Configuring BGP

Yannis Xydas

Planning to Deploy BGP

- Prior to deploying a BGP routing solution, the following should be considered:
  - IP addressing plan
  - Network topology
  - BGP relationship with service provider(s)
- Once the requirements have been assessed, the implementation plan can be created.

Implementing Basic BGP

- The information necessary to implement BGP routing includes the following:
  - The AS numbers of enterprise and service provider.
  - The IP addresses of all the neighbors (peers) involved.
  - The networks that are to be advertised into BGP
- In the implementation plan, basic BGP tasks include the following:
  - Define the BGP process
  - Establish the neighbor relationships
  - Advertise the networks into BGP

Verifying BGP

- After implementing BGP, verification should confirm proper deployment on each router.
- Verification tasks include verifying:
  - That the appropriate BGP neighbor relationships and adjacencies are established.
  - That the BGP table is populated with the necessary information.
  - That IP routing table is populated with the necessary information.
  - That there is connectivity in the network between routers and to other devices.
  - That BGP behaves as expected in a case of a topology change, by testing link failure and router failure events.

Documenting

- After a successful BGP deployment, the solution and verification process and results should be documented for future reference.
- Documentation should include:
  - A topology map
  - The IP addressing plan
  - The autonomous system hierarchy
  - The networks and interfaces included in BGP on each router
  - The default and any special metrics configured
  - The verification results.

Enable BGP Routing

- Define BGP as the IP routing protocol.

```
Router(config)#
router bgp autonomous-system
```

- The *autonomous-system* value is either an internally generated number (if not connecting to a provider network) or obtained from an ISP or RIR.
  - It is a required parameter.
  - It can be any positive integer in the range from 1 to 65535.
- Only one instance of BGP can be configured on the router at a single time.

4-byte AS numbers

- On Cisco routers there are two formats used to configure a 4-byte AS number:
  - **asplain**: The Cisco implementation.
  - **asdot**: The RFC 5396 implementation.
    - Use the **`bgp asnotation dot`** command to configure.
    - AS numbers must be written using the asdot format, or the regular expression match will fail.
- **Note:** The 4-byte AS number will not be used in this chapter; therefore, all examples use the 2-byte AS numbering format.

Defining BGP Neighbors

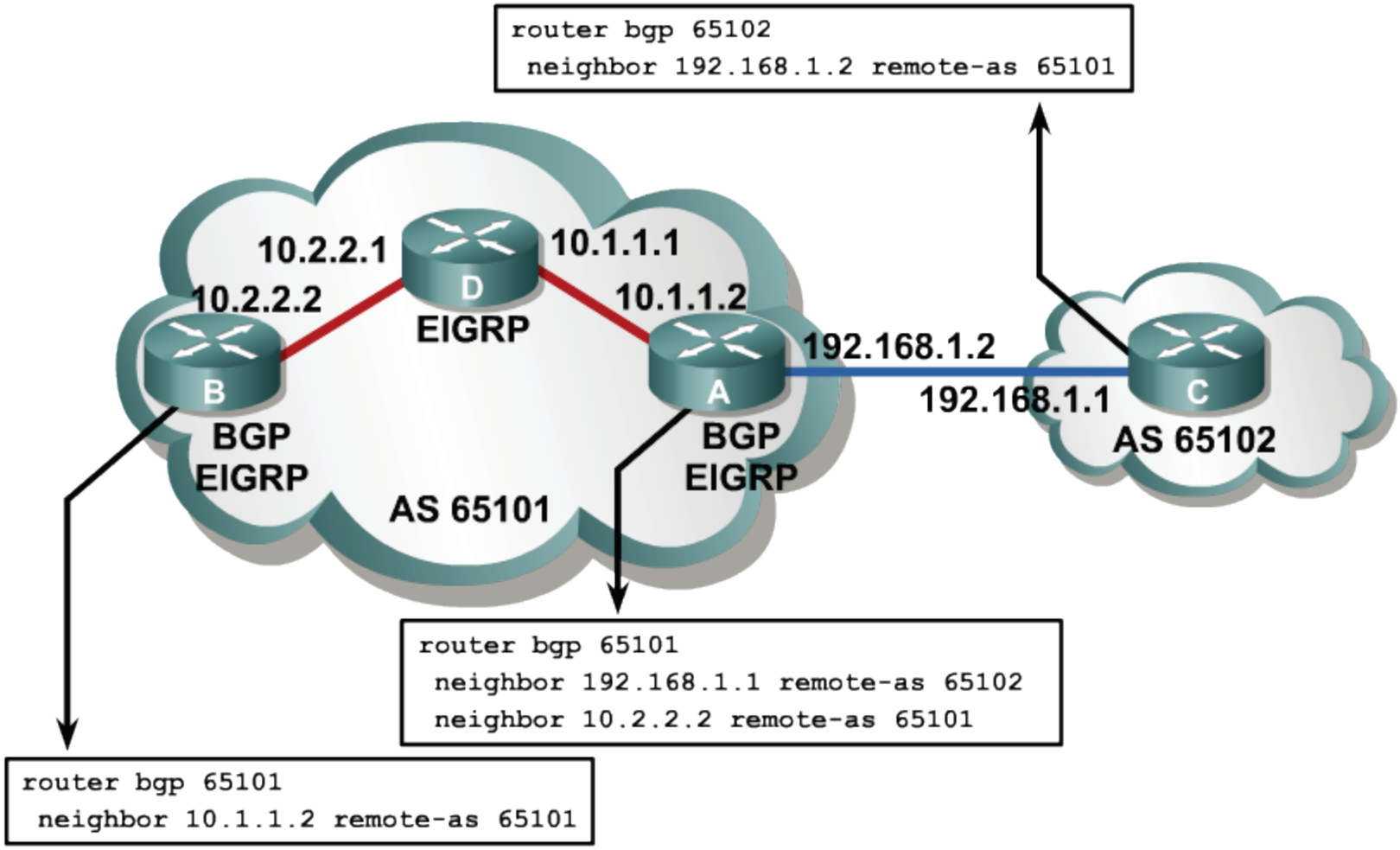
- Identify peer router with which to establish a BGP session.

```
Router(config-router)#
```

```
neighbor {ip-address | peer-group-name} remote-as
  autonomous-system
```

- The *ip-address* is the destination address of the BGP peer.
  - The address must be reachable before attempting to establish the BGP relationship.
- The *autonomous-system* value is used to identify if the session is with internal BGP (IBGP) peers or with external BGP (EBGP) peers.
  - If the value is the same as the router's AS, then an IBGP session is attempted.
  - If the value is not the same as the router's AS, then an EBGP session is attempted.

Example: BGP `neighbor` Command



BGP Peer Groups

- In BGP, neighbors are often configured with the same update policies.
- To simplify configuration and make updating more efficient, neighbors with the same update policies can be grouped into **peer groups**.
  - Recommended approach when there are many BGP peers.
- Instead of separately defining the same policies for each neighbor, a peer group can be defined with these policies assigned to the peer group.
  - Individual neighbors are then made members of the peer group.
  - Members of the peer group inherit all the peer group's configuration options.
  - Only options that affect the inbound updates can be overridden.

Defining a BGP Peer Group

- Create a peer group on the local router.

```
Router(config-router)#
```

```
neighbor peer-group-name peer-group
```

- The *peer-group-name* is the name of the BGP peer group to be created.
- The name is local to the router on which it is configured and is not passed to any other router.

Assign Neighbors to the Peer Group

- Assign neighbors as part of the peer group.

```
Router(config-router)#
```

neighbor *ip-address* **peer-group** *peer-group-name*

- The *ip-address* is the IP address of the neighbor that is to be assigned as a member of the peer group.
- The *peer-group-name* must already exist.
  - Note: The **clear ip bgp peer-group** *peer-group-name* EXEC command can be used to reset the BGP connections for all members of a peer group.

Shut Down a BGP Neighbor

- To disable an existing BGP neighbor or peer group relationship.

```
Router(config-router)#
```

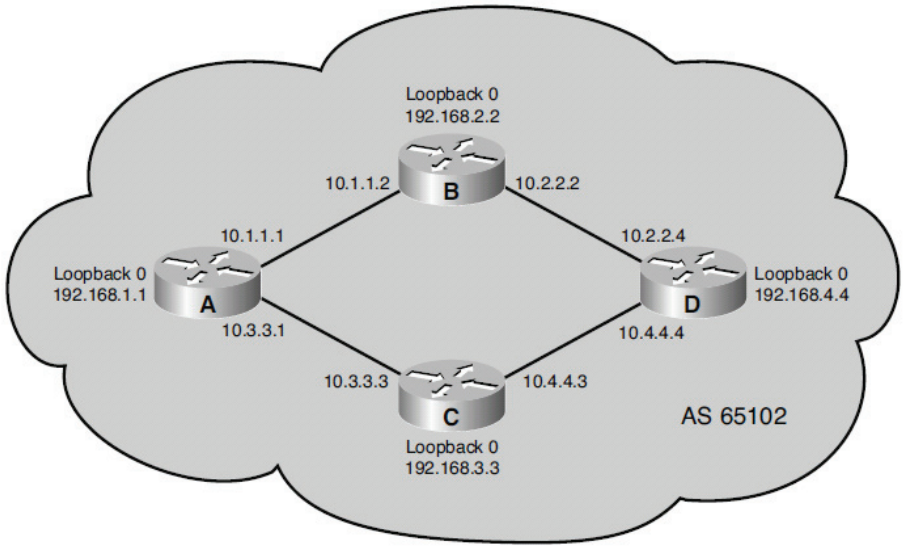
neighbor {*ip-address* | *peer-group-name*} **shutdown**

- Useful when making major policy changes to a neighboring router.
- The command not only terminates the session, but also removes all associated routing information.
- To re-enable the neighbor prepend the **no** keyword to the command.

IBGP Source IP Address Problem

- BGP does not accept unsolicited updates.
  - It must be aware of every neighboring router and have a **`neighbor`** statement for it.
- For example, when a router creates and forwards a packet, the IP address of the outbound interface is used as that packet's source address by default.
  - For BGP packets, this source IP address must match the address in the corresponding **`neighbor`** statement on the other router or the routers will not establish the BGP session.
  - This is not a problem for EBGP neighbors as they are typically directly connected.

IBGP Source IP Address Problem



- When multiple paths exist between IBGP neighbors, the BGP source address can cause problems:
  - Router D uses the **`neighbor 10.3.3.1 remote-as 65102`** command to establish a relationship with A.
  - However, router A is sending BGP packets to D via B therefore the source IP address of the packets is 10.1.1.1.
  - The IBGP session between A and D cannot be established because D does not recognize 10.1.1.1 as a BGP neighbor.

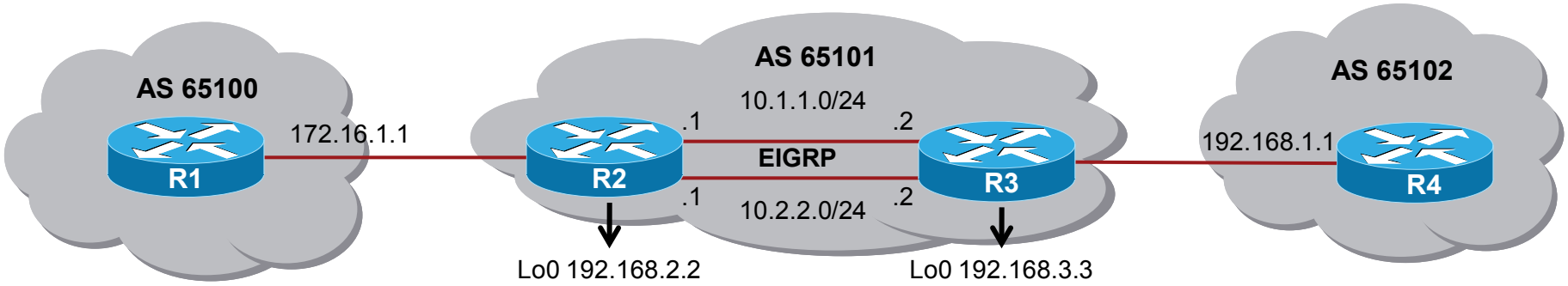
IBGP Source IP Address Solution

- Establish the IBGP session using a loopback interface.

```
Router(config-router)#
neighbor {ip-address | peer-group-name} update-source
  loopback interface-number
```

- Informs the router to use a loopback interface address for all BGP packets.
- Overrides the default source IP address for BGP packets.
- Typically only used with IBGP sessions.
- As an added bonus, physical interfaces can go down for any number of reasons but loopbacks never fail.

IBGP Source IP Address Example




```
R2(config)# router bgp 65101
R2(config-router)# neighbor 172.16.1.1 remote-as 65100
R2(config-router)# neighbor 192.168.3.3 remote-as 65101
R2(config-router)# neighbor 192.168.3.3 update-source loopback0
R2(config-router)# exit
R2(config)# router eigrp 1
R2(config-router)# network 10.0.0.0
R2(config-router)# network 192.168.2.0
R2(config-router)#
```

```
R3(config)# router bgp 65101
R3(config-router)# neighbor 192.168.1.1 remote-as 65102
R3(config-router)# neighbor 192.168.2.2 remote-as 65101
R3(config-router)# neighbor 192.168.2.2 update-source loopback0
R3(config-router)# exit
R3(config)# router eigrp 1
R3(config-router)# network 10.0.0.0
R3(config-router)# network 192.168.3.0
R3(config-router)#
```

EBGP Dual-Homed Problem

- R1 in AS 65102 is dual-homed with R2 in AS 65101.
- A problem can occur if R1 only uses a single **`neighbor`** statement pointing to 192.168.1.18 on R2 .
  - If that link fails, the BGP session between these AS is lost, and no packets pass from one autonomous system to the next, even though another link exists.
- A solution is configuring two **`neighbor`** statements on R1 pointing to 192.168.1.18 and 192.168.1.34.
  - However, this doubles the BGP updates from R1 to R2.

EBGP Dual-Homed Solution

- The ideal solution is to:
  - Use loopback addresses.
  - Configure static routes to reach the loopback address of the other router.
  - Configure the **`neighbor ebgp-multihop`** command to inform the BGP process that this neighbor is more than one hop away.

Enable Multihop EBGP

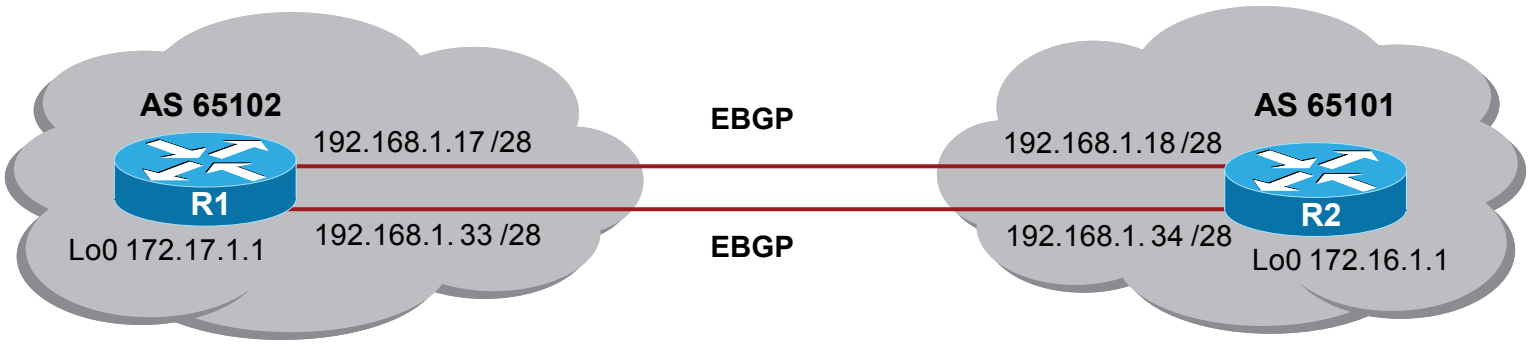
- Increase the time-to-live (TTL) for EBGP connections.

```
Router(config-router)#
```

neighbor {*ip-address* | *peer-group-name*} **ebgp-multihop** [*ttl*]

- This command is of value when redundant paths exist between EBGP neighbors.
- The default *ttl* is 1, therefore BGP peers must be directly connected.
  - The range is from 1 to 255 hops.
- Increasing the *ttl* enables BGP to establish EBGP connections beyond one hop and also enables BGP to perform load balancing.

Multihop EBGP Example




```
R1(config)# router bgp 65102
R1(config-router)# neighbor 172.16.1.1 remote-as 65101
R1(config-router)# neighbor 172.16.1.1 update-source loopback0
R1(config-router)# neighbor 172.16.1.1 ebgp-multihop 2
R1(config-router)#  exit
R1(config)# ip route 172.16.1.1 255.255.255.255 192.168.1.18
R1(config)# ip route 172.16.1.1 255.255.255.255 192.168.1.34
R1(config)#
```

```
R2(config)# router bgp 65101
R2(config-router)# neighbor 172.17.1.1 remote-as 65102
R2(config-router)# neighbor 172.17.1.1 update-source loopback0
R2(config-router)# neighbor 172.17.1.1 ebgp-multihop 2
R2(config-router)#  exit
R2(config)# ip route 172.17.1.1 255.255.255.255 192.168.1.17
R2(config)# ip route 172.17.1.1 255.255.255.255 192.168.1.33
R2(config)#
```

Advertising EBGP Routes to IBGP Peers

- When an EBGP router receives an update from an EBGP neighbor and forwards the update to its IBGP peers, the source IP address will still be that of the EBGP router.
  - IBGP neighbors will have to be configured to reach that external IP address.
- Another solution is to override a router's default behavior and force it to advertise itself as the next-hop address for routes sent to a neighbor.
  - To do so, use the **`neighbor next-hop-self`** router configuration command

`neighbor next-hop-self` Command

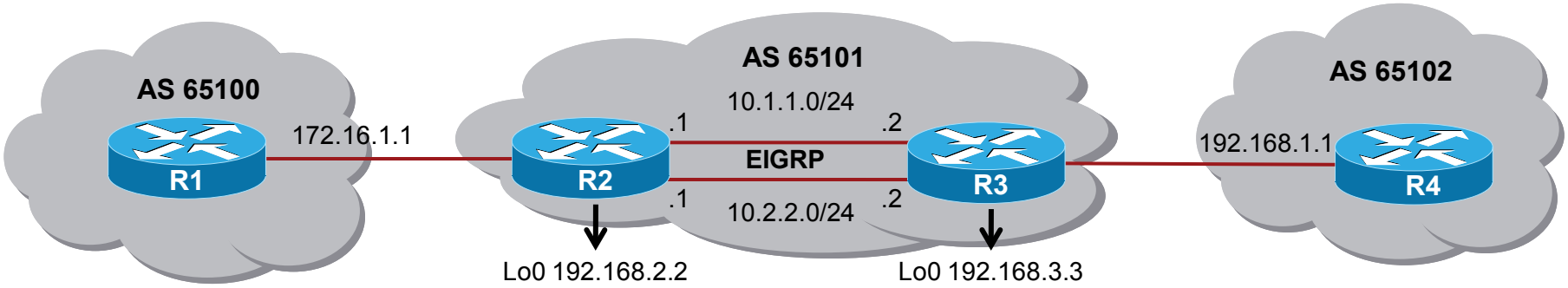
- Configure the router as the next hop for a BGP-speaking peer.

```
Router(config-router)#
```

`neighbor {`*`ip-address`*** `|` ***`peer-group-name`***`} next-hop-self`**

- The command forces BGP to advertise itself as the source of the routes.
- The *`ip-address`* identifies the peer router to which advertisements will be sent, with this router identified as the next hop.
- This command is useful in unmeshed networks (such as Frame Relay) where BGP neighbors may not have direct access to all other neighbors on the same IP subnet.

