BGP4 Overview

- Built for reliability, scalability, and control - not speed.
- Is a **path-vector protocol**. Its route to a network consists of a list of ASs on the path to that network.
- An EGP, between ISPs, an enterprise and an ISP, etc. Routing between ASs is called *interdomain routing*
- Uses TCP port 179. BGP peers exchange incremental, triggered route updates and periodic keepalives.
- The administrative distance for eBGP routes is 20. The administrative distance for iBGP routes is 200.

- Routers can run only one instance/process of BGP at a time
- Routers running BGP are called *BGP speakers*; neighbors are *BGP peers*
- Networks not in the local router's routing table will not be sent out in updates
- Routes learned by the BGP process are propagated by default but are often filtered by a routing policy.
- When an update about a network leaves an AS, that ASN is prepended to the list of ASs that have handled that update. When an AS receives an update, it examines the AS list. If it finds its own ASN in that list, the update is discarded (loop prevention/built in split-horizon)
- no auto-summary and no synchronization are defaults in BGP

Because BPG uses TCP, it has reliability, error recovery, and flow control.
Before a BGP speaker can peer with neighbor, neighbor must be statically defined; TCP session-capable, IP reachable. The underlying TCP session can be shown with 'show tcp brief'. BGP peering uses TCP port 179, You also get the buffering and other TCP benefits (resent segments if timer reaches 0, acknowledgement may be delayed up to a 1 second to determine if any data should be sent, etc.)

Autonomous System and ASNs
AS is a group of networks under a common administration. The Internet Assigned Numbers Authority (IANA) outlined the ASN ranges from 0 to 65,535, with 64,512 - 65,534 ASNs to be private. Being private means this ASN connect to only one other ASN (sometimes multiple ASN) and these ASNs can't cause loop by themselves.

ASN range	0	1 - 64,495	64,496 - 64,511	64,512 - 65,534	65,535
Purpose	Reserved	Public ASN	Documentation	Private ASN	Reserved

Public ASNs are assigned by RIPE NCC and registered in RIPE database, private ASNs can be removed on eBGP configuration by using 'neighbor ebgp-neighbor-address remove-private-as' command.

Path Attributes: AS_PATH and routing
- Path attributes (PAs) are factors that allow BGP to select a route over another. By default, no BGP PAs have been set, and BGP uses AS_PATH (autonomous system path) PA when choosing the best route among many routes.
- When a router uses BGP to advertise a route with AS_PATH, it will list ASNs the path will go through, to allow finding the best route with shortest AS_PATH and prevent routing loops.
- A route originated from ASN 3 then flows to ASN 7 then 9 and arrives at ASN 1, it will have an AS_PATH of (3, 7, 9). BGP, by default, chooses the route with the least amount of ASNs (distance vector).
- Looping is prevented by ignoring route updates that contain the current AS's ASN. However, having a duplicated ASN means ASN 3 can't learn a route to the duplicated ASN 3.

Internal and External BGP
The BGP peering relationship is either between routers in the same (iBGP) or different (eBGP) autonomous system. If the ASN configured in a device's **router bgp** and **neighbor** statements match, BGP initiates an internal session (iBGP); otherwise an external session (eBGP). The biggest difference between the way iBGP and eBGP neighbors work is the update of AS_PATH. When advertising to an iBGP peer, no ASN is added- it isn't needed. When advertising to an eBGP peer, it's ASN is added.
There are 2 types of routing:
- Hot-potato routing: traffic exits the AS via the closest exit point.
- Cold-potato routing: traffic exits the AS via the path closest to the destination.

BGP Databases
BGP uses three databases:
- BGP DB, or RIB (Routing Information Base): Networks known by BGP, with their paths PAs **- show ip bgp**
- Routing table: Paths to each network used by the router, and the next hop for each **- show ip route**
- Neighbor database: All configured BGP neighbors. **show ip bgp summary**

BGP States and Message Types
The BGP process has different states, similar to how OSPF operates. There is a formalized acronym for this you probably won't need to know unless you are taking Cisco's exams or something, "Finite State Machine" (FSM). The messages don't have specific names like LSA's do in OSPF, just typical hello, update, etc.

