2.b [ii] IPv6 traffic filter
4.2.b [iii] Unicast reverse path forwarding

4.2.c Implement and troubleshoot IPv6 first hop security
4.2.c [i] RA guard
4.2.c [ii] DHCP guard
4.2.c [iii] Binding table
4.2.c [iv] Device tracking
4.2.c [v] ND inspection/snooping
4.2.c [vi] Source guard
4.2.c [vii] PACL

4.3 Troubleshooting infrastructure security

4.3.a Use IOS troubleshooting tools
4.3.a [i] debug, conditional debug
4.3.a [ii] ping, traceroute with extended options
4.3.a [iii] Embedded packet capture
4.3.b Apply troubleshooting methodologies
4.3.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]
4.3.b [ii] Design and implement valid solutions according to constraints
4.3.b [iii] Verify and monitor resolution

4.3.c Interpret packet capture
4.3.c [i] Using wireshark trace analyzer
4.3.c [ii] Using IOS embedded packet capture

5.0 Infrastructure Services

5.1 System management

5.1.a Implement and troubleshoot device management
5.1.a [i] Console and VTY
5.1.a [ii] telnet, HTTP, HTTPS, SSH, SCP
5.1.a [iii] [T]FTP

5.1.b Implement and troubleshoot SNMP
5.1.b [i] v2c, v3

5.1.c Implement and troubleshoot logging
5.1.c [i] Local logging, syslog, debug, conditional debug
5.1.c [ii] Timestamp

5.2 Quality of service

5.2.a Implement and troubleshoot end to end QoS
5.2.a [i] CoS and DSCP mapping

5.2.b Implement, optimize and troubleshoot QoS using MQC
5.2.b [i] Classification
5.2.b [ii] Network based application recognition [NBAR]
5.2.b [iii] Marking using IP precedence, DSCP, CoS, ECN
5.2.b [iv] Policing, shaping
5.2.b [v] Congestion management [queuing]
5.2.b [vi] HQoS, sub-rate ethernet link
5.2.b [vii] Congestion avoidance [WRED]

5.3 Network services

5.3.a Implement and troubleshoot first-hop redundancy protocols
5.3.a [i] HSRP, GLBP, VRRP
5.3.a [ii] Redundancy using IPv6 RS/RA

5.3.b Implement and troubleshoot network time protocol
5.3.b [i] NTP master, client, version 3, version 4
5.3.b [ii] NTP authentication

5.3.c Implement and troubleshoot IPv4 and IPv6 DHCP
5.3.c [i] DHCP client, IOS DHCP server, DHCP relay
5.3.c [ii] DHCP options
5.3.c [iii] DHCP protocol operations
5.3.c [iv] SLAAC/DHCPv6 interaction
5.3.c [v] Stateful, stateless DHCPv6
5.3.c [vi] DHCPv6 prefix delegation

5.3.d Implement and troubleshoot IPv4 network address translation
5.3.d [i] Static NAT, dynamic NAT, policy-based NAT, PAT
5.3.d [ii] NAT ALG

5.4 Network optimization

5.4.a Implement and troubleshoot IP SLA
5.4.a [i] ICMP, UDP, jitter, VoIP

5.4.b Implement and troubleshoot tracking object
5.4.b [i] Tracking object, tracking list
5.4.b [ii] Tracking different entities [e.g. interfaces, routes, IPSLA, and such]

5.4.c Implement and troubleshoot netflow

5.4.c [i] Netflow v5, v9
5.4.c [ii] Local retrieval
5.4.c [iii] Export [configuration only]
5.4.d Implement and troubleshoot embedded event manager
5.4.d [i] EEM policy using applet

5.5 Troubleshooting infrastructure services

5.5.a Use IOS troubleshooting tools
5.5.a [i] debug, conditional debug
5.5.a [ii] ping, traceroute with extended options
5.5.a [iii] Embedded packet capture

5.5.b Apply troubleshooting methodologies
5.5.b [i] Diagnose the root cause of networking issue [analyze symptoms, identify and describe root cause]
5.5.b [ii] Design and implement valid solutions according to constraints
5.5.b [iii] Verify and monitor resolution

5.5.c Interpret packet capture
5.5.c [i] Using wireshark trace analyzer
5.5.c [ii] Using IOS embedded packet capture