Next Hop Self Example




```
R2(config)# router bgp 65101
R2(config-router)# neighbor 172.16.1.1 remote-as 65100
R2(config-router)# neighbor 192.168.3.3 remote-as 65101
R2(config-router)# neighbor 192.168.3.3 update-source loopback0
R2(config-router)# neighbor 192.168.3.3 next-hop-self
R2(config-router)# exit
R2(config)# router eigrp 1
R2(config-router)# network 10.0.0.0
R2(config-router)# network 192.168.2.0
R2(config-router)#
```

BGP Synchronization

- Recall that the BGP synchronization rule states that:
  - "*A BGP router should not use, or advertise a route learned by IBGP, unless that route is local or is learned from the IGP*."
- By default synchronization is disabled, therefore BGP can use and advertise to an external BGP neighbor routes learned from an IBGP neighbor that are not present in the local routing table.
  - Use the **`synchronization`** router configuration command to enable BGP synchronization so that a router will not advertise routes in BGP until it learns them in an IGP.
  - The **`no synchronization`** router configuration command disables synchronization.

Defining Networks That BGP Advertises

- Two options are available to advertise networks into BGP:
  - The **`network`** command.
  - Redistributing IGP routes into BGP.
- Note: Redistributing is not recommended because it could result in unstable BGP tables.

Identify BGP Networks

- Enable BGP to advertise a network if it is present.

```
Router(config-router)#
```

network *network-number* **[mask** *network-mask***] [route-map** *map-tag***]**

- The BGP **network** command determines which networks this router advertises.
  - Unlike IGPs, the command does not start BGP on specific interfaces.
- The **mask** parameter indicates that BGP-4 supports subnetting and supernetting.
  - If the mask is not specified, this command announces only the classful network
- It is also important to note that the prefix must exactly match (address and mask) an entry in the IP routing table.

BGP Route Must Be in IP Routing Table

- It is important to understand that any network (both address and mask) must exist in the routing table for the network to be advertised in BGP.
- For example, to summarize many networks and advertise a CIDR block 192.168.0.0/16, configure:

```
network 192.168.0.0 mask 255.255.0.0
ip route 192.168.0.0 255.255.0.0 null0
```

- Now BGP can find an exact match in the routing table and announce the 192.168.0.0/16 network to its neighbors.
  - The advertised static route would never actually be used since BGP would contain longer prefix matching routes in its routing table.

Clearing the BGP Session

- When policies such as access lists or attributes are changed, the Cisco IOS applies changes on only those updates received or sent *after* and not existing routes in the BGP and routing tables.
  - It can take a long time for the policy to be applied to all networks.
- There are three ways to ensure that the policy change is immediately applied to all affected prefixes and paths.
  - Hard reset
  - Soft reset (outbound and inbound)
  - Route refresh

Hard Reset of BGP Sessions

- Reset all BGP connections with this router.

```
Router#
clear ip bgp {* | neighbor-address}
```

- Entire BGP forwarding table is discarded.
- BGP session makes the transition from established to idle; everything must be relearned.
- When the *neighbor-address* value is used, it resets only a single neighbor and BGP session. Everything from this neighbor must be relearned.
  - It is less severe than **clear ip bgp ***.

Soft Reset Outbound

- Resets all BGP connections without loss of routes.

```
Router#
clear ip bgp {* | neighbor-address} [soft out]
```

- The connection remains established and the command does not reset the BGP session.
  - Instead the router creates a new update and sends the whole table to the specified neighbors.
- This update includes withdrawal commands for networks that the neighbor will not see anymore based on the new outbound policy.
- This option is highly recommended when you are changing outbound policy.

Soft Reset Inbound: Method #1

- Two commands are required.

```
Router(config-router)#
```

```
neighbor {ip-address} soft-reconfiguration inbound
```

- Use this command when changes need to be made without forcing the other side to resend everything.
- It causes the BGP router to retain an unfiltered table of what a neighbor had sent but can be memory intensive.

```
Router#
```

```
clear ip bgp {* | neighbor-address} [soft in]
```

- Causes the router to use the stored unfiltered table to generate new inbound updates and the new results are placed in the BGP forwarding database.

Soft Reset Inbound: Method #2

- Also called route refresh.

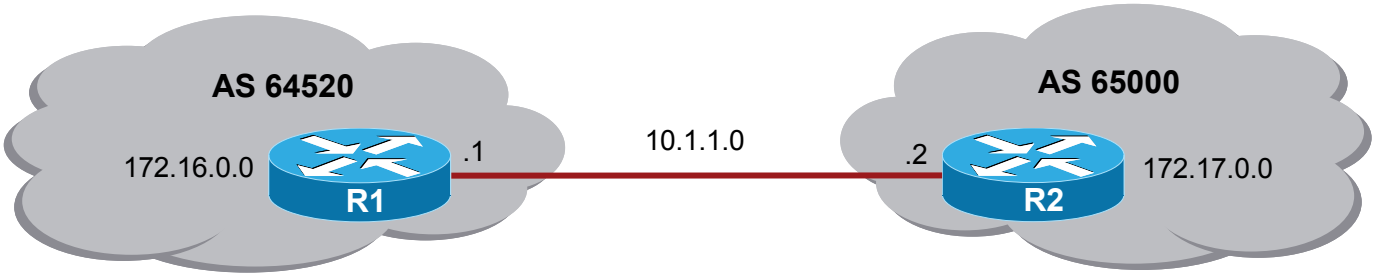
```
Router#
clear ip bgp {* | neighbor-address} [soft in | in]
```

- This dynamically soft resets inbound updates.
- Unlike method #1, this method requires no preconfiguration and requires significantly less memory.

Monitoring Received BGP Routes

Command	Description
show ip bgp neighbors {*address*} **received-routes**	Displays all received routes (both accepted and rejected) from the specified neighbor.
show ip bgp neighbors {*address*} **routes**	Displays all routes that are received and accepted from the specified neighbor. This output is a subset of the output displayed by the **received-routes** keyword.
show ip bgp	Displays entries in the BGP table.
show ip bgp neighbors {*address*} **advertised-routes**	Displays all BGP routes that have been advertised to neighbors.

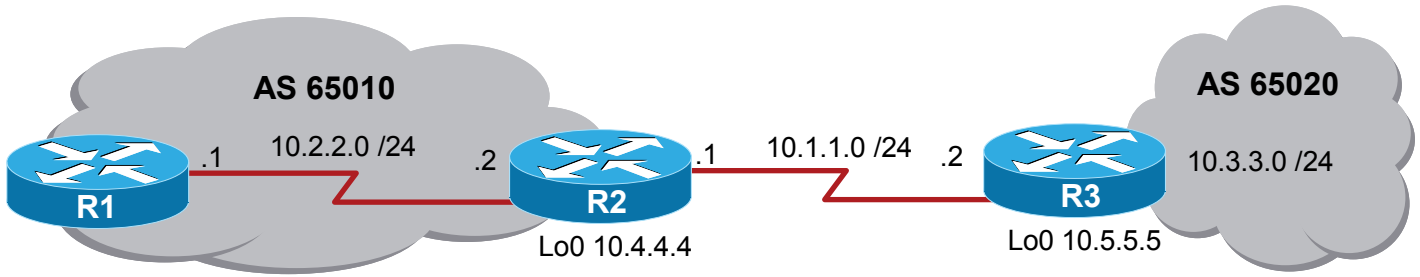
BGP Configuration Example #1




```
R1(config)# router bgp 64520
R1(config-router)# neighbor 10.1.1.2 remote-as 65000
R1(config-router)# network 172.16.0.0
R1(config-router)#
```

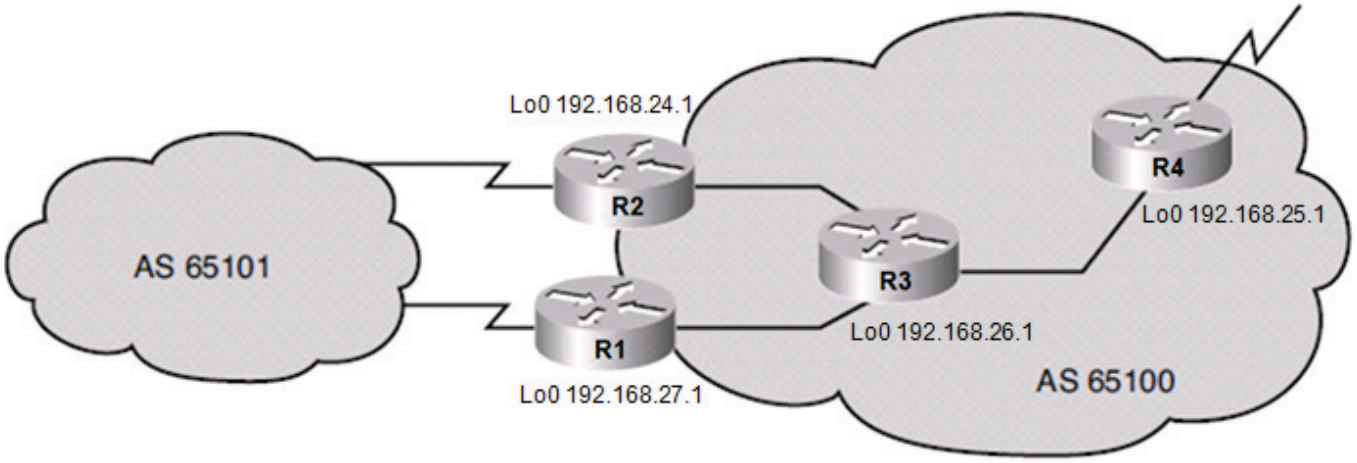
```
R2(config)# router bgp 65000
R2(config-router)# neighbor 10.1.1.1 remote-as 64520
R2(config-router)# network 172.17.0.0
R2(config-router)#
```

BGP Configuration Example #2



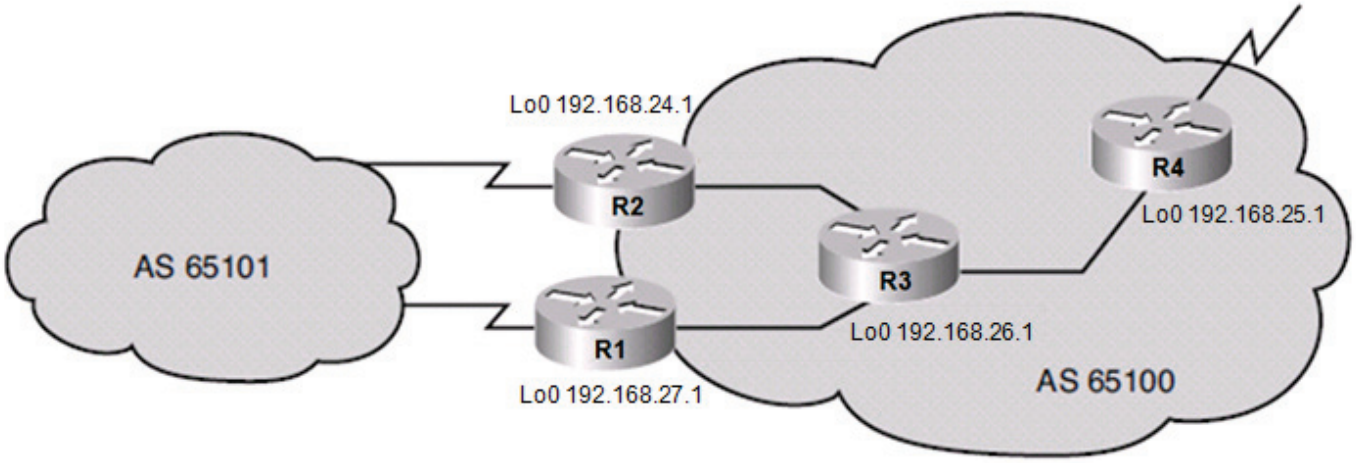

```
R2(config)# router bgp 65010
R2(config-router)# neighbor 10.1.1.2 remote-as 65020
R2(config-router)# network 10.2.2.0 mask 255.255.255.0
R2(config-router)# network 10.4.4.0 mask 255.255.255.0
R2(config-router)#
```

BGP Without Peer Group Example



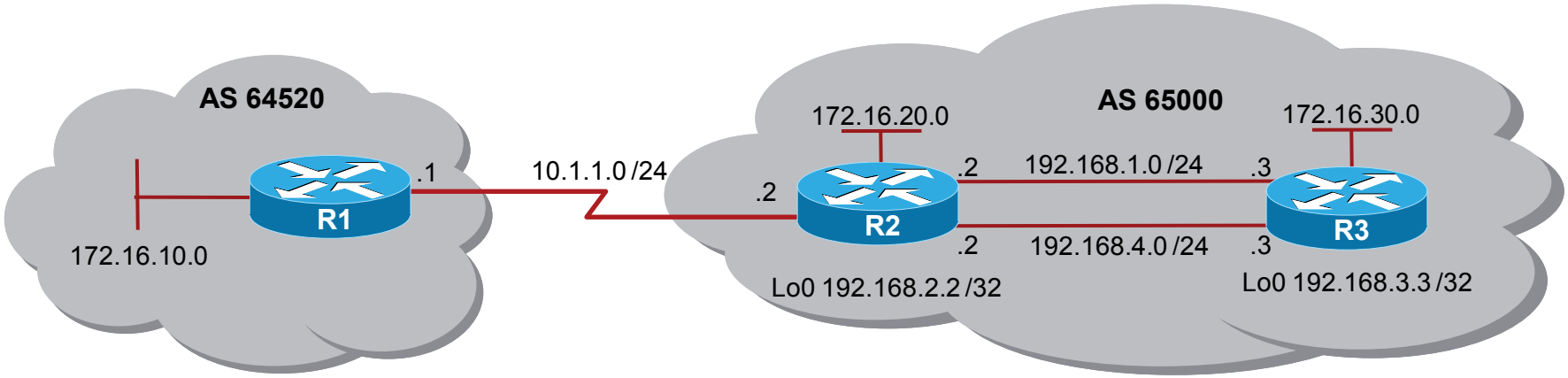

```
R1(config)# router bgp 65100
R1(config-router)# neighbor 192.168.24.1 remote-as 65100
R1(config-router)# neighbor 192.168.24.1 update-source loopback 0
R1(config-router)# neighbor 192.168.24.1 next-hop-self
R1(config-router)# neighbor 192.168.24.1 distribute-list 20 out
R1(config-router)#
R1(config-router)# neighbor 192.168.25.1 remote-as 65100
R1(config-router)# neighbor 192.168.25.1 update-source loopback 0
R1(config-router)# neighbor 192.168.25.1 next-hop-self
R1(config-router)# neighbor 192.168.25.1 distribute-list 20 out
R1(config-router)#
R1(config-router)# neighbor 192.168.26.1 remote-as 65100
R1(config-router)# neighbor 192.168.26.1 update-source loopback 0
R1(config-router)# neighbor 192.168.26.1 next-hop-self
R1(config-router)# neighbor 192.168.26.1 distribute-list 20 out
R1(config-router)#
```

BGP With Peer Group Example




```
R1(config)# router bgp 65100
R1(config-router)# neighbor INTERNAL peer-group
R1(config-router)# neighbor INTERNAL remote-as 65100
R1(config-router)# neighbor INTERNAL update-source loopback 0
R1(config-router)# neighbor INTERNAL next-hop-self
R1(config-router)# neighbor INTERNAL distribute-list 20 out
R1(config-router)# neighbor 192.168.24.1 peer-group INTERNAL
R1(config-router)# neighbor 192.168.25.1 peer-group INTERNAL
R1(config-router)# neighbor 192.168.26.1 peer-group INTERNAL
R1(config-router)#
```

IBGP and EBGP Example




```
R2(config)# router bgp 65000
R2(config-router)# neighbor 10.1.1.1 remote-as 64520
R2(config-router)# neighbor 192.168.3.3 remote-as 65000
R2(config-router)# neighbor 192.168.3.3 update-source loopback 0
R2(config-router)# neighbor 192.168.3.3 next-hop-self
R2(config-router)# network 172.16.20.0 mask 255.255.255.0
R2(config-router)# network 192.168.1.0
R2(config-router)# network 192.168.4.0
R2(config-router)# no synchronization
R2(config-router)#
```

Verifying and Troubleshooting BGP

Verifying and Troubleshooting BGP

Command	Description
show ip bgp	Displays entries in the BGP table. Specify a network number to get more specific information about a particular network.
show ip bgp neighbors	Displays detailed information about the TCP and BGP connections to neighbors.
show ip bgp summary	Displays the status of all BGP connections.
show ip bgp neighbors {*address*} advertised-routes	Displays all BGP routes that have been advertised to neighbors.
show ip bgp rib-failure	Displays BGP routes that were not installed in the routing information base (RIB), and the reason that they were not installed.
debug ip bgp [dampening | events | keepalives | updates]	

Verifying BGP: `show ip bgp`

Display the BGP topology database (the BGP table).

The status codes are shown in the first column of each line of output.

- * means that the next-hop address (in the fifth column) is valid.

- **r** means a RIB failure and the route was not installed in the RIB.

A **>** in the second column indicates the best path for a route selected by BGP.

This route is offered to the IP routing table.

The third column is either blank or has an "**i**" in it.

- If it has an **i**, an IBGP neighbor advertised this route to this router.

- If it is blank, BGP learned that route from an external peer.

```
R1# show ip bgp
BGP table version is 14, local router ID is 172.31.11.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal, r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
     Network          Next Hop            Metric LocPrf Weight Path
*>  10.1.0.0/24      0.0.0.0                  0         32768 i
*  i                 10.1.0.2                 0    100      0 i
*>  10.1.1.0/24      0.0.0.0                  0         32768 i
*>i10.1.2.0/24       10.1.0.2                 0    100      0 i
*>  10.97.97.0/24    172.31.1.3                             0 64998 64997 i
*                    172.31.11.4                            0 64999 64997 i
*  i                 172.31.11.4              0    100      0 64999 64997 i
*>  10.254.0.0/24    172.31.1.3               0             0 64998 i
*                    172.31.11.4                            0 64999 64998 i
*  i                 172.31.1.3               0    100      0 64998 i
r>  172.31.1.0/24    172.31.1.3               0             0 64998 i
r                    172.31.11.4                            0 64999 64998 i
r  i                 172.31.1.3               0    100      0 64998 i
*>  172.31.2.0/24    172.31.1.3               0             0 64998 i
```

This section lists three BGP path attributes: metric (MED), local preference, and weight.

The Path section lists the AS path. The last AS # is the originating AS.

If blank the route is from the current autonomous system.

The last column displays the ORIGIN attribute).

- **i** means the original router probably used a **`network`** command to introduce this network into BGP.

- **?** means the route was probably redistributed from an IGP into the BGP process.