1. Router tries to establish TCP connection with IP address configured in 'neighbor' command at port 179
2. After 3-way handshake, first BGP message sent (Open), with parameters to establish neighborship.
3. If a match, neighborship formed (Established), update messages sent (list of PAs and prefixes)

State	Meaning
Idle	Administratively down or awaiting the next retry attempt. TCP session should be initiated by remote peer. Listening for a connection.
Connect	TCP handshake. If successful, router sends Open message and changes to OpenSent. If not, router resets the ConnectRetry timer and goes to Active state. If timer reaches 0 while the router is in Connect state, timer is reset and another attempt is made, remaining in Connect.
Active	TCP connection has been completed, but no BGP messages sent to the peer yet. It can also mean Connect wasn't successful on it's first attempt so it's trying to initiate another session. If ConnectRetry timer expires, it kicks back to Connect state. If peering stops at this stage, it means TCP session attempts have been made from an unexpected IP address, which are rejected. If that happens, it remains in Active and resets ConnectRetry.
OpenSent/ OpenReceive	TCP connection exists, an Open message has been sent to peer. Transitions to OpenReceive to wait for initial keepalive from peer to move into OpenConfirm state. If TCP disconnect received, router terminates session, resets ConnectRetry timer, and goes back to Active.
OpenConfirm	Open messages finished, initial keepalive received. Sends Keepalives to peer to see configs match, and listens. On receipt of keepalive, moves to Established, else moves to Idle
Established	Achieved after receiving more Keepalives from peer. All neighbor parameters match, the neighbor relationship works, and the peers can now exchange Update messages.

To check state, use **sh ip bgp summary**. The State/PfxRcd column shows a numeric value for a state of Established- if not in an Established state, the value in this heading lists the text name of the current BGP state

All BGP messages share the same header, composed of:
- 16 byte marker field: set to all 1s to detect a loss of synchronization.
- 2 byte length field: indicate total length of BGP message, range from 19 to 4096
- There is a PA field in Update (AS Path Attribute)
- 1 byte type field: type code, indicate different BGP messages - the number in () below0

Message (type#)	Purpose
Open (1)	First message after TCP connection has established to get neighborship started. BGP version, ASN, hold timer, BGP identifier, optional authentication
Keepalive (4)	Maintain peering. Contains only header. Keepalives exchanged every 60 sec by default. By default, hold timer is 3x keepalive interval. Updates can also reset keepalive interval
Update (2)	Used to exchange PAs and the associated prefix/length (NLRI) that uses those attributes. Each contains a single set of PAs and all related NLRI. Also contain withdrawn routes.
Notification (3)	Signals error with code, subcode, and data. Often signals to resets connection

All prefixes, except filtered by **neighbor <ipaddr> route-map <name> in**, are in the routing table for calculation.

Show ip bgp	List entire BGP routing table - AS_PATH PA with the last ASN; Internal is 'i'. '*' is valid, and '>' is best route
show ip bgp 0.0.0.0 0.0.0.0	List possible default routes.
show ip bgp *prefix [subnet-mask]*	List possible routes, per prefix.
show ip bgp neighbors	All possible routes, session status for each neighbor
show ip bgp neighbors *ip- addr* **received-routes**	Routes learned from one neighbor, before filtering
show ip bgp neighbors *ip- addr* **routes**	Routes learned from neighbor that passed filters.
show ip bgp neighbors *ip- addr* **advertised-routes**	Routes advertised to neighbor after outbound filtering.
show ip bgp summary	Number of prefixes learned per neighbor. Session state

To verify neighborship **show ip bgp neighbors**, and **show tcp brief** to display the TCP connection
debug ip bgp, debug ip bgp events or **debug ip bgp ipv4 unicast** (IOS v12.4)
Use **neighbor <ipaddr> shutdown** to shut down that connection, move to Idle (keeps neighbor config).

Some common failure causes include
- AS number misconfiguration,
- neighbor IP address misconfiguration,
- a neighbor with no neighbor statement for your router, and
- a neighbor with no route to the source address of your router's BGP messages.

eBGP neighborship
To configure BGP, you need at least 2 commands:
config)#router bgp <asn>
config-router)#neighbor <ipaddr> remote-as <remote-asn>

For BGP neighbor relationship to form,
- TCP connection between them must first be established
- Both routers need complimentary **neighbor <ipaddr> remote-as <asn>** with matching ASN and the IP address of the interface for which TCP packets will exit (if the interface is not explicitly defined)
- BGP RID must not be the same
- MD5 authentication must pass (if configured)

BGP RID is established by
1) **bgp router-id <rid>** - OR -
2) highest IP address of any up/up loopback interfaces at BGP initialization
3) highest IP address of any up/up normal interface at BGP initialization

To configure authentication, use **neighbor <ipaddr> password <password>** on both routers. The <ipaddr> refers to the IP address of the other router, while password must be identical for neighborship to succeed.