Verifying BGP: `show ip bgp rib-failure`

- Displays BGP routes that were not installed in the RIB and the reason that they were not installed.
- In this example, the displayed routes were not installed because a route or routes with a better administrative distance already existed in the RIB.

```
R1# show ip bgp rib-failure
Network Next Hop RIB-failure RIB-NH Matches
172.31.1.0/24 172.31.1.3 Higher admin distance n/a
172.31.11.0/24 172.31.11.4 Higher admin distance n/a
```

Verifying BGP: `show ip bgp summary`

Verify the BGP neighbor relationship.

```
R1# show ip bgp summary
BGP router identifier 10.1.1.1, local AS number 65001
BGP table version is 124, main routing table version 124
9 network entries using 1053 bytes of memory
22 path entries using 1144 bytes of memory
12/5 BGP path/bestpath attribute entries using 1488 bytes of memory
6 BGP AS-PATH entries using 144 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 3829 total bytes of memory
BGP activity 58/49 prefixes, 72/50 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd

10.1.0.2        4 65001      11      11      124    0    0 00:02:28        8
172.31.1.3      4 64998      21      18      124    0    0 00:01:13        6
172.31.11.4     4 64999      11      10      124    0    0 00:01:11        6
```

Verifying BGP: debug ip bgp updates

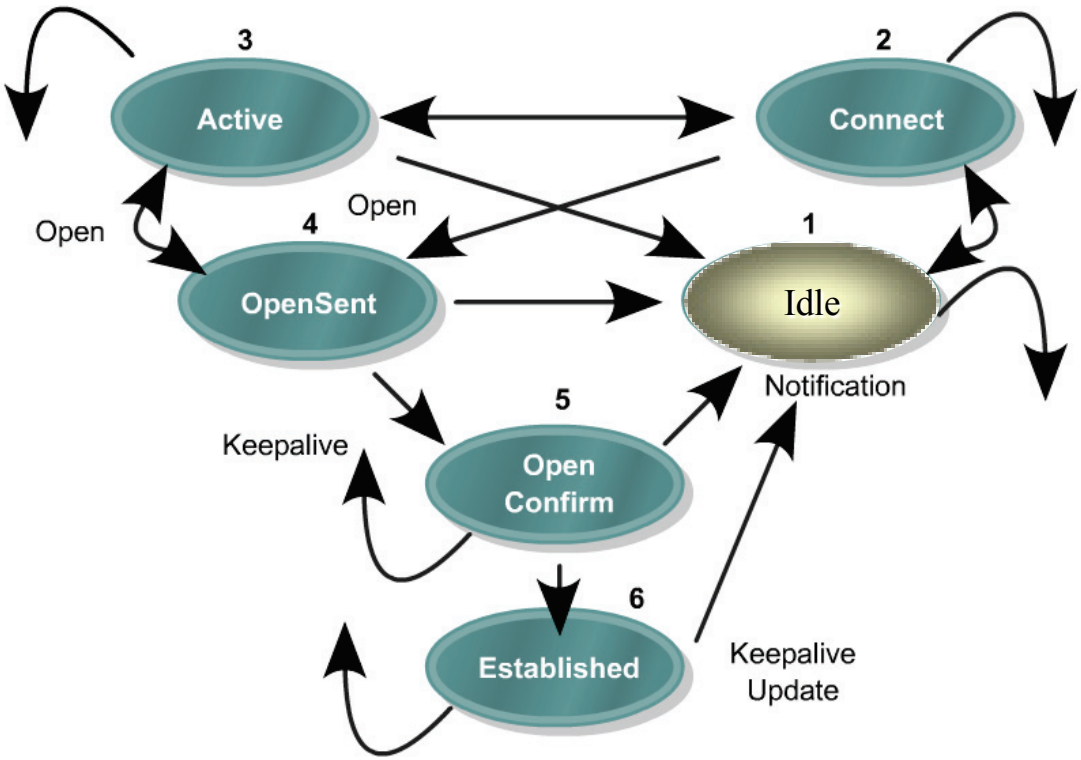
Verify the BGP neighbor relationship.

```
R1# debug ip bgp updates
Mobile router debugging is on for address family: IPv4 Unicast
R1# clear ip bgp 10.1.0.2
<output omitted>
*May 24 11:06:41.309: %BGP-5-ADJCHANGE: neighbor 10.1.0.2 Up
*May 24 11:06:41.309: BGP(0): 10.1.0.2 send UPDATE (format) 10.1.1.0/24, next 10.1.0.1, metric 0,
path Local
*May 24 11:06:41.309: BGP(0): 10.1.0.2 send UPDATE (prepend, chgflags: 0x0) 10.1.0.0/24, next
10.1.0.1, metric 0, path Local
*May 24 11:06:41.309: BGP(0): 10.1.0.2 NEXT_HOP part 1 net 10.97.97.0/24, next 172.31.11.4
*May 24 11:06:41.309: BGP(0): 10.1.0.2 send UPDATE (format) 10.97.97.0/24, next 172.31.11.4, metric
0, path 64999 64997
*May 24 11:06:41.309: BGP(0): 10.1.0.2 NEXT_HOP part 1 net 172.31.22.0/24, next 172.31.11.4
*May 24 11:06:41.309: BGP(0): 10.1.0.2 send UPDATE (format) 172.31.22.0/24, next 172.31.11.4,
metric 0, path 64999
<output omitted>
*May 24 11:06:41.349: BGP(0): 10.1.0.2 rcvd UPDATE w/ attr: nexthop 10.1.0.2, origin i, localpref
100, metric 0
*May 24 11:06:41.349: BGP(0): 10.1.0.2 rcvd 10.1.2.0/24
*May 24 11:06:41.349: BGP(0): 10.1.0.2 rcvd 10.1.0.0/24
```

BGP States

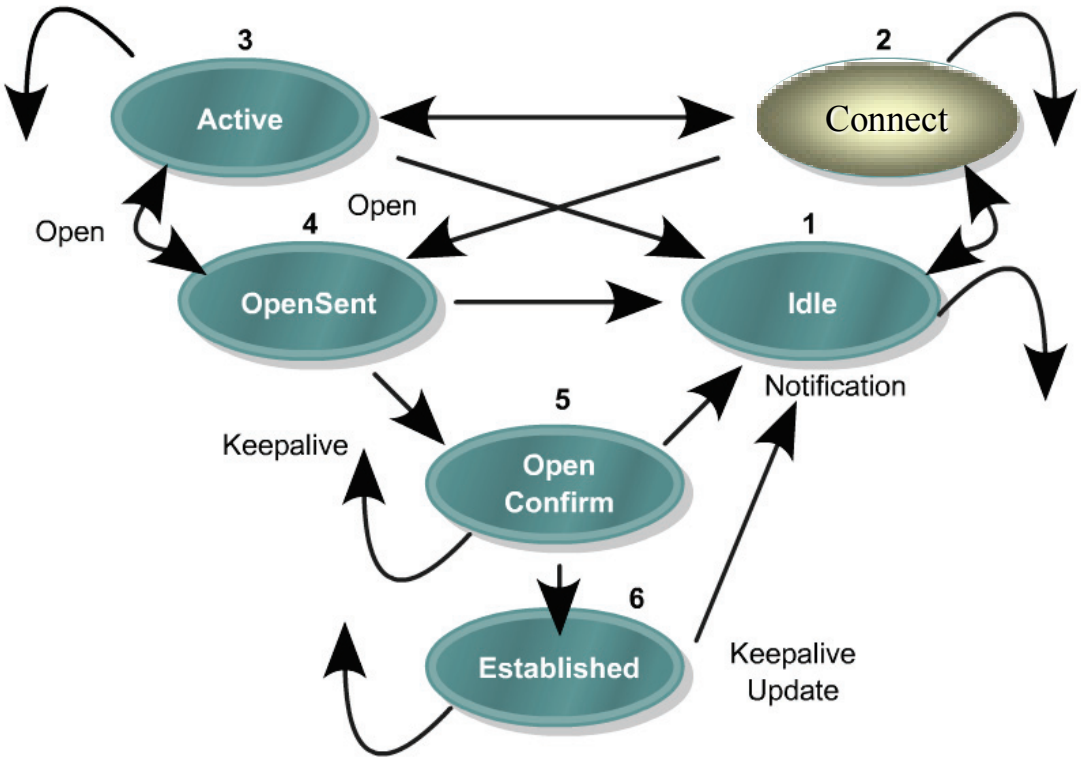
- BGP is a state machine that takes a router through the following states with its neighbors:
  - **Idle**
  - **Connect**
  - **Open sent**
  - **Open confirm**
  - **Established**
- The **Idle** state begins once the **`neighbor`** command is configured.

Idle State



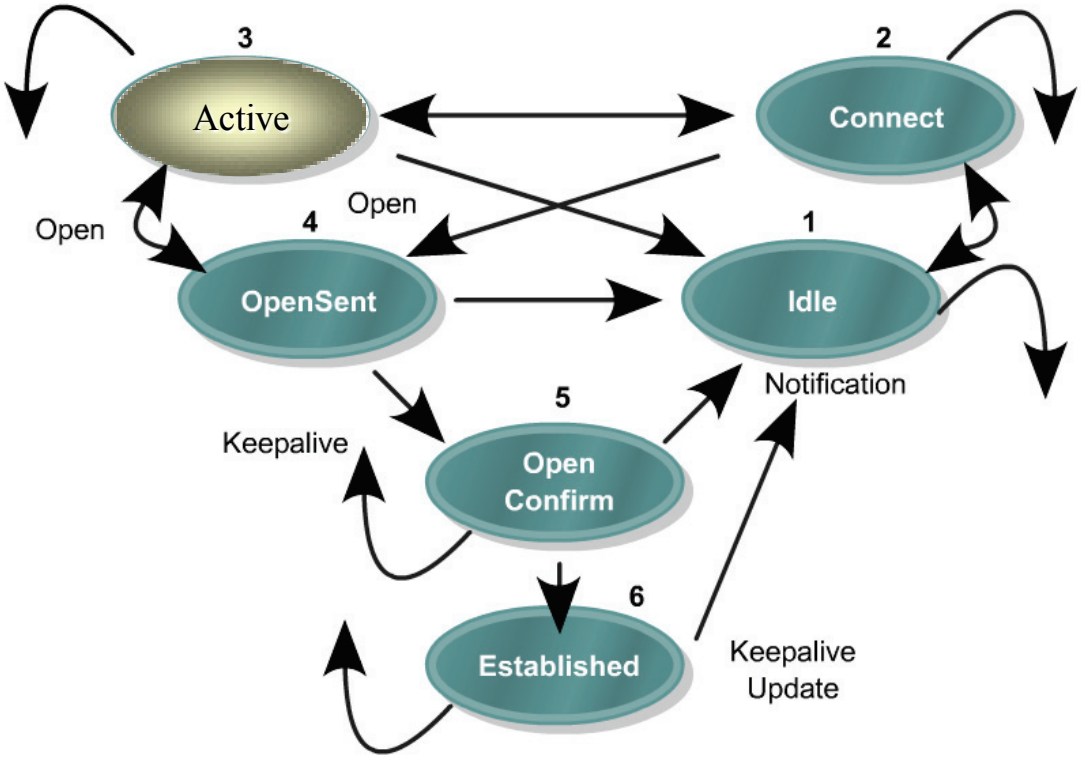
- The router is searching the routing table to see whether a route exists to reach the neighbor.
- If a router remains in this state then the router is:
  - Waiting for a static route to that IP address or network to be configured.
  - Waiting for the IGP to learn about this network from another router.

Connect State



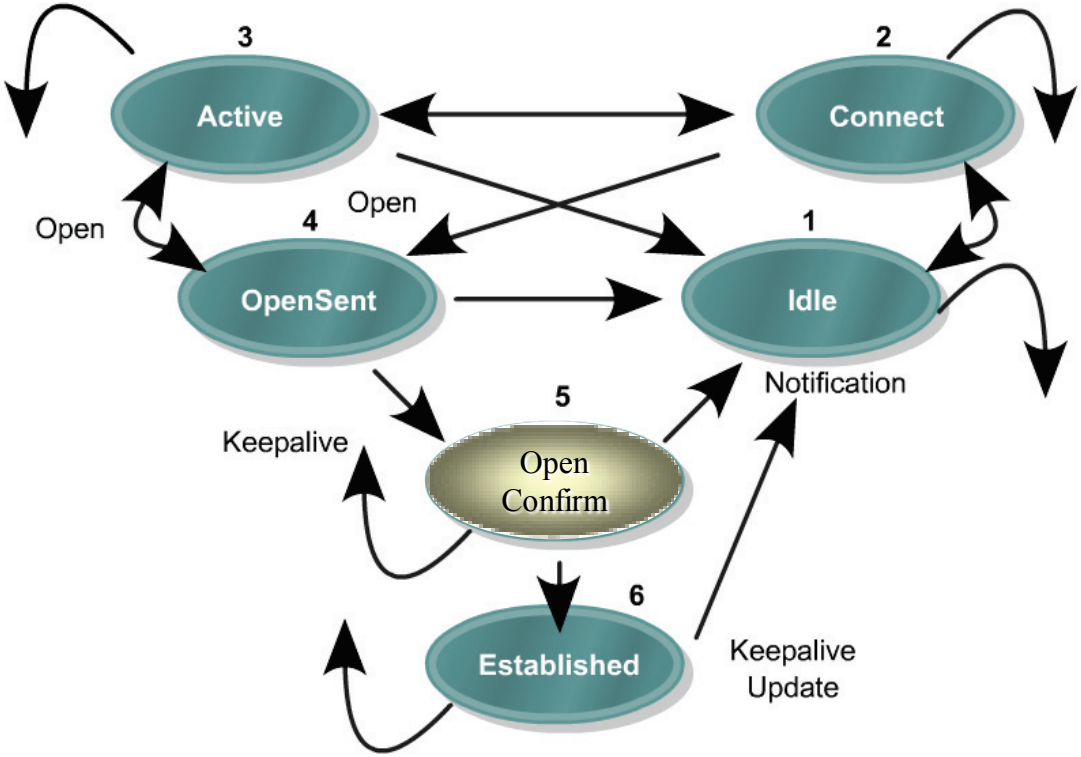
- The router found a route to the neighbor and has completed the three-way TCP handshake.

Active State



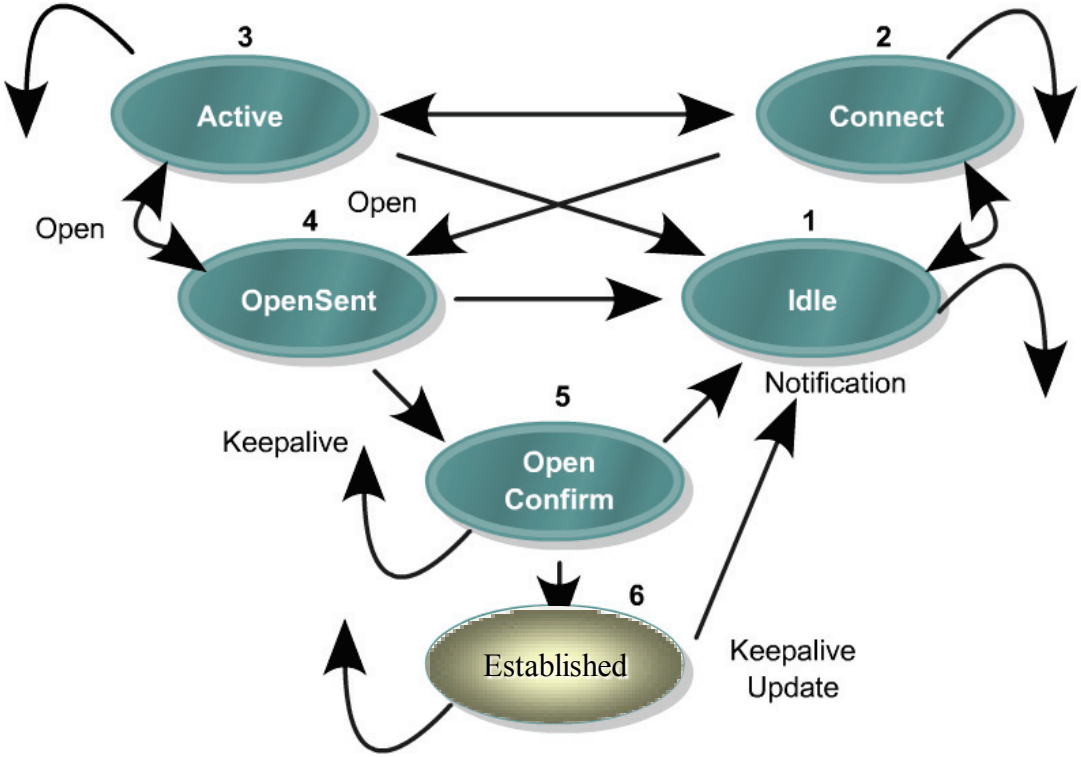
- BGP is trying to acquire a peer by initiating a TCP connection.
- If it is successful, it transitions to OpenSent otherwise the state returns to Idle.
- If the router remains is this state it means that the router has not received a response (open confirm packet) back from the neighbor.
  - Reasons for this include missing neighbor statement or incorrect AS number.

Open Confirm



- The router received agreement on the parameters for establishing a session.

Established State



- This is the desired state for a neighbor relationship.
- It means peering is established and routing begins.

Verifying BGP: `show ip bgp neighbors`

Verify the BGP neighbor relationship.

```
R1# show ip bgp neighbors
BGP neighbor is 172.31.1.3,  remote AS 64998, external link
  BGP version 4, remote router ID 172.31.2.3
  BGP state = Established, up for 00:19:10
  Last read 00:00:10, last write 00:00:10, hold time is 180, keepalive
interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  7          7
    Notifications:          0          0
    Updates:               13         38
<output omitted>
```

BGP Path Manipulation Using Route Maps

Configuring Route Maps in BGP

Sample implementation plan:

- Define and name the route map with the **`route-map`** command.
  - Define the conditions to match (the **`match`** statements).
  - Define the action to be taken when there is a match (the **`set`** statements).
- Define which attribute to alter using the **`neighbor route-map`** router configuration command.
  - Filters incoming or outgoing BGP routes.
- Verify results.

Implementing Route Maps in BGP

```
Router(config)#
route-map map-tag [permit | deny] [sequence-number]
```

- Defines the route map conditions.

```
Router(config-route-map)#
match {criteria}
```

- Defines the criteria to match.

```
Router(config-route-map)#
set {actions}
```

- Defines the action to be taken on a match.

```
Router(config-router)#
neighbor {ip-address | peer-group-name} route-map map-name
{in | out}
```

- Applies the route-map to filter incoming or outgoing BGP routes to a neighbor.

`match` Commands Used in BGP

Command	Description
`match as-path`	Matches the AS_PATH attribute
`match ip address`	Matches any routes that have a destination network number address that is permitted by a standard or extended ACL
`match metric`	Matches routes with the metric specified
`match community`	Matches a BGP community
`match interface`	Matches any routes that have the next hop out of one of the interfaces specified
`match ip next-hop`	Matches any routes that have a next-hop router address that is passed by one of the ACLs specified
`match ip route-source`	Matches routes that have been advertised by routers and access servers at the address that is specified by the ACLs
`match route-type`	Matches routes of the specified type
`match tag`	Matches tag of a route

** Partial list*

`match as-path` Command

- Match a BGP autonomous system path access list.

```
Router(config-route-map)#
```

```
match as-path path-list-number
```

- The *`path-list-number`* is the AS path access list.
  - It can be an integer from 1 to 199.
- The value set by this command overrides global values.

match ip-address Command

- Specify criteria to be matched using ACLs or prefix lists.

```
Router(config-route-map)#
```

match ip address *{access-list-number | name} [...access-list-number | name]* | **prefix-list** *prefix-list-name [..prefix-list-name]*

Parameter	Description
access-list-number | *name*	The number or name of a standard or extended access list to be used to test incoming packets. If multiple access lists are specified, matching any one results in a match.
prefix-list *prefix-list-name*	Specifies the name of a prefix list to be used to test packets. If multiple prefix lists are specified, matching any one results in a match.

`set` Commands Used in BGP

Command	Description
`set weight`	Sets the BGP weight value
`set local-preference`	Sets the LOCAL-PREF attribute value
`set as-path`	Modifes an AS path for BGP routes
`set origin`	Sets the ORIGIN attribute value
`set metric`	Sets the Multi-Exit_Disc (MED) value
`set community`	Sets the BGP communities attribute
`set automatic-tag`	Computes automatically the tag value
`set ip next-hop`	Indicates which IP address to output packets
`set interface`	Indicates which interface to output packets
`set ip default next-hop`	Indicates which default IP address to use to output packets
`set default interface`	Indicates which default interface to use to output packets

** Partial list*

`set weight` Command

- Specify the BGP weight for the routing table.

```
Router(config-route-map)#
```

```
set weight number
```

- The *number* is the weight value.
  - It can be an integer ranging from 0 to 65535.
- The implemented weight is based on the first matched AS path.
- Weights assigned with this command override the weights assigned using the **neighbor weight** command.

`set local-preference` Command

- Specify a preference value for the AS path.

```
Router(config-route-map)#
```

```
set local-preference number-value
```

- The *`number-value`* is the preference value.
  - An integer from 0 to 4294967295.
  - Default 100.

`set as-path` Command

- Modify an AS path for BGP routes.

```
Router(config-route-map)#
```

```
set as-path {tag | prepend as-path-string}
```

Parameter	Description
tag	Converts the tag of a route into an autonomous system path. Applies only when redistributing routes into BGP.
prepend	Appends the string following the keyword **prepend** to the AS path of the route that is matched by the route map. Applies to inbound and outbound BGP route maps.
as-path-string	AS number to prepend to the AS_PATH attribute. The range of values for this argument is 1 to 65535. Up to 10 AS numbers can be entered.

`set metric` Command

- Specify a preference value for the AS path.

```
Router(config-route-map)#
```

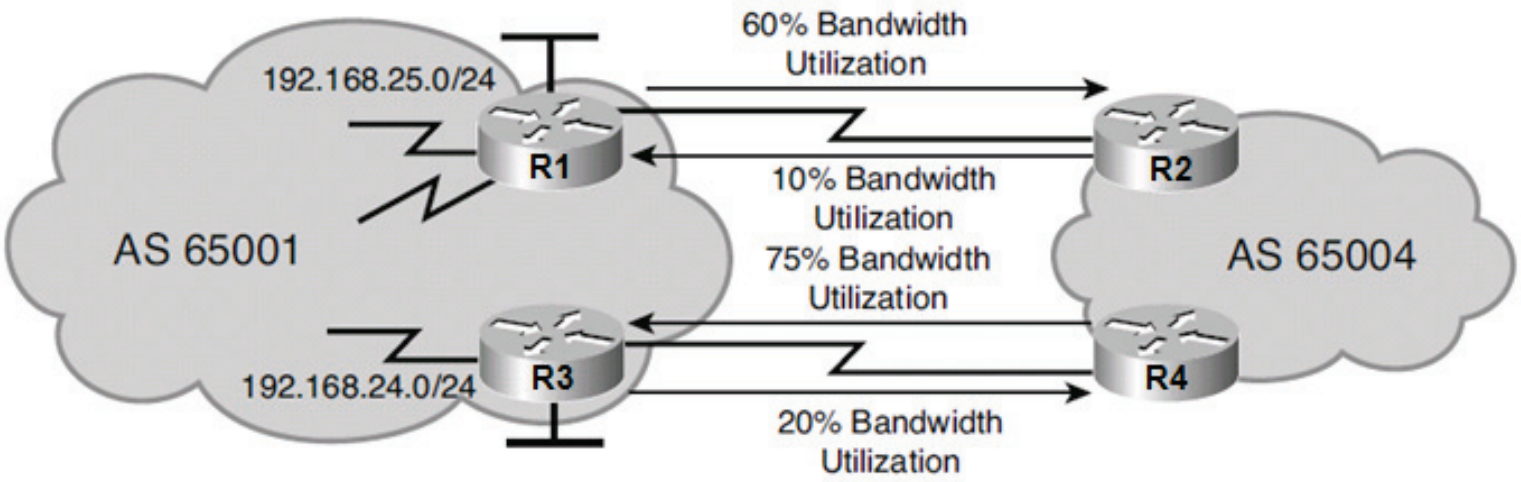
```
set metric metric-value
```

- The *`metric-value`* is use to set the MED attribute.
  - An integer from 0 to 294967295.

BGP Path Manipulation

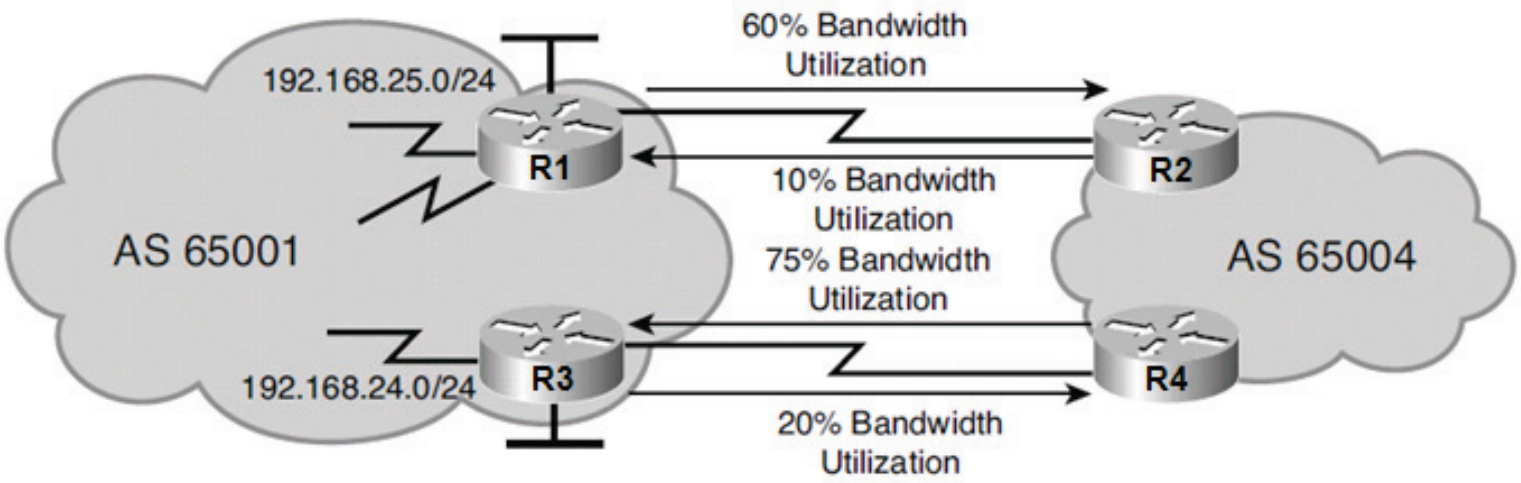
- Unlike IGPs, BGP was never designed to choose the quickest path.
- BGP was designed to manipulate traffic flow to maximize or minimize bandwidth use.

BGP Without Routing Policy Example #1



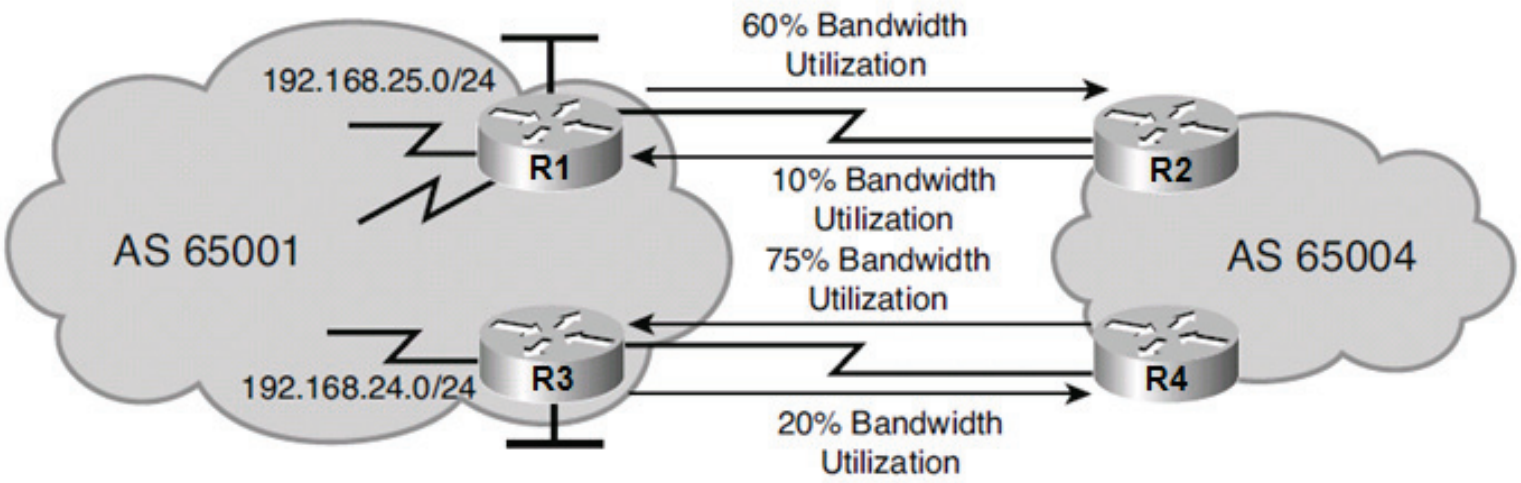
- In this example consider that:
  - R1 is using 60% of its outbound bandwidth to AS 65004.
  - R3 is using 20% of its outbound bandwidth to AS 65004.
  - R2 is using 10% of its outbound bandwidth to AS 65001.
  - R4 is using 75% of its outbound bandwidth to AS 65001.
- Traffic should be diverted using the local preference attribute.
  - The weight attribute could not be used in this scenario since there are two edge routers.

Which traffic should be re-routed?



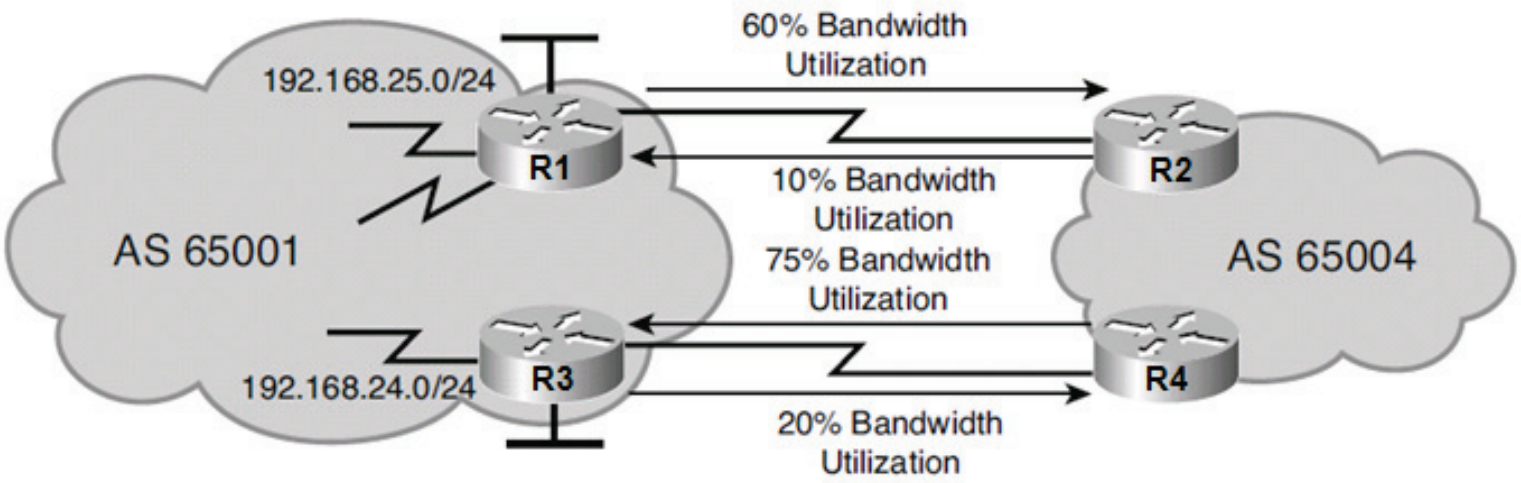
- To determine which path to manipulate, perform a traffic analysis on Internet-bound traffic by examining the most heavily visited addresses, web pages, or domain names.
  - Examine network management records or accounting information.
- If a heavily accessed traffic pattern is identified, a route map could be used to divert that traffic over the lesser used links

BGP With Routing Policy Example #1



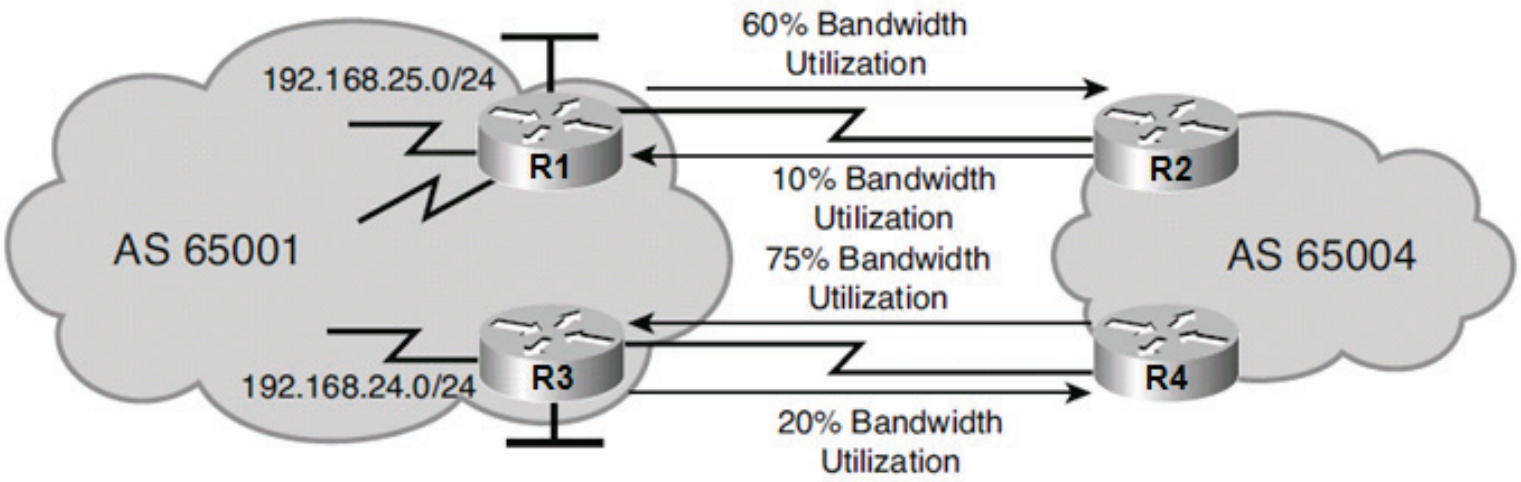
- For example, assume that 35% of all traffic from AS 65001 has been going to http://www.cisco.com.
  - The administrator does a reverse DNS lookup and obtains the Cisco IP address and AS number.
- A route map can be used to change the local preference to manipulate packets destined to Cisco's network over the less used links.

BGP Routing Policy Example #2



- Notice that the inbound load to R3 (75%) is much higher in bandwidth utilization than the inbound load to R1 (10%).
- The BGP MED attribute can be used to manipulate how traffic enters autonomous system 65001.
- For example, R1 in AS 65001 can announce a lower MED for routes to network 192.168.25.0/24 to AS 65004 than R3 announces.

BGP Routing Policy Example #2



- Keep in mind that the MED is considered a *recommendation* because the receiving autonomous system can override it by manipulating another variable that is considered before the MED is evaluated.
- For example, R2 and R4 in AS 65004 could be configured with their own local preference policy which would override the MED recommendation from AS 65001.

BGP Route Selection Process

1. Prefer highest Weight
2. Prefer highest LOCAL_PREF
3. Prefer locally generated routes
4. Prefer shortest AS_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID
10. Prefer routes with lowest neighbor IP address

Change the Weight

- The weight attribute is used only when one router is multihomed and determines the best path to leave the AS.
  - Only the local router is influenced.
  - Higher weight routes are preferred.
- There are two ways to alter the route weight:
  - To change the weight for all updates from a neighbor use the neighbor weight router configuration command.
  - To change the weight of specific routes / as path, use route maps.

BGP Route Selection Process

1. Prefer highest Weight
2. Prefer highest LOCAL_PREF
3. Prefer locally generated routes
4. Prefer shortest AS_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID
10. Prefer routes with lowest neighbor IP address

Changing the Default Weight Example

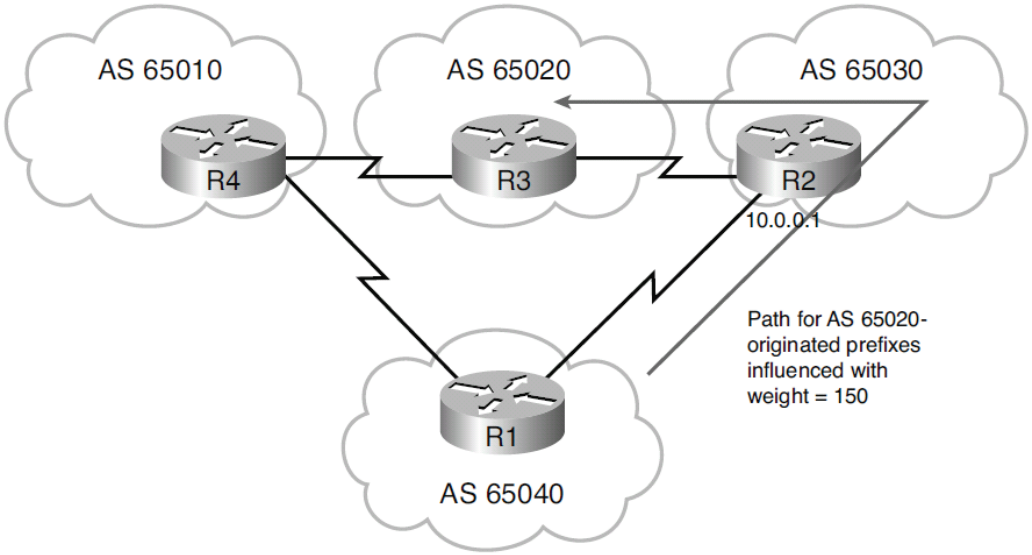
- Assign a default weight to all routes from a peer.

```
Router(config-router)#
```

```
neighbor {ip-address | peer-group-name} weight number
```

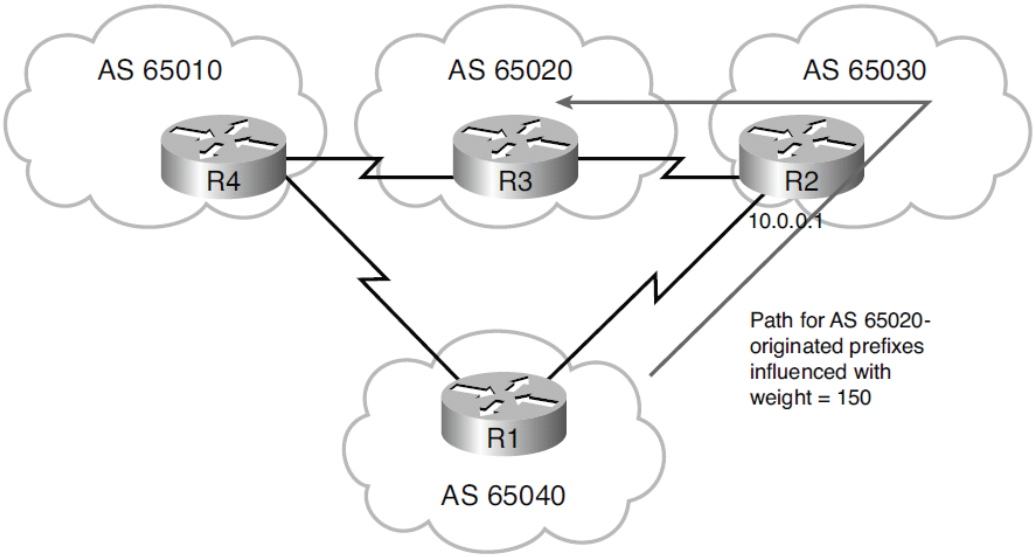
- Routes learned through another BGP peer have a default weight of 0 and routes sourced by the local router have a default weight of 32768.
- The *number* is the weight to assign.
  - Acceptable values are from 0 to 65535.
- The route with the highest weight will be chosen as the preferred route when multiple routes are available to a particular network.
- **Note**: The weights assigned with the **set weight route-map** command override the weights assigned using the **neighbor weight** command.

Changing Weight with Route Map Example



- In this example consider that:
  - The routing policy dictates that for any network originated by AS 65020, use the path to AS 65030 as the primary way out of AS 65040.
  - If R1 needs to access routes connected to R3, then it goes through R2.
- This can be achieved by placing a higher weight (150) on all incoming announcements from AS 65030 (10.0.0.1), which carry the information about the network originated in AS 65020.