To verify neighborship **show ip bgp neighbors**, and **show tcp brief** to display the TCP connection
Use **neighbor <ipaddr> shutdown** to shut down that connection, move to Idle (keeps neighbor config).

The 'neighbor' command can be configured with the **peer-group** parameter, so one set of commands can be applied to all neighbors in the peer-group (either external or internal, not both)
Simplify configuration and reduce updates.
neighbor <name> peer-group
neighbor <ipaddr> peer-group <name>
(neighbor must have 'neighbor remote-as' set).

In most IGPs, the network command starts the routing process on an interface.
In BGP, the command tells the router to originate an advertisement for that network.
- network does not have to be connected - it just has to be in the routing table.
- can even be a network in a different AS (not usually recommended, but next-hop-self sort of does it)

BGP assumes you are using the default classful subnet mask.
- need to advertise a subnet? Use the optional keyword **mask** and specify the subnet mask to use
- routing table must contain an exact match (prefix and subnet mask)

Network declarations need to be SPECIFIC
If you misconfigure a **network** command, like **network 192.168.1.1 mask 255.255.255.0**, BGP will look for exactly 192.168.1.1/24 in the routing table. With no match, BGP won't announce it to neighbors.

If you issue the command **network 192.168.0.0 mask 255.255.0.0** to advertise a CIDR block, BGP will look for ONLY 192.168.0.0/16 in the routing table. It may find 192.168.1.0/24 or 192.168.1.1/32; however, it may never find 192.168.0.0/16. In this case, you can configure a static route towards a null interface so BGP can find an exact match in the routing table: **ip route 192.168.0.0 255.255.0.0 null0**

BGP and Loopback Addresses

- Recall that a loopback interface provides a more robust configuration - it never goes down, so it adds more stability than physical the ones, and BGP will still operate if the link to the closest interface fails.

If the physically connected port is down:

- The router can still send the packet without being interrupted because loopback is always up.
- If another interface tries to reach the neighbor it will be blocked when source address doesn't match.

When multiple links exist between 2 BGP routers, and you would like to establish BGP neighborship between them, there are 2 options:

- configure a connection for each physical interface; consumes bandwidth and memory.
- Configure connections using virtual (loopback) interfaces, which requires less bandwidth and memory, and ensure the interface is always up/up.

eBGP assumes that neighbors are directly connected and peering with the IP of that specific interface. If not, you must use **neighbor *ip-address* ebgp-multihop *number-of-hops*** command. If you are peering with loopback interface IP addresses, you are going to have to use it.

iBGP assumes that internal neighbors might not be directly connected, so this command is not needed. With loopback IP addresses in iBGP, you must change the source to match the loopback with **neighbor** *ip-address* **update-source** *interface*

Router(config-router)#**neighbor 172.16.1.2 update-source loopback0**

- Without neighbor **update-source**, iBGP will use the closest IP interface to the peer. In the case of a point-to-point eBGP session, this command is not needed because there is only one path for BGP to use.

Using loopback and update-source for connections whether iBGP or eBGP is considered a best practice. It ensures an alternate route to the specific router is available if the physical interface goes down

BGP - Basic Configuration

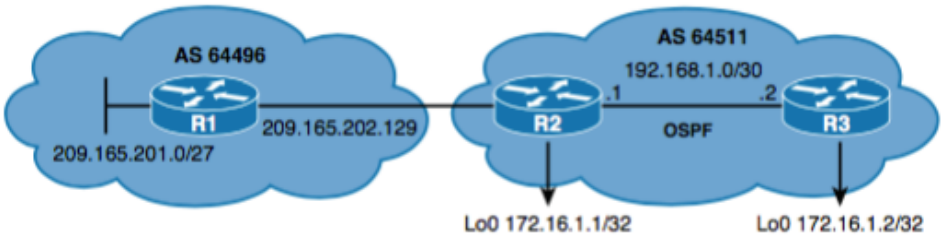
router bgp 100.	Starts BGP routing process 100
neighbor 192.31.7.1 remote-as 200	Add a peer for sessions. The ASN tells if eBGP or iBGP
network 192.135.250.0	What locally learned networks to advertise.
network 128.107.0.0 mask 255.255.255.0	Specify an individual subnet.
(**no) neighbor 24.1.1.2 shutdown**"	Disable/-enable an active session
timers bgp 90 240	Timers. Keepalive (default 60 sec) Holdtime (default 180 sec)
neighbor 172.16.1.2 update-source loopback0	Use this specific interface
(**no**) **synchronizatio**n	iBGP route must be added to table by other IGP before used