Changing Weight with Route Map Example




```
R1(config)# route-map SET-WEIGHT permit 10
R1(config-route-map)# match as-path 10
R1(config-route-map)# set weight 150
R1(config-route-map)#
R1(config-route-map)# route-map SET-WEIGHT permit 20
R1(config-route-map)# set weight 100
R1(config-route-map)# exit
R1(config)# ip as-path access-list 10 permit _65020$
R1(config)#
R1(config)# router bgp 65040
R1(config-router)# neighbor 10.0.0.1 remote-as 65030
R1(config-router)# neighbor 10.0.0.1 route-map SET-WEIGHT in
```

Configure an Autonomous System ACL

- Configure an autonomous system path filter.

```
Router(config-router)#
ip as-path access-list acl-number {permit | deny}
  regexp
```

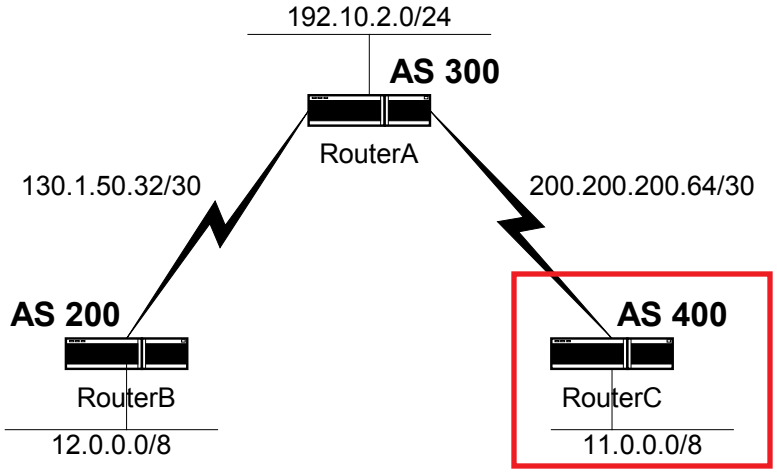
- Similar to an IP ACL, this command is used to configure an AS path filter using a regular expression .
- The *acl-number* is a value from 1 to 500 that specifies the AS_PATH access list number.
- The *regexp* regular expression defines the AS-path filter.

Regular Expression Syntax

- **Atom**: A single character.
  - `.` matches any single character.
  - `^` matches the start of the input string.
  - `$` matches the end of the input string.
  - `\` matches the character.
- **Piece**: one of these symbols
  - `*` matches 0 or more sequences of the atom.
  - `+` matches 1 or more sequences of the atom.
  - `?` matches the atom or the null string.
- **Branch**: 1 or more concatenated pieces.
- **Range**: A sequence of characters within square brackets.
  - Example is [abcd].

Regular Expression Examples

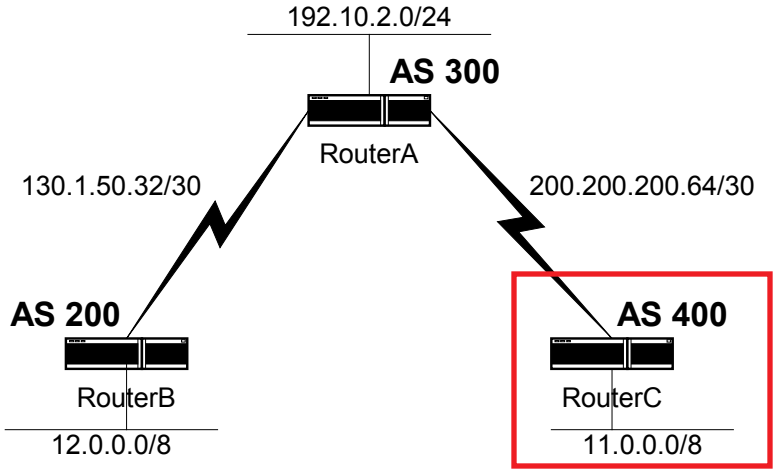
Regular Expression	Resulting Expression
`a*`	Expression indicates any occurrence of the letter "a", which includes none
`a+`	indicates that at least one occurrence of the letter "a" must be present
`ab?a`	Expression matches "aa" or "aba".
`_100_`	Expression means via AS100 (matches beginning or ending of an input stream)
`_100$`	Expression indicates an origin of AS100.
`^100 .*`	Expression indicates transmission from AS100
`^$`	Expression indicates origination from this AS




```
RouterC#show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0       200.200.200.65           0             0 300 i
```

```
RouterC# show ip bgp regexp ^300
```

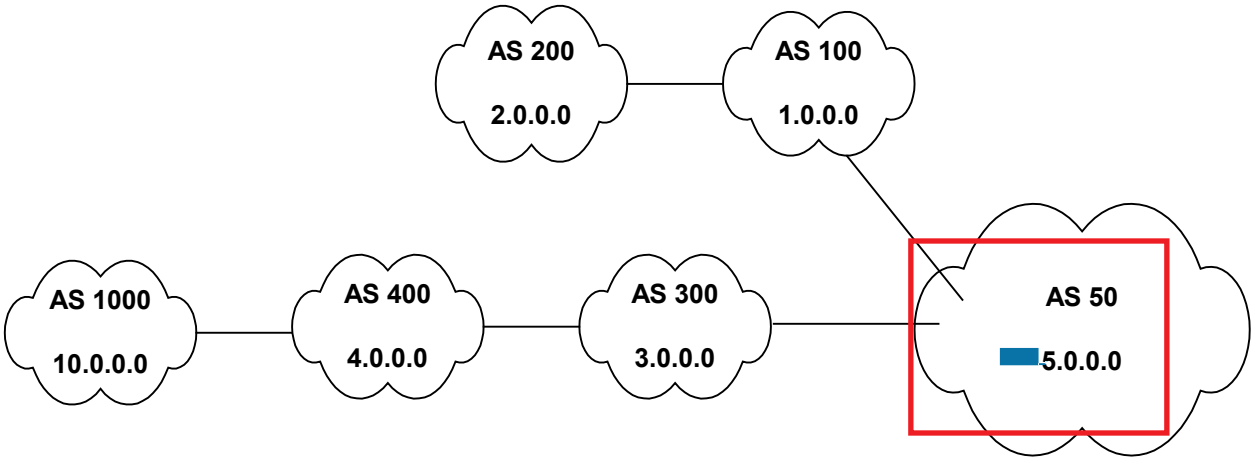
- Match beginning of input string, AS_PATH, = 300
- Last prepended AS was 300:
- Routes matched: 12.0.0.0 and 192.10.2.0




```
RouterC#show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0       200.200.200.65           0             0 300 i
```

```
RouterC# show ip bgp regexp 200$
```

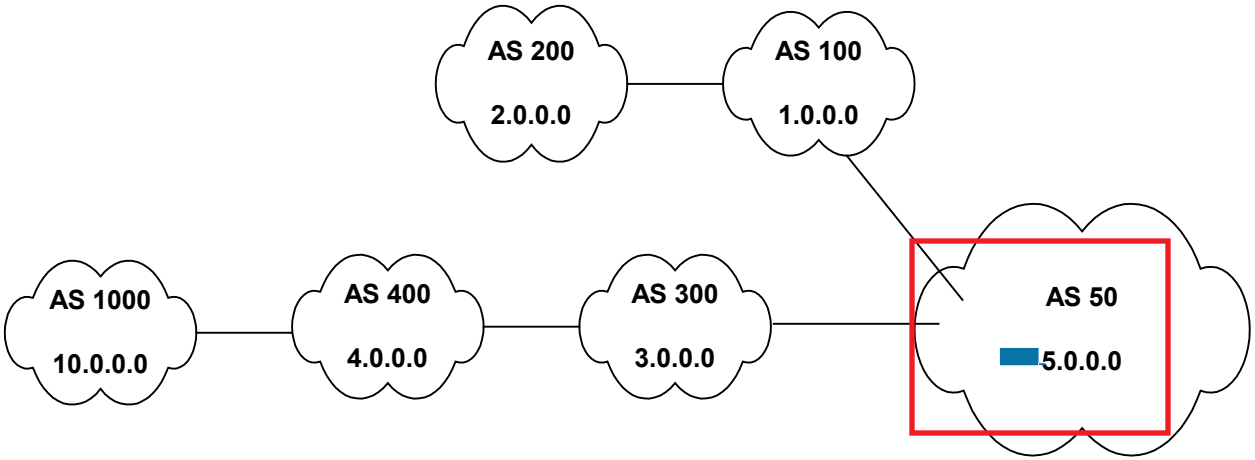
- Match end of input string, AS_PATH, = 200
- Originating AS = 200:
- Routes matched : 12.0.0.0




```
AS50#show ip bgp
   Network                Path
*> 5.0.0.0                i
*> 1.0.0.0                100 i
*> 2.0.0.0                100 200 i
*> 3.0.0.0                300 i
*> 4.0.0.0                300 400 i
*> 10.0.0.0                    300 400 1000 I
```

```
AS50#show ip bpg regexp 100
```

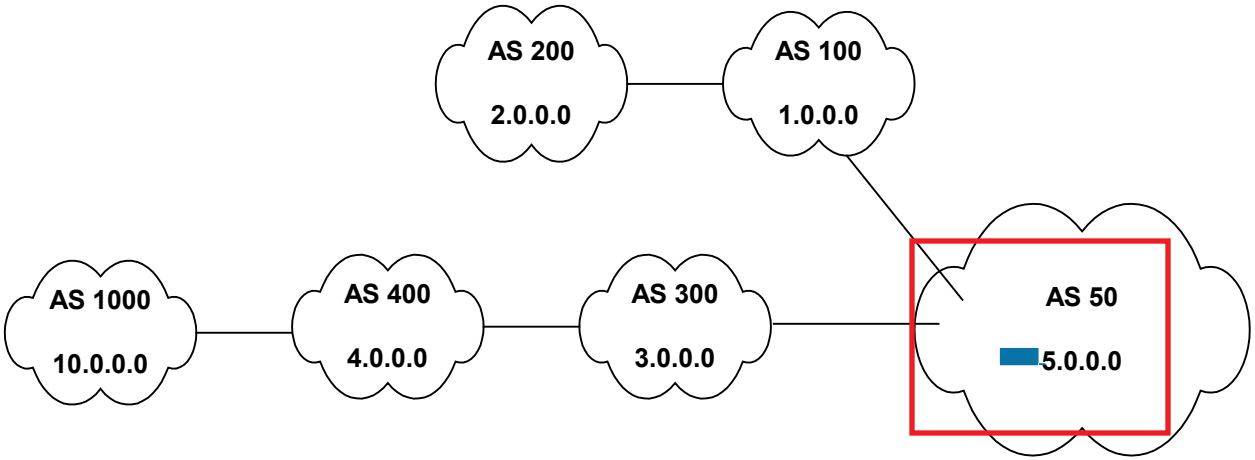
- Match input string, AS_PATH, containing 100, including 1000
- Routes matched : 1.0.0.0, 2.0.0.0, 10.0.0.0




```
AS50#show ip bgp
   Network                Path
*> 5.0.0.0                i
*> 1.0.0.0                100 i
*> 2.0.0.0                100 200 i
*> 3.0.0.0                300 i
*> 4.0.0.0                300 400 i
*> 10.0.0.0                    300 400 1000 I
```

```
AS50#show ip bpg regexp ^100_
```

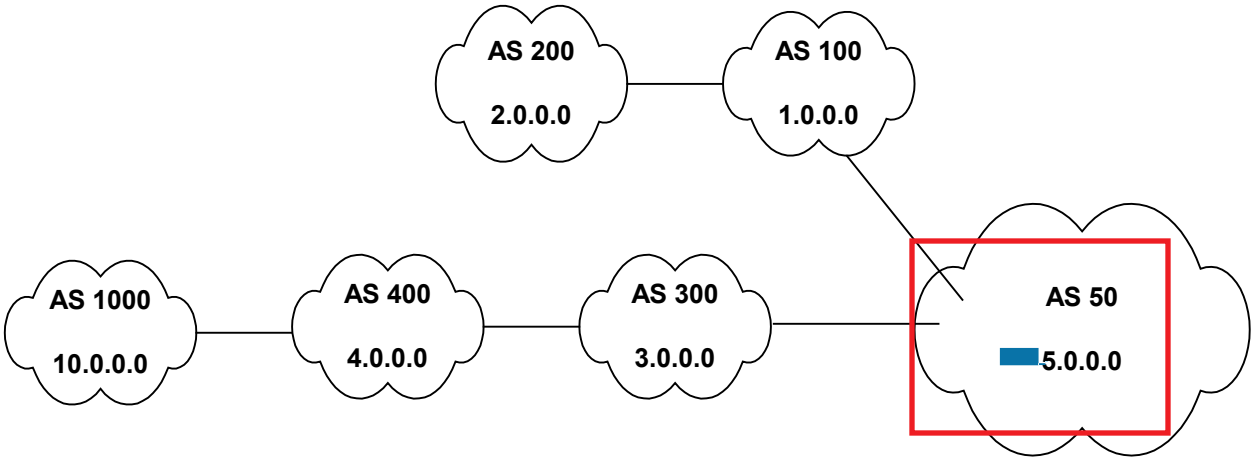
- Match beginning of input string, AS_PATH, = 100
- Last prepended AS was 100:
- Routes matched : 1.0.0.0, 2.0.0.0




```
AS50#show ip bgp
   Network                Path
*> 5.0.0.0                i
*> 1.0.0.0                100 i
*> 2.0.0.0                100 200 i
*> 3.0.0.0                300 i
*> 4.0.0.0                300 400 i
*> 10.0.0.0                   300 400 1000 I
```

```
AS50# show ip bgp regexp _400$
```

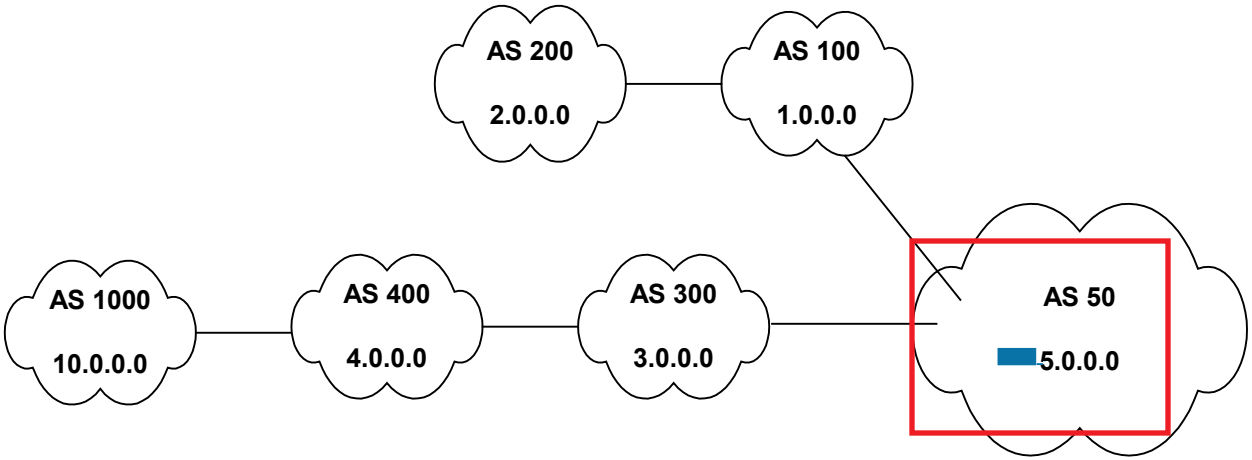
- Match end of input string, AS_PATH, = 400
- Originating AS = 400:
- Routes matched : 4.0.0.0




```
AS50#show ip bgp
   Network                Path
*> 5.0.0.0                i
*> 1.0.0.0                100 i
*> 2.0.0.0                100 200 i
*> 3.0.0.0                300 i
*> 4.0.0.0                300 400 i
*> 10.0.0.0                    300 400 1000 I
```

```
AS50#show ip bpg regexp _400_
```

- Match anywhere in input string, AS_PATH, 400
- Routes matched : 4.0.0.0, 10.0.0.0




```
AS50#show ip bgp
   Network                Path
*> 5.0.0.0                i
*> 1.0.0.0                100 i
*> 2.0.0.0                100 200 i
*> 3.0.0.0                300 i
*> 4.0.0.0                300 400 i
*> 10.0.0.0                    300 400 1000 I
```

```
AS50#show ip bgp regexp ^300$
```

- Match input string that starts and ends at 300
- Routes that originated from directly connected AS 300 customer
- Routes matched : 3.0.0.0

Change the Local Preference

- The local preference is used only within an AS (between IBGP speakers) to determine the best path to leave the AS.
  - Higher values are preferred.
  - The local preference is set to 100 by default.
- There are two ways to alter the local preference:
  - To change the default local-preference for all routes advertised by the router use the **`bgp default local-preference`** *`value`* router configuration command.
  - To change the local-preference of specific routes / as path, use route maps.

BGP Route Selection Process

1. Prefer highest Weight
2. Prefer highest LOCAL_PREF
3. Prefer locally generated routes
4. Prefer shortest AS_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID
10. Prefer routes with lowest neighbor IP address

Setting Default Local Preference Example

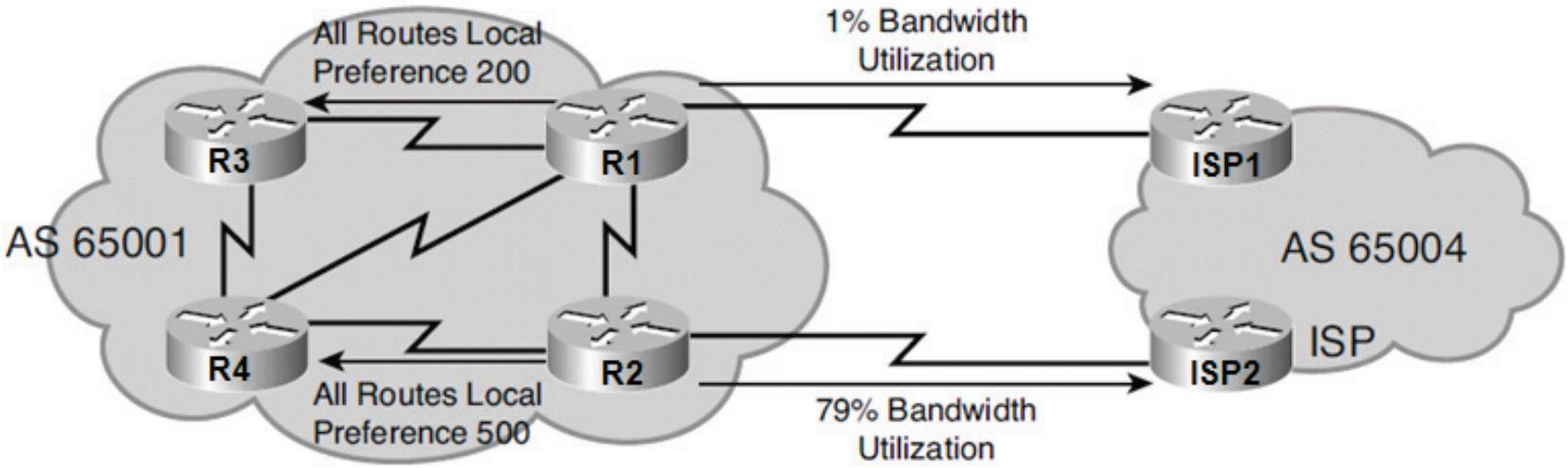
- Change the default local preference for outgoing routes.

```
Router(config-router)#
```

```
bgp default local-preference number
```

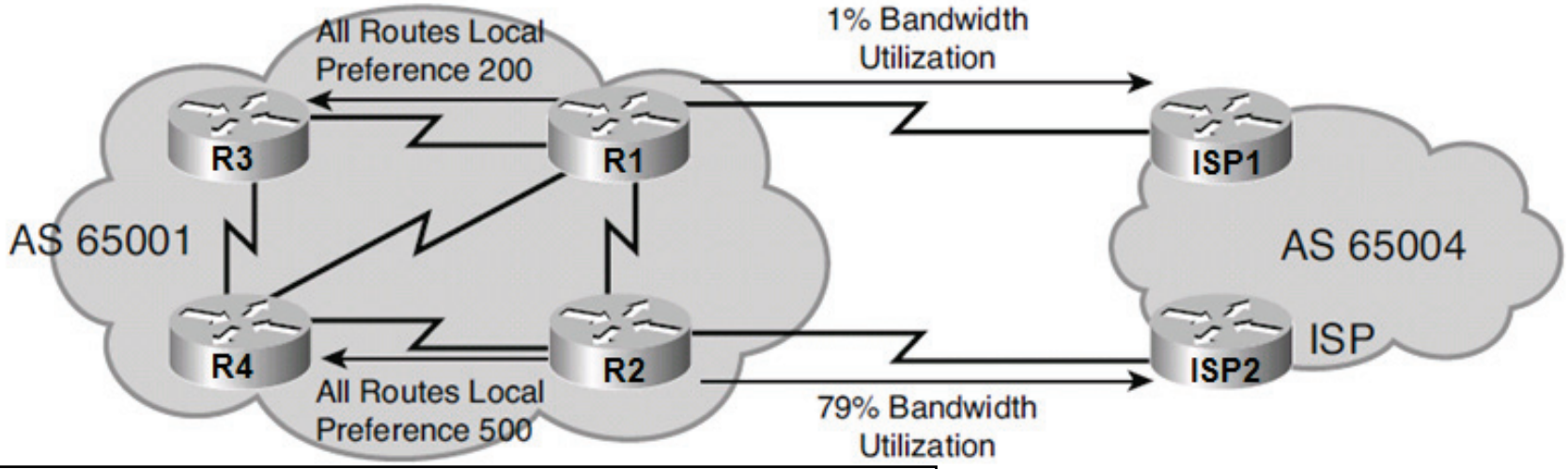
- The local preference attribute applies a degree of preference to a route during the BGP best path selection process.
  - The attribute is exchanged only between iBGP peers.
  - The route with the highest local preference is preferred.
- The *number* is the local preference value from 0 to 4294967295.
  - Cisco IOS software applies a local preference value of 100.
- Note: The local preference assigned with the **set local-preference route-map** command override the weights assigned using this command.

Setting Default Local Preference Example



- The BGP routing policy in this example dictates that:
  - The default local preference for all routes on R1 should be set to 200.
  - The default local preference for all routes on R2 should be set to 500.

Setting Default Local Preference Example

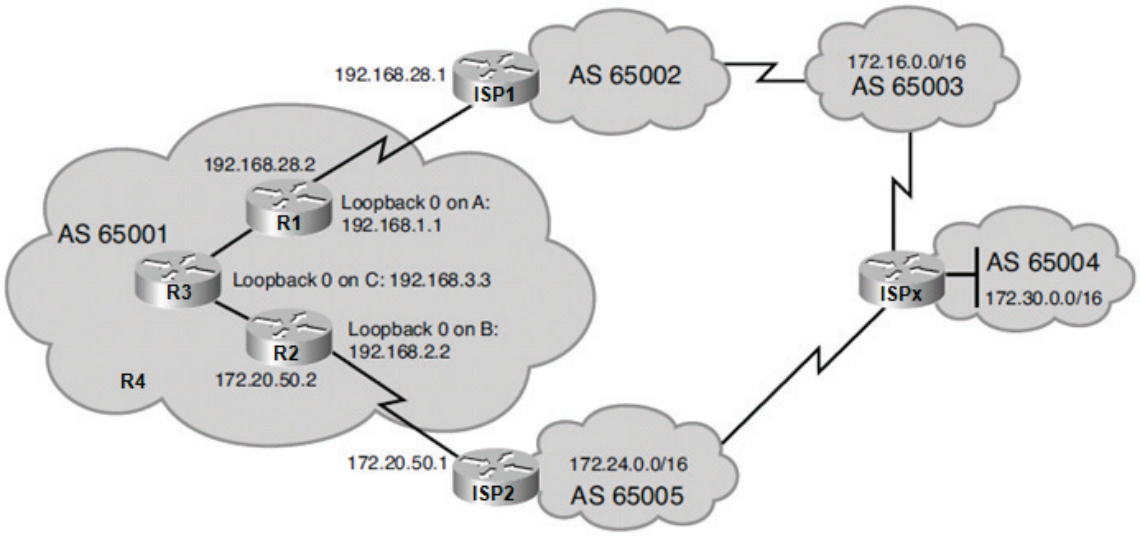



```
R1(config)# router bgp 65001
R1(config-router)# bgp default local-preference 200
R1(config-router)#
```

```
R2(config)# router bgp 65001
R2(config-router)# bgp default local-preference 500
R2(config-router)#
```

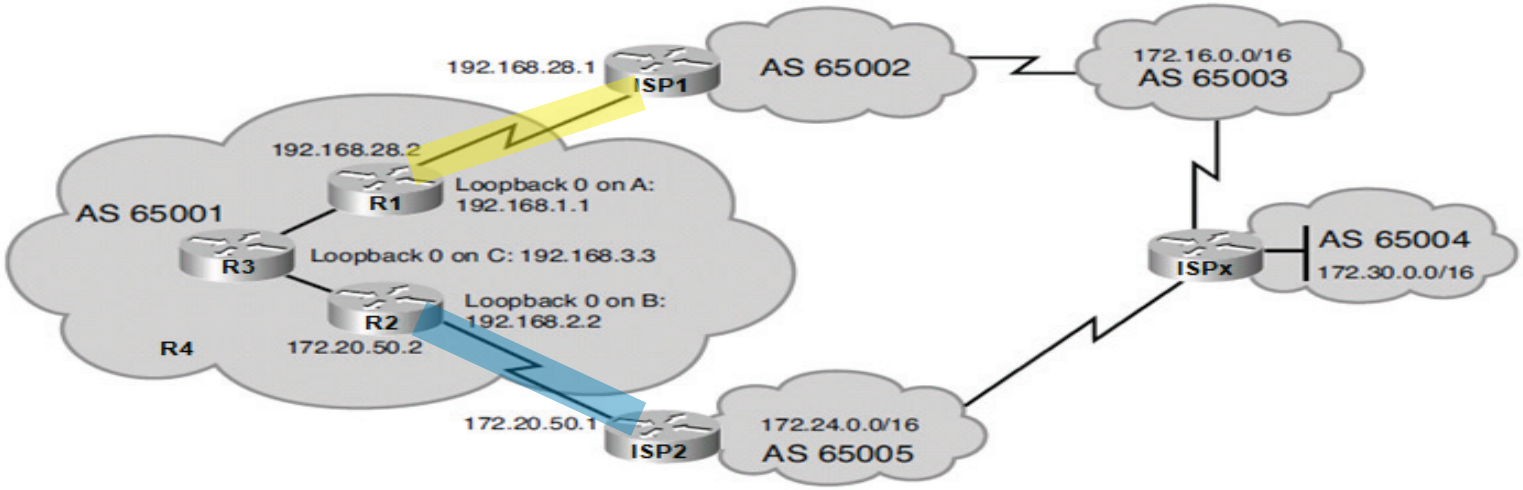
- The resulting configuration makes the IBGP routers in AS 65001 send all Internet bound traffic to R2, but the R1 to ISP1 link is underutilized.
  - Route maps could be configured to select specific routes to have a higher local preference.

Local Preference and Route Map Example



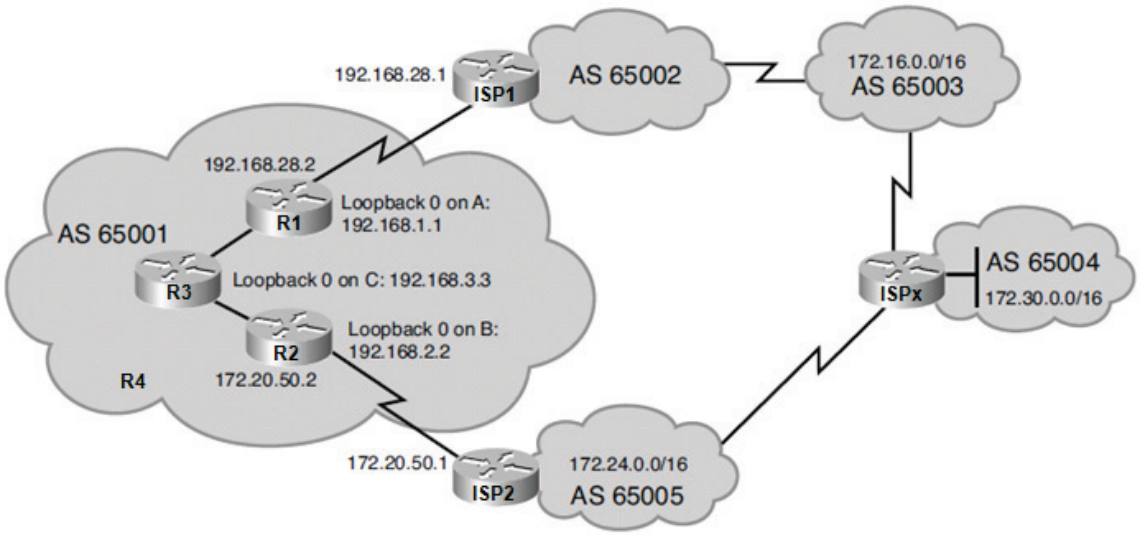
- The BGP routing policy results in the following:
  - All routes have a weight of 0 and a default local preference of 100.
  - BGP uses the shortest AS-path to select the best routes as follows:
    - For network 172.16.0.0, the shortest AS-path is through ISP1.
    - For network 172.24.0.0, the shortest AS-path is through ISP2.
    - For network 172.30.0.0, the shortest AS-path is through ISP2.

Local Preference and Route Map Example



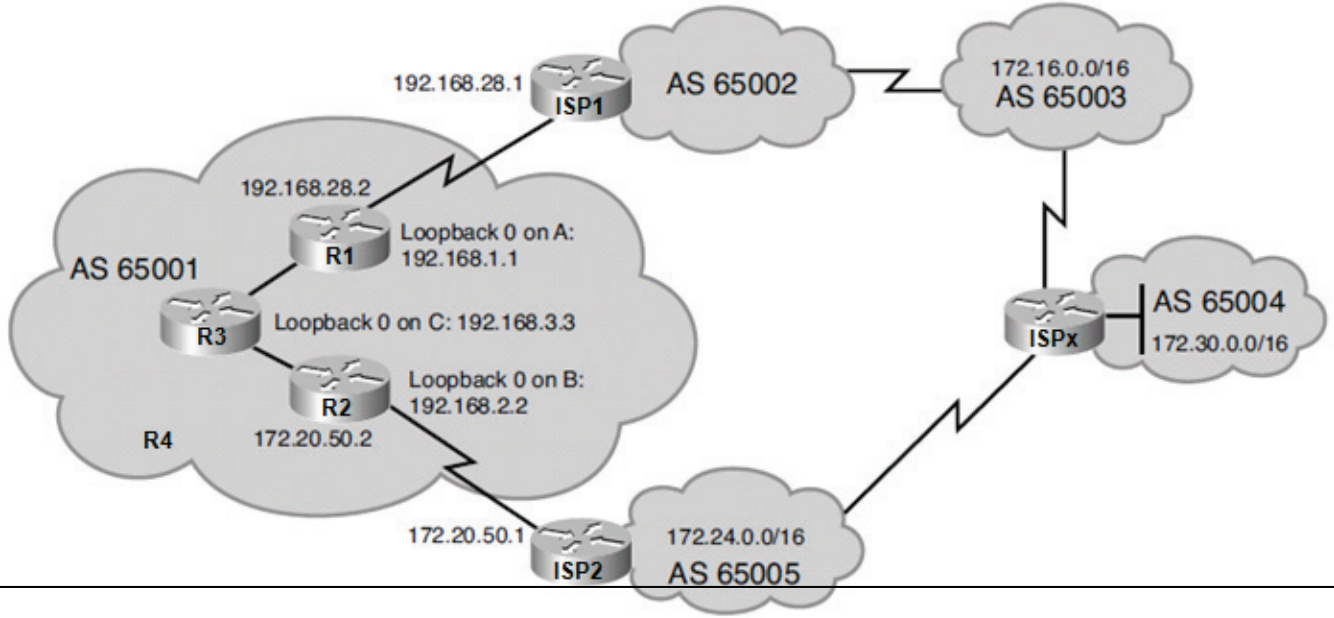

```
R3# show ip bgp
BGP table version is 7, local router ID is 192.168.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,  r
   RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
Network             Next Hop      Metric LocPrf  Weight Path
* i172.16.0.0       172.20.50.1              100       0 65005 65004 65003 i
*>i                 192.168.28.1             100       0 65002 65003 i
*>i172.24.0.0       172.20.50.1              100       0 65005 i
* i                 192.168.28.1             100       0 65002 65003 65004 65005 i
*>i172.30.0.0       172.20.50.1              100       0 65005 65004 i
* i                 192.168.28.1             100       0 65002 65003 65004i
```

Local Preference and Route Map Example



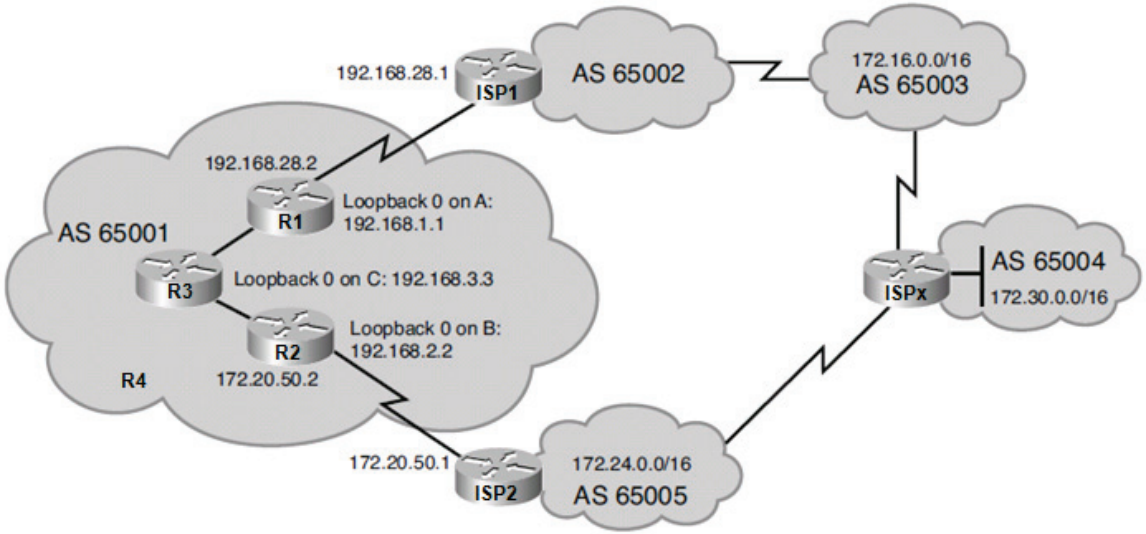
- A traffic analysis reveals the following traffic patterns:
  - 10% of traffic flows from R1 to ISP1 to network 172.16.0.0.
  - 50% of Internet traffic flow from R2 to ISP2 to networks network 172.24.0.0 and network 172.30.0.0.
  - The remaining 40 percent is going to other destinations.
- A solution is to use route maps to divert traffic to 172.30.0.0 through R1.

Local Preference and Route Map Example



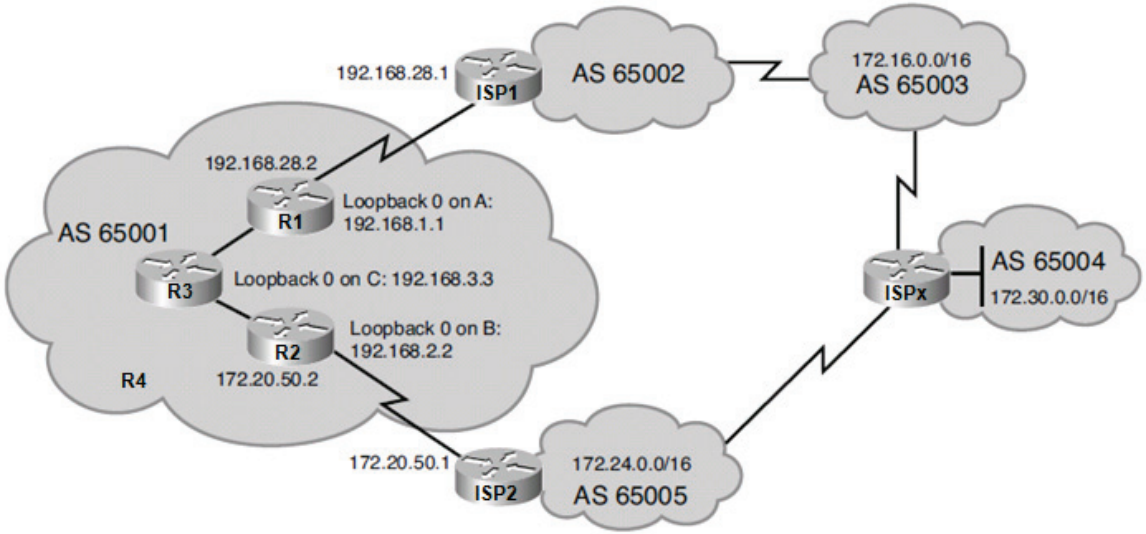

```
R1(config)# access-list 65 permit 172.30.0.0 0.0.255.255
R1(config)#
R1(config)# route-map LOCAL_PREF permit 10
R1(config-route-map)# match ip address 65
R1(config-route-map)# set local-preference 400
R1(config-route-map)#
R1(config-route-map)# route-map LOCAL_PREF permit 20
R1(config-route-map)# exit
R1(config)#
```

Local Preference and Route Map Example




```
R1(config)# router bgp 65001
R1(config-router)# neighbor 192.168.2.2 remote-as 65001
R1(config-router)# neighbor 192.168.2.2 update-source loopback0
R1(config-router)# neighbor 192.168.3.3 remote-as 65001
R1(config-router)# neighbor 192.168.3.3 update-source loopback0
R1(config-router)# neighbor 192.168.28.1 remote-as 65002
R1(config-router)# neighbor 192.168.28.1 route-map LOCAL_PREF in
R1(config-router)# exit
R1(config)#
```

Local Preference and Route Map Example




```
R3# show ip bgp
BGP table version is 7, local router ID is 192.168.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,  r
   RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
* i172.16.0.0       172.20.50.1                100        0 65005 65004 65003 i
*>i                 192.168.28.1               100        0 65002 65003 i
*>i172.24.0.0       172.20.50.1                100        0 65005 i
* i                 192.168.28.1               100        0 65002 65003 65004 65005 i
* i172.30.0.0       172.20.50.1                100        0 65005 65004 i
*>i                 192.168.28.1               400        0 65002 65003 65004i
```

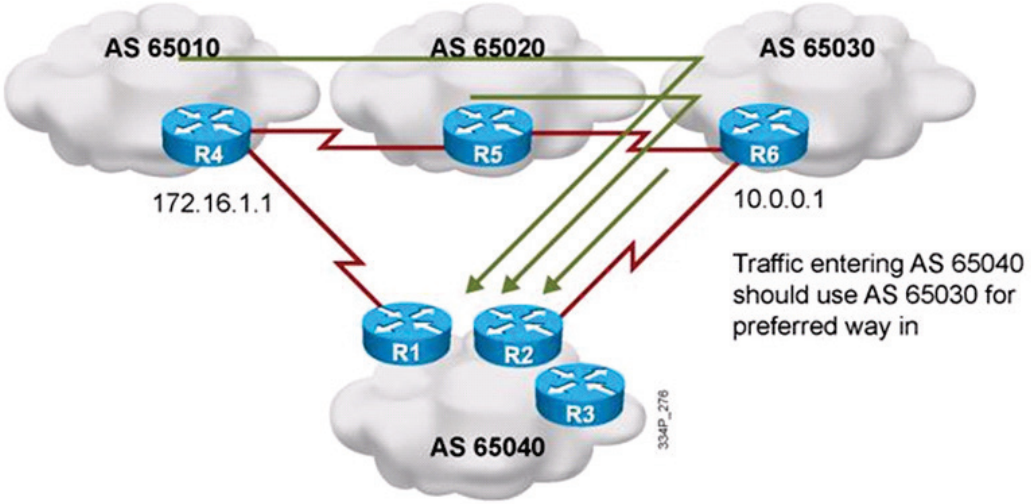
Modifying the AS Path

- By default, if no BGP path selection tools are configured to influence traffic flow (i.e. weight, local-preference), BGP uses the shortest AS path, regardless of available bandwidth.
- To influence the path selection based on the AS_PATH, configure AS-path prepending.
  - The AS path is extended with multiple copies of the AS number of the sender making it appear longer.