Sychronization is off by default in later IOS. Using iBGP meshes is preferred. Redistribution from BGP into an IGP when using BGP for MPLS is reasonable and commonly done. See SWITCH book pg 617 on avoiding loops

iBGP Next-Hop Behavior

- The eBGP next-hop attribute is the IP address that is used to reach the advertising router; an edge router belonging to the autonomous system "next door", in most cases, the IP address of the connection between peers.
- For iBGP, the eBGP next-hop address is not changed, carried into the local autonomous system as-is.

The next-hop-self command is a way to get around this (illustrated below) - R2 tells R3 that it instead will be the next-hop address when trying to reach networks outside its autonomous system.

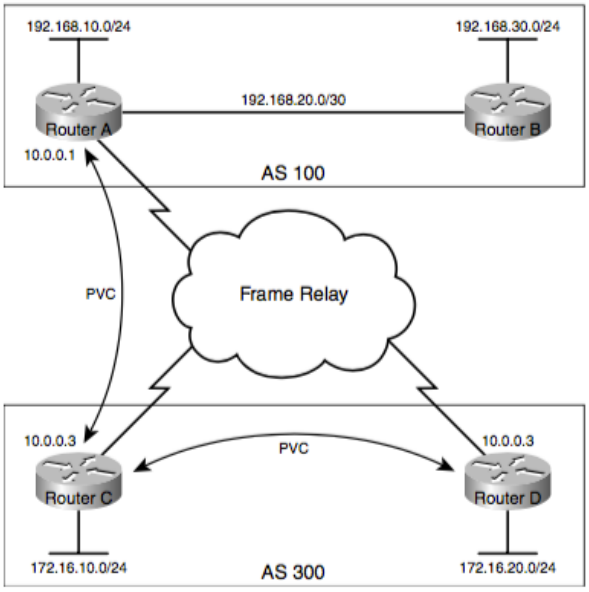


Setup on R2:

router bgp 64511	Starts the BGP routing process.
neighbor 209.165.202.129 remote-as 64496	Identifies R1 as an eBGP neighbor
neighbor 172.16.1.2 remote-as 64511	Identifies R3 as an iBGP neighbor
neighbor 172.16.1.2 update-source loopback0	Loopback0 IP is source for all BGP TCP packets to R3
neighbor 172.16.1.2 next-hop-self	Advertises itself as next hop for networks from other AS

R3 will then use R2 as the next hop instead of using the eBGP next-hop of 209.165.202.129.

Here is another example in a slightly different setup where it is almost mandatory:



RouterC(config)#**router bgp 300**
RouterC(config-router)#**neighbor 10.0.0.1 remote-as 100**
RouterC(config-router)#**neighbor 10.0.0.1 next-hop-self**

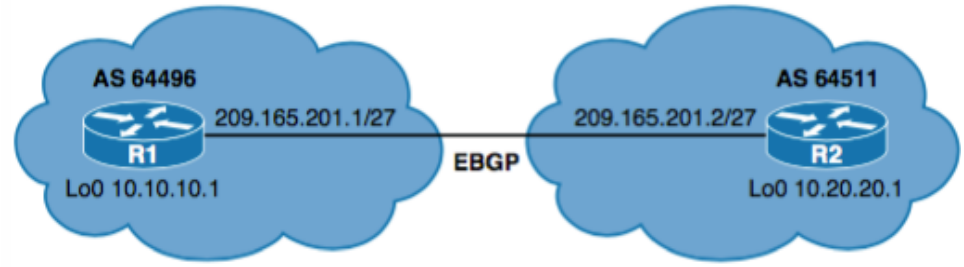
- Forces all updates destined to the neighbor at 10.0.0.1 to advertise this router as the next hop.