BGP Route Selection Process

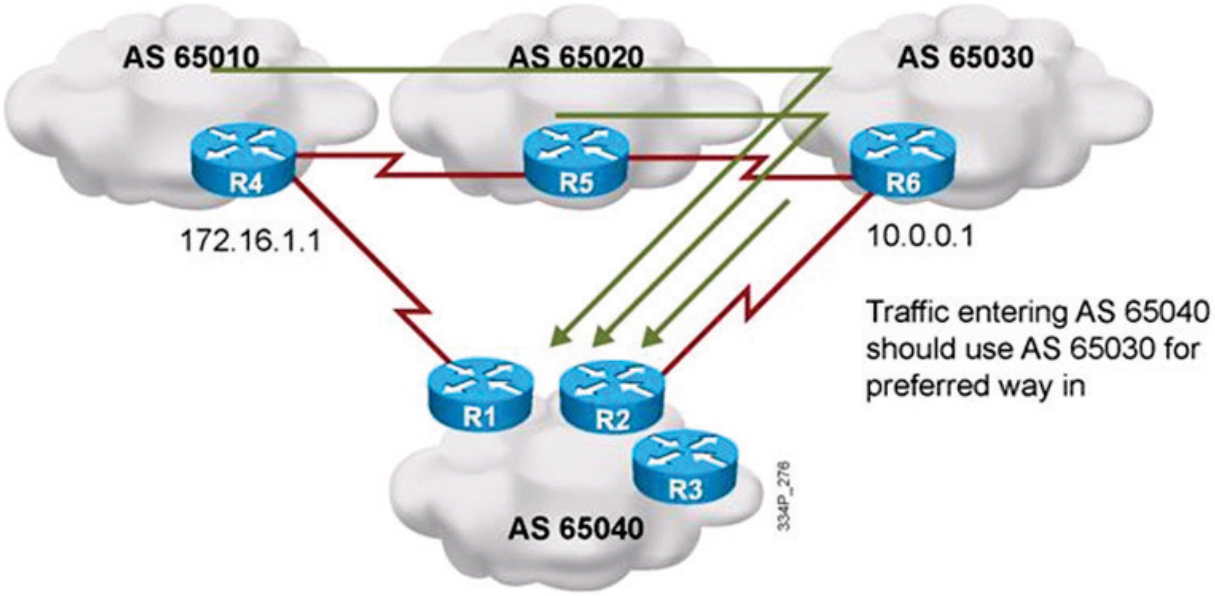
1. Prefer highest Weight
2. Prefer highest LOCAL_PREF
3. Prefer locally generated routes
4. Prefer shortest AS_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID
10. Prefer routes with lowest neighbor IP address

Modifying the AS Path Example



- The BGP routing policy in this example dictates that:
  - Traffic entering AS 65040 should be through R6 in AS 65030 and not R4 in AS 65010.
- One way to do this is make R1 advertise the AS 65040 networks with a less desirable AS path by configuring AS-path prepending.

Modifying the AS Path Example




```
R1(config)# route-map SET-AS-PATH permit 10
R1(config-route-map)# set as-path prepend 65040 65040 65040
R1(config-route-map)# exit
R1(config)# router bgp 65040
R1(config-router)# neighbor 172.16.1.1 remote-as 65010
R1(config-router)# neighbor 172.16.1.1 route-map SET-AS-PATH out
R1(config-router)# exit
R1(config)#
```

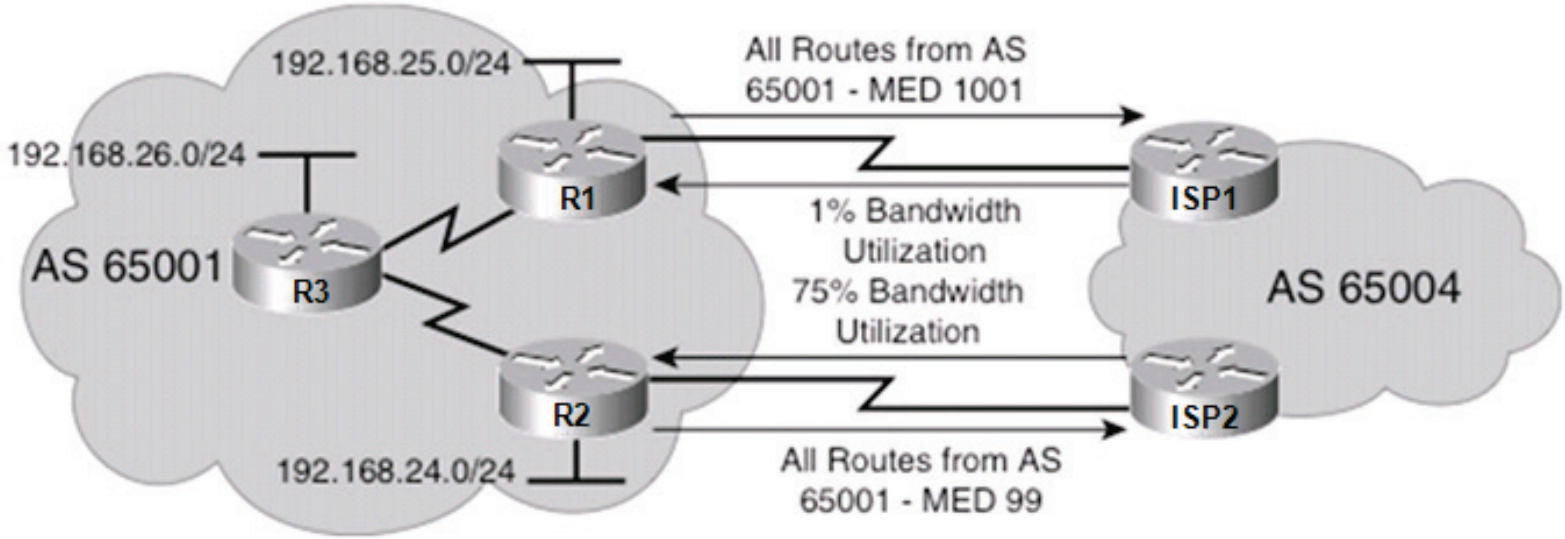
Setting the MED

- MED is used to decide how to enter an AS when multiple paths exist.
  - When comparing MED values for the same destination network in the BGP path-selection process, the lowest MED value is preferred.
  - Default is 0.
- However, because MED is evaluated late in the BGP path-selection process, it usually has no influence.
- There are two ways to alter the MED:
  - To change the MED for all routes use the **`default-metric`** router configuration command.
  - To change the MED of specific routes / as path, use route maps.

BGP Route Selection Process

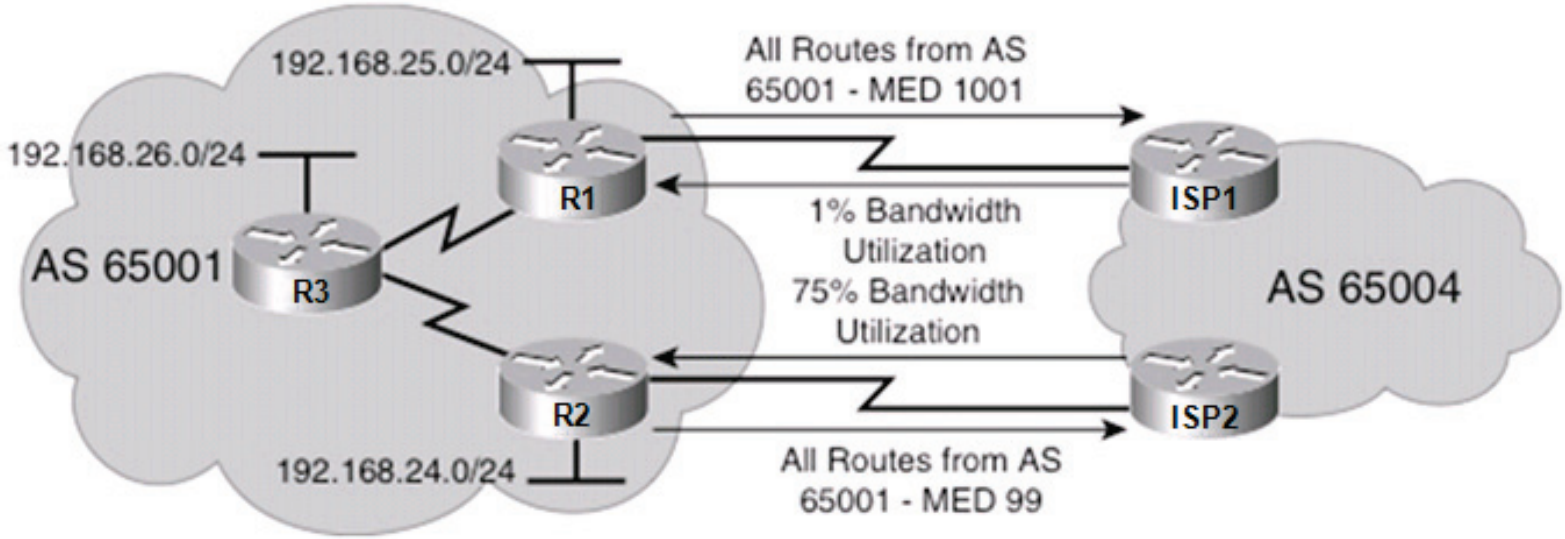
1. Prefer highest Weight
2. Prefer highest LOCAL_PREF
3. Prefer locally generated routes
4. Prefer shortest AS_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID
10. Prefer routes with lowest neighbor IP address

Setting the Default MED Example



- The BGP routing policy in this example dictates that:
  - The default MED of R1 should be changed to 1001.
  - The default MED of R2 should be changed to 99.

Setting the Default MED Example

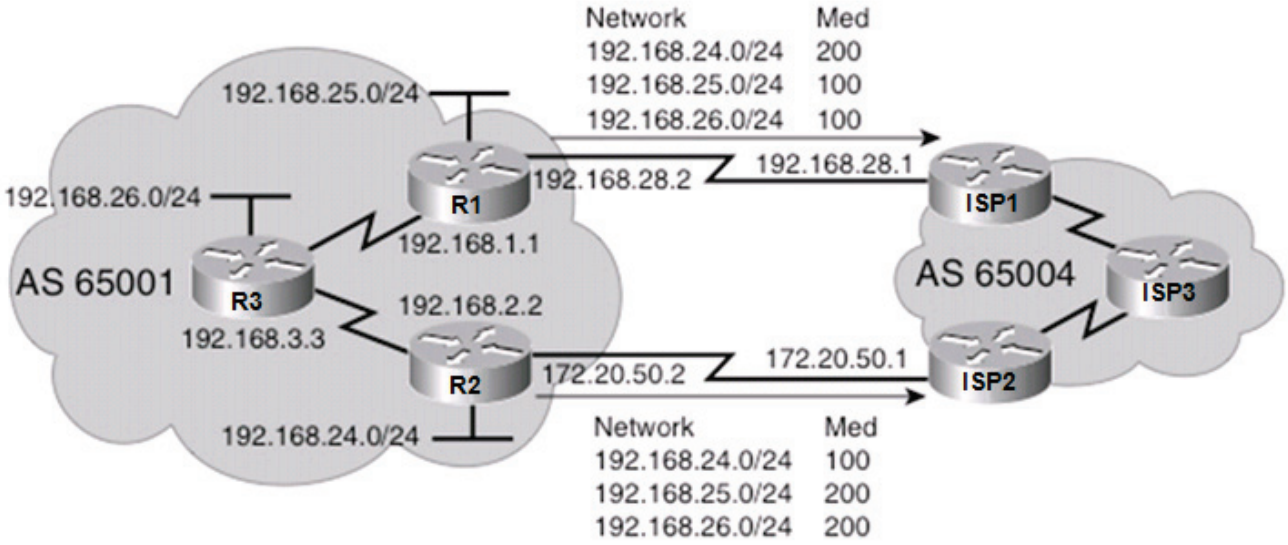



```
R1(config)# router bgp 65001
R1(config-router)# default metric 1001
R1(config-router)#
```

```
R2(config)# router bgp 65001
R2(config-router)# default metric 99
R2(config-router)#
```

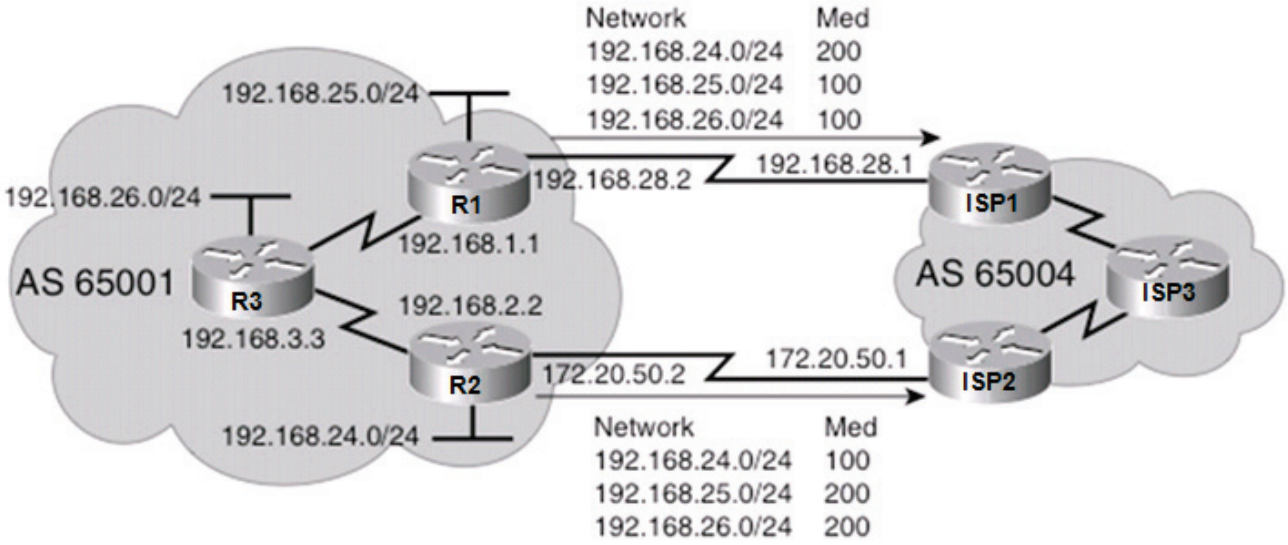
- The results are that the inbound bandwidth utilization on:
  - R1 to ISP1 link has decreased to almost nothing except for BGP routing updates.
  - R2 to ISP2 link has increased due to all returning packets from AS 65004.
  - A better solution is to have route maps configured that will make some networks have a lower MED through R1 and other networks to have a lower MED through R2.

Setting the MED with Route Maps Example



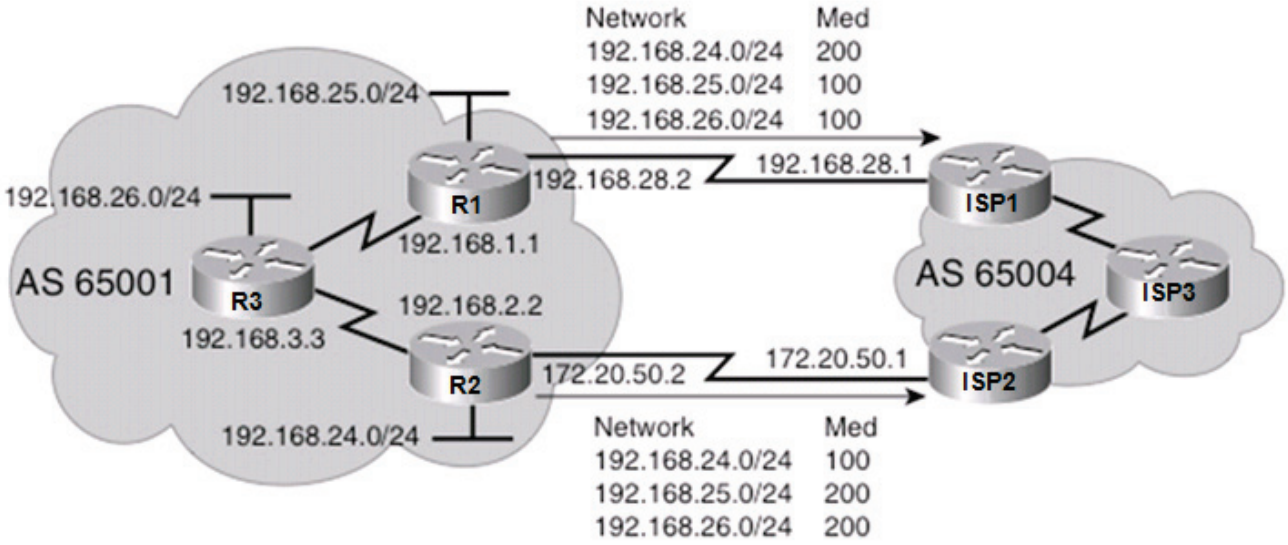

```
R1(config)# access-list 66 permit 192.168.25.0 0.0.0.255
R1(config)# access-list 66 permit 192.168.26.0 0.0.0.255
R1(config)#
R1(config)# route-map MED-65004 permit 10
R1(config-route-map)# match ip address 66
R1(config-route-map)# set metric 100
R1(config-route-map)#
R1(config-route-map)# route-map MED-65004 permit 100
R1(config-route-map)# set metric 200
R1(config-route-map)# exit
R1(config)#
```

Setting the MED with Route Maps Example



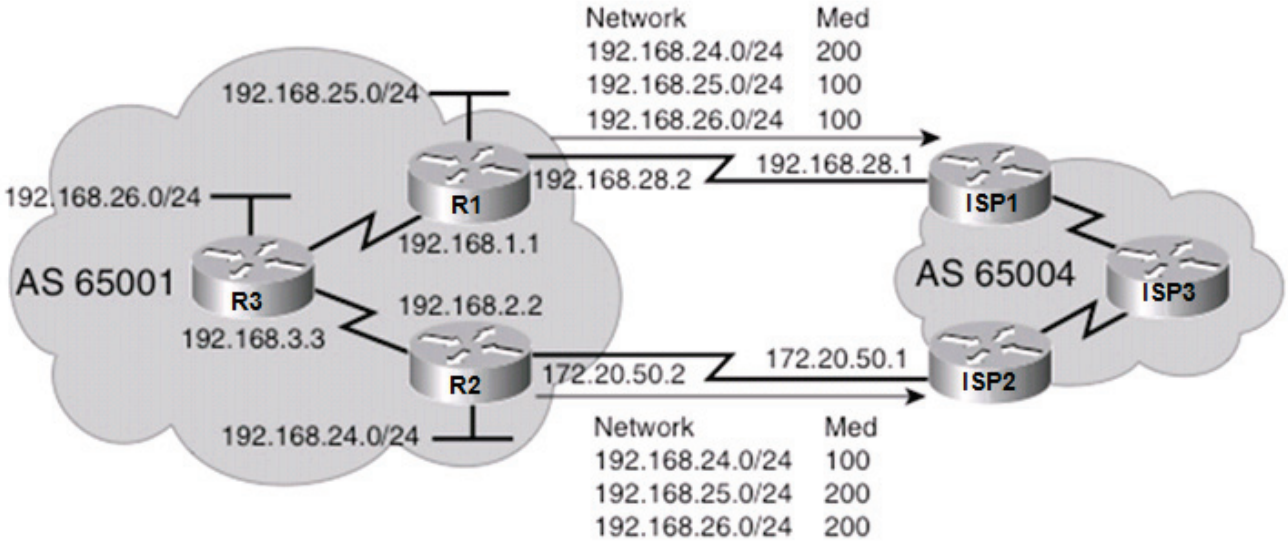

```
R1(config)# router bgp 65001
R1(config-router)# neighbor 192.168.2.2 remote-as 65001
R1(config-router)# neighbor 192.168.2.2 update-source loopback0
R1(config-router)# neighbor 192.168.3.3 remote-as 65001
R1(config-router)# neighbor 192.168.3.3 update-source loopback0
R1(config-router)# neighbor 192.168.28.1 remote-as 65004
R1(config-router)# neighbor 192.168.28.1 route-map MED-65004 out
R1(config-router)#exit
```