- Router C advertises 172.16.20.0 to Router A with next hop of 10.0.0.3 as if the common media were Ethernet.
- Routing will fail - Router A has no direct PVC to Router D and cannot reach the next hop
- To remedy this situation, neighbor next-hop-self causes Router C to advertise 172.16.20.0 with the next-hop attribute set to it's 10.0.0.3.
- This proves useful in non-meshed networks (such as Frame Relay or X.25) where BGP neighbors might not have direct access to all other neighbors on the same IP subnet.

Related
RouterC(config-router)#**neighbor 10.0.0.1 next-hop-unchanged**
Enables an eBGP multihop peer to propagate the next hop unchanged.

Do not use with route reflector setups
This command should not be configured on a route reflector, and the neighbor next-hop-self command should not be used to modify the next-hop attribute for a route reflector when this feature is enabled for a route reflector client.

eBGP Multihop
By default, eBGP neighbors exchange packets with a TTL set to 1. Even though eBGP neighbors are usually directly connected (over a WAN connection) to establish sessions, sometimes one of the directly connected routers is unable to run BGP. The command **ebgp-multihop** allows for a logical connection to be made between peer routers, even if not directly connected - up to 255 hops away and still be able to create an eBGP session. The **ebgp-multihop** is only used for eBGP sessions, and must be configured on each peer (each end). If you attempt to establish eBGP session between loopbacks, BGP packets will be dropped due to an expired TTL.



R1(config)# **ip route 10.20.20.1 255.255.255.255 209.165.201.2**

- Define a static route to Loopback0 on R2.

R1(config)# **router bgp 64496**
R1(config-router)# **neighbor 10.20.20.1 remote-as 64511**

- Identifies a peer router at 10.20.20.1

R1(config-router)# **neighbor 10.20.20.1 update-source loopback0**
R1(config-router)# **neighbor 10.20.20.1 ebgp-multihop 2**
- Allows for two routers (not directly connected) to establish an eBGP session with TTL of 2.

R2(config)# **ip route 10.10.10.1 255.255.255.255 209.165.201.1**
- Define a static route to Loopback0 on R1.

R2(config)# **router bgp 64511**
R2(config-router)# **neighbor 10.10.10.1 remote-as 64496**
R2(config-router)# **neighbor 10.10.10.1 update-souce loopback0**
R2(config-router)# **neighbor 10.10.10.1 ebgp-multihop 2**
- Allows for two routers (not directly connected) to establish an eBGP session with TTL of 2.

Loopbacks on eBGP need ebgp-multihop
If redundant links exist between two eBGP neighbors and loopback addresses are used, you must configure **ebgp-multihop** because of the default TTL of 1. Otherwise, the router decrements the TTL before giving the packet to the loopback interface, meaning that the normal IP forwarding logic discards the packet. Configuring the value to 2 solves the problem.

Default Routes, default-originate to give default to specific neighbors
For a default route to send to all neighbors/peers you just use **network 0.0.0.0** (be sure it's in the routing table)
If you want the default route of 0.0.0.0 to only be advertised to a specific neighbor, use the command **neighbor 192.168.100.1 default-originate**

Verifying BGP Connections

show ip bgp	Displays entries in the BGP table
show ip bgp <ipsubnet/cidr> **longer-prefixes**	Show which route is older by placing its entry later
show ip bgp neighbors	Info about BGP/ TCP connections to neighbors, neighbor info
show ip bgp neighbors <address> **[received | routes | advertised]**	Monitor routes received/ learned from a specific neighbor.
show ip bgp rib-failure	List networks not put into the Routing Information Base (RIB) and the reason not used
show ip bgp summary	Status of all BGP connections, memory use of BGP databases, activity stats and list of BGP neighbors
show ip route bgp	Displays the BGP entries from the routing table

show ip bgp [peer-group | regexp | community-list]
show access-lists and **show ip access-lists**

Troubleshooting BGP Connections
clear ip bgp * - Forces BGP to clear its table and resets all BGP sessions.
clear ip bgp 10.1.1.1 - Resets the specific BGP session with the neighbor at 10.1.1.1.
clear ip bgp 10.1.1.2 soft out - Forces the remote router to resend all BGP info to neighbor **without** resetting the connection. Routes from this neighbor are not lost. Works when you are changing outbound policy, does not help if you are changing an inbound policy. The **soft** keyword is optional; **clear ip bgp out** will do a soft reset for all outbound updates.

neighbor 10.1.1.2 soft-reconfiguration inbound Causes the router to store all updates from this neighbor in case the inbound policy is changed. This is memory intensive!

clear ip bgp 10.1.1.2 soft in - Uses the stored information to generate new inbound updates.
clear ip bgp {*|10.1.1.2} [soft in | in] - Creates a dynamic soft reset of inbound BGP routing table updates. Routes are not withdrawn. Updates are not stored locally. The connection remains established. See the note that follows for more information on when this command can be used.

Hard reset can be harsh - it zaps the underlying TCP connection, brings down the neighborship, removes all BGP table entries learned from that neighbor. It takes a long time to complete and interrupts routing in the interim, flapping where peers disassociate themselves, force a full set of routing updates generating a ton of traffic.

Soft resets are more acceptable: the router doesn't bring down the neighborship or TCP connection, and just resends adjusted outgoing updates, which then adjust the BGP table. IOS 12.0.2, put in a soft reset enhancement feature known as *route refresh*. It's not dependent on stored routing table update information; requires no preconfiguration and requires less memory than previous soft methods for inbound routing table updates. To check, use **show ip bgp neighbors** - "Received route refresh capability from peer" means route refresh is supported:

When a BGP session is reset and soft reconfiguration is used, several commands enable you to monitor BGP routes that are received, sent, or filtered:
show ip bgp
show ip bgp neighbor <address> [received | routes | advertised]

Router#**debug ip bgp**	Displays information related to processing BGP
Router#**debug ip bgp updates**	Displays information about the processing of BGP update

Using the form **clear ip bgp *** is both processor and memory intensive - use only in smaller environments.
Try to apply to a specific network or session with a neighbor with the **clear ip bgp** *specific-network*
However, you may need to use this form when the following changes occur:
- Additions or changes to the BGP-related access lists
- Changes to BGP-related weights
- Changes to BGP-related distribution lists
- Changes in the BGP timer's specifications
- Changes to the BGP administrative distance
- Changes to BGP-related route maps

Contents of show ip bgp - A Deeper Look

The command **show ip bgp** shows the BGP topology database, which includes all the networks BGP knows about, the next hop, some of the attributes, and the AS path for each route.

```
route-server>show ip bgp
BGP table version is 22285573, local router ID is 12.0.1.28
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
* 3.0.0.0     12.123.137.124                 0     7018 2914 9304 80 i
*>            12.123.1.236                   0     7018 2914 9304 80 i
* 3.51.92.0/23 12.123.137.124                0     7018 ?
*             12.122.125.4     2366          0     7018 ?
*>            12.123.1.236                   0     7018 ?
* 8.6.6.0/24  12.123.137.124                 0     7018 701 14744 14744 14276 i
*             12.123.145.124                 0     7018 701 14744 14744 14276 i
*>            12.123.1.236                   0     7018 701 14744 14744 14276 i
```

Networks are listed in numerical order.
The first three columns list each route's status. They are one-character fields, squeezed together

The asterisk (*) in the first column means that the route has a valid next hop.
Other options:
- s - suppressed: BGP knows about the network but is not advertising it, often if part of a summarized route.
- d - dampened: BGP stopped advertising a network that flaps too often until it is stable for a period of time.
- h - history: BGP knows about this network but does not currently have a valid route to it.
- r - RIB failure: advertised route but not put in routing table. Another protocol may have route with a better AD.
- S - stale: Used with nonstop forwarding - route needs to be refreshed when the peer is reestablished.

The second column has a greater-than sign (>) beside the route that was selected as the best path to that network. In the example, the second route was selected for network 3.0.0.0.
The third column is blank in the example, which means that the router learned all the routes from an external neighbor. A route learned from an IBGP neighbor would have an "I" in the third column. (a - aggregate, i - internal)

The fourth column lists the networks. Those without a subnet mask, such as network 3.0.0.0, use their classful mask. As seen in the example, when the router learns about the same network from multiple sources, it lists only the network once.

The fifth column lists the next-hop address for each route. This might or might not be a directly connected router. A next-hop of 0.0.0.0 means that the local router originated the route (and/or, the router has non-BGP routes to the network).