Setting the MED with Route Maps Example



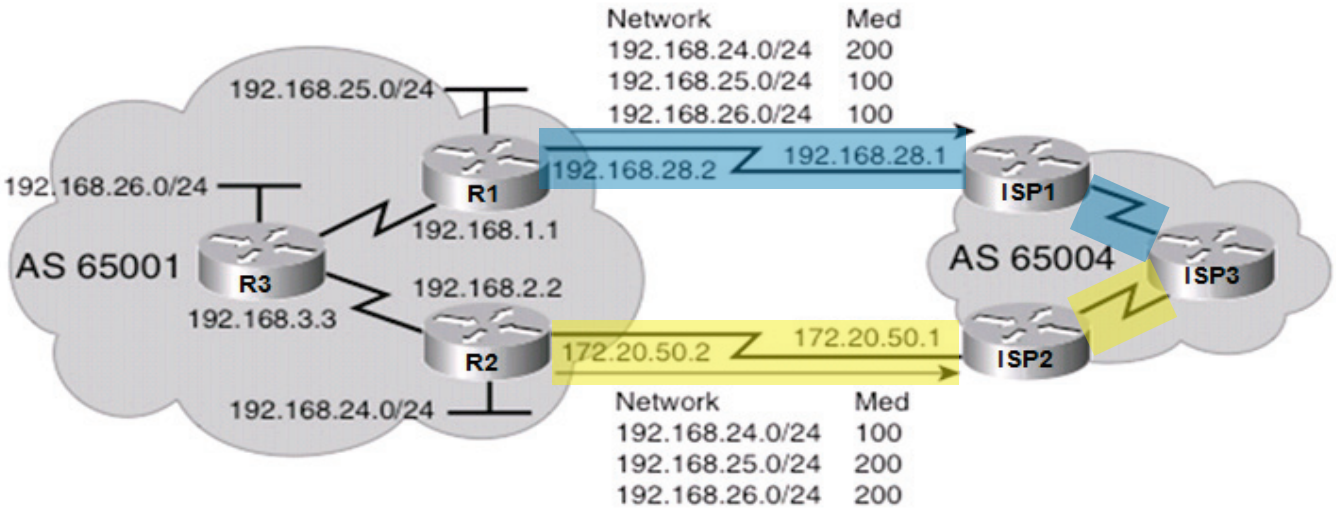

```
R2(config)# access-list 66 permit 192.168.24.0 0.0.0.255
R2(config)#
R2(config)# route-map MED-65004 permit 10
R2(config-route-map)# match ip address 66
R2(config-route-map)# set metric 100
R2(config-route-map)#
R2(config-route-map)# route-map MED-65004 permit 100
R2(config-route-map)# set metric 200
R2(config-route-map)# exit
R2(config)#
```

Setting the MED with Route Maps Example




```
R2(config)# router bgp 65001
R2(config-router)# neighbor 192.168.1.1 remote-as 65001
R2(config-router)# neighbor 192.168.1.1 update-source loopback0
R2(config-router)# neighbor 192.168.3.3 remote-as 65001
R2(config-router)# neighbor 192.168.3.3 update-source loopback0
R2(config-router)# neighbor 172.20.50.1 remote-as 65004
R2(config-router)# neighbor 172.20.50.1 route-map MED-65004 out
R2(config-router)# exit
R2(config)#
```

Setting the MED with Route Maps Example




```
ISP3# show ip bgp
BGP table version is 7, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
   internal,  r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*>i192.168.24.0     172.20.50.2              100     100         0 65001 i
* i                  192.168.28.2             200     100         0 65001 i
* i192.168.25.0     172.20.50.2              200     100         0 65001 i
*>i                  192.168.28.2             100     100         0 65001 i
* i192.168.26.0     172.20.50.2              200     100         0 65001 i
*>i                  192.168.28.2             100     100         0 65001 i
```

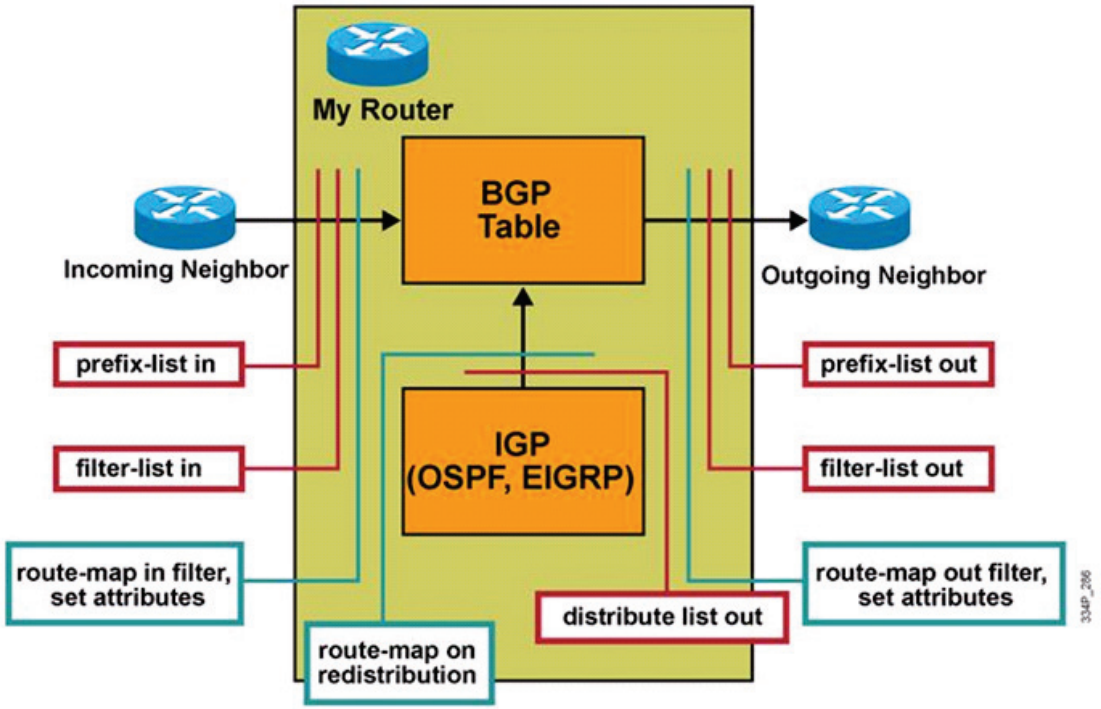
Filtering BGP Routing Updates

Filtering BGP Routing Updates

- BGP can potentially receive a high number of routing updates.
  - To optimize BGP configuration, route filtering may be applied.
- Filtering includes:
  - Filter lists
  - Prefix lists
  - Route maps

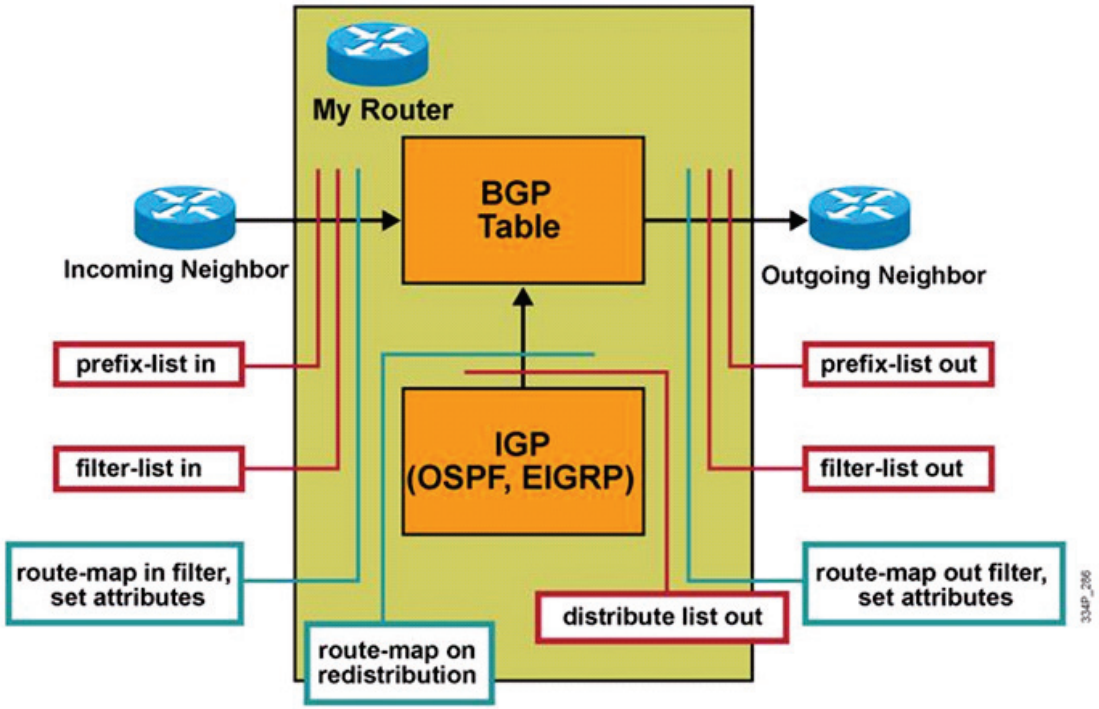
Filtering BGP Routing Updates

- Incoming routes are subject to prefix lists, filter-lists, and route maps before they will be accepted into the BGP table.
  - Similarly, outgoing routes must pass the outgoing route-maps, filter list, and prefix list before they will be transmitted to the neighbor.



Filtering BGP Routing Updates

- If redistributing from an IGP into BGP, the routes must successfully pass any prefix list or route map applied to the redistribution process before the route is injected into the BGP table.



Apply a BGP Filter To Routes

- Apply a filter list to routes from or to a neighbor.

```
Router(config-router)#
neighbor {ip-address | peer-group-name} filter-list
  access-list-number {in | out}
```

Parameter	Description
ip-address	IP address of the BGP neighbor.
peer-group-name	Name of a BGP peer group.
access-list-number	Number of an AS-path access list.
in	Access list is applied to incoming routes.
out	Access list is applied to outgoing routes.

Planning BGP Filtering Using Prefix Lists

- When planning BGP filter configuration using prefix lists, the following steps should be documented:
  - Define the traffic filtering requirements, including the following:
    - Filtering updates
    - Controlling redistribution
  - Configure the **`ip prefix-list`** statements.
  - Apply the prefix list to filter inbound or outbound updates using the **`neighbor prefix-list`** router configuration command.

Configure a Prefix List

- Define a prefix list.

```
Router(config)#
ip prefix-list {list-name | list-number} [seq seq-value] {deny |
  permit} network/length [ge ge-value] [le le-value]
```

Parameter	Description
list-name	The name of the prefix list that will be created (it is case sensitive).
list-number	The number of the prefix list that will be created.
seq *seq-value*	A 32-bit sequence number of the **prefix-list** statement. Default sequence numbers are in increments of 5 (5, 10, 15, and so on).
deny | **permit**	The action taken when a match is found.
network **/** *length*	The prefix to be matched and the length of the prefix. The network is a 32-bit address; the length is a decimal number.
ge *ge-value*	(Optional) The range of the prefix length to be matched. The range is assumed to be from *ge-value* to 32 if only the **ge** attribute is specified.
le *le-value*	(Optional) The range of the prefix length to be matched. The range is assumed to be from length to *le-value* if only the **le** attribute is specified.

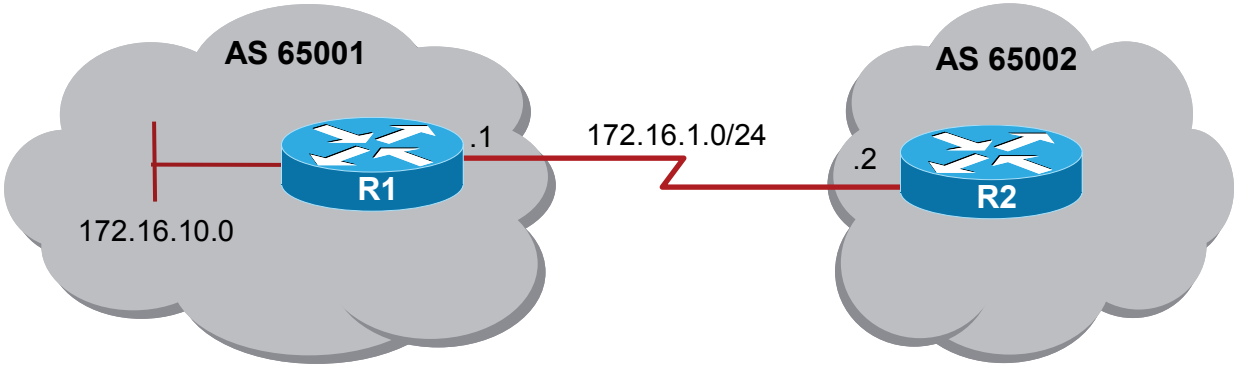
Apply a Prefix List

- Apply a prefix list to routes from or to a neighbor.

```
Router(config-router)#
neighbor {ip-address | peer-group-name} prefix-list prefix-list-name
  {in | out}
```

Parameter	Description
ip-address	IP address of the BGP neighbor.
peer-group-name	Name of a BGP peer group.
prefix-list-name	Name of a prefix list.
in	Prefix list is applied to incoming advertisements.
out	Prefix list is applied to outgoing advertisements.

BGP Filtering Using Prefix Lists Example

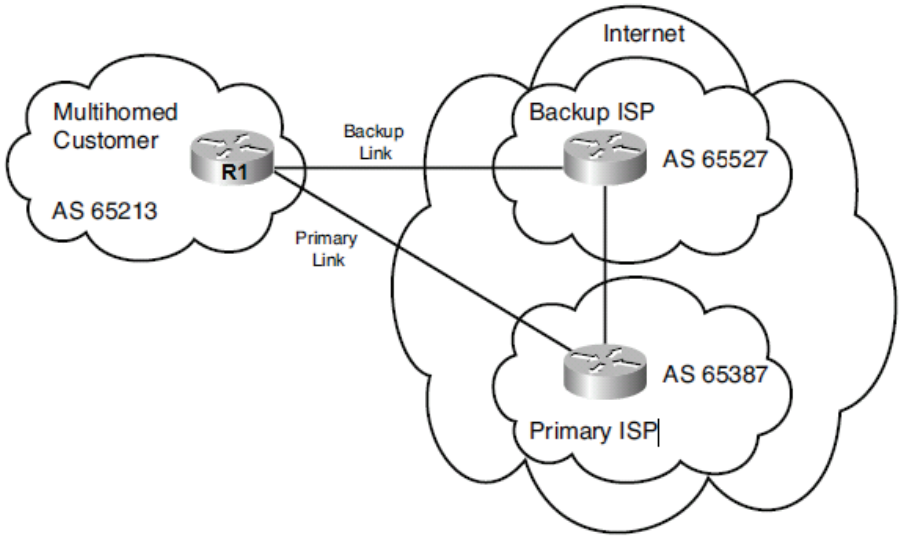



```
R1(config)# ip prefix-list ANY-8to24-NET permit 0.0.0.0/0 ge 8 le 24
R1(config)# router bgp 65001
R1(config-router)# neighbor 172.16.1.2 remote-as 65002
R1(config-router)# neighbor 172.16.1.2 prefix-list ANY-8to24-NET in
R1(config-router)# end
R1#
R1# show ip prefix-list detail ANY-8to24-NET
ip prefix-list ANY-8to24-NET:
Description: test-list
count: 1, range entries: 1, sequences: 10 - 10, refcount: 3
seq 10 permit 0.0.0.0/0 ge 8 le 24 (hit count: 0, refcount: 1)
```

Planning BGP Filtering Using Route Maps

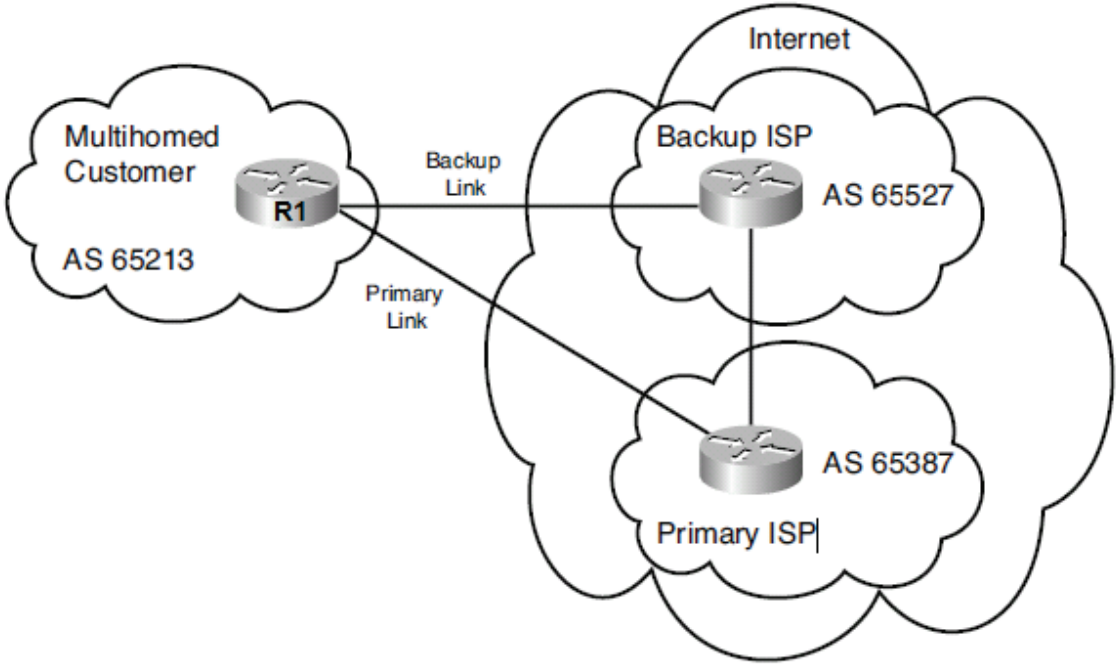
- When planning BGP filter configuration using route maps, the following steps should be documented:
  - Define the route map, including:
    - The **`match`** statements
    - The **`set`** statements
  - Configure route filtering using the route map.

BGP Filtering Using Route Maps




```
R1(config)# ip as-path access-list 10 permit _65387$
R1(config)# ip prefix-list DEF-ONLY seq 10 permit 0.0.0.0/0
R1(config)#
R1(config)# route-map FILTER permit 10
R1(config-route-map)# match ip address prefix-list DEF-ONLY
R1(config-route-map)# match as-path 10
R1(config-route-map)# set weight 150
R1(config-route-map)#
R1(config-route-map)# route-map FILTER permit 20
R1(config-route-map)# match ip address prefix-list DEF-ONLY
R1(config-route-map)# set weight 100
R1(config-route-map)# exit
```

BGP Filtering Using Route Maps




```
R1(config)# router bgp 65213
R1(config-router)# neighbor 10.2.3.4 remote-as 65527
R1(config-router)# neighbor 10.2.3.4 route-map FILTER in
R1(config-router)# neighbor 10.4.5.6 remote-as 65387
R1(config-router)# neighbor 10.4.5.6 route-map FILTER in
R1(config-router)#
```

Configuring BGP

Yannis Xydas