Inter-autonomous system metric - If a Med value was received with the route, it is listed in the Metric column. Notice that the advertisement for network 3.51.92.0/23 from the router at 12.122.125.4 has a large Med value of 2366. Because the default Local Preference is used for each of the routes shown, no local preference value is displayed. The default Weight value of 0 is listed, however.

Toward the end is the AS path for each network. Reading this field from left to right, the first AS number shown is the adjacent AS this router learned the route from. After that, the AS paths that this route traversed are shown in order. The last AS number listed is the originating AS. In the example, our router received an advertisement about network 3.0.0.0 from its neighbor AS 7018, which heard about it from AS 2914, which heard about it from AS 9304. And AS 9304 learned the route from AS 80, which originated it. A blank AS path means that the route was originated in the local AS.

NOTE
In the AS Path column, note that network 8.6.6.0 shows AS 14744 twice in its AS path list. Most likely AS 14744 has prepended an extra copy of its AS number to make the path through it less attractive than the path through other autonomous systems. In this case it did not work because the only paths to 8.6.6.0 this router knows about all go through AS 14744.

The last column shows how BGP originally learned about the route.
- Networks 3.0.0.0 and 8.6.6.0 show an “i” for their origin codes. This means that the originating router had a network statement for that route.
- Network 3.51.92.0 shows a “?” as its origin. This means that the route was redistributed into BGP; BGP considers it an “incomplete” route.
- You will likely never see the third possibility, an “e,” because that means BGP learned the route from the Exterior Gateway Protocol (EGP), which is no longer in use.

If you specify an address this is how it will display:

router#**show ip bgp 66.66.66.66**

```
BGP routing table entry for 66.66.66.66/32
Paths: (2 available, best #1)
  66.66.66.66/32
  3 i comm 65535:65281
  172.16.23.3 from 172.16.23.3 (peer 3.3.3.3)
    Origin IGP, local pref 100, weight 0, valid, best
     BGP Path Selection Criterion: Lowest BGP Neighbor Router-ID
    IGP Metric: 0 IGP Pref: 0 IGP Protocol: DIRECT
    IGP Next Hop: 0.0.0.0 Route Age: 0:03:22
  66.66.66.66/32 (Second best)
  77 i comm 65535:65281
  172.16.12.1 from 172.16.12.1 (peer 172.16.12.1)
    Origin IGP, local pref 100, weight 0, valid
    IGP Metric: 0 IGP Pref: 0 IGP Protocol: DIRECT
    IGP Next Hop: 0.0.0.0 Route Age: 0:02:21
```

IGP Metric	Specifies the IS-IS metric or OSPF cost value.
LocPrf	Local preference value. See the set local-preference route map configuration command. Default value is 100.
Weight	Weight of the route as defined by set weight and router bgp route map commands.
IGP Protocol	IGP Protocol: IS-IS or OSPF.
Originator	Specifies the router ID of the originator of the route in the local AS.
Cluster list	A sequence of cluster ID values for the reflection path that the route has passed.
Path	Autonomous system path to the destination network. One entry in this field per autonomous system in the path.
IGP Next-Hop	IP address of the next system used when forwarding packets to the destination network. 0.0.0.0 in this field indicates router has non-BGP routes to the network.
Origin codes	Identifies the origin of the entry. Valid values previously mentioned (i, e, ?)
Origin	Identifies the origin of the entry as one of the following: IGP, EGP, Not clear, Best.
Route Age	Specifies the time in *hours*:*minutes*:*seconds* that a route has been valid.
BGP path selection criteria	Displays tie-breaking criterion for best path selection.
Second best	Displays second best path information.

BGP Path Attributes - Manipulating Path Selection

Routes learned via BGP have properties referred to as *BGP attributes*, and an understanding of how they influence route selection is required for the design of robust networks.

PA	Description	Route Direction
NEXT_HOP	Next-hop IP address used to reach a prefix.	N/A
Weight[1]	Range 0 - 2^16 -1, set when receiving updates. Not advertised to any BGP peers. Cisco proprietary	Outbound
LOCAL_PREF	Range 0 - 2^32 -1, set and spread throughout an AS to influence the choice of best route for all routers in that AS	Outbound
AS_PATH (length)	The number of ASNs in the AS_Path PA.	Outbound, Inbound
ORIGIN	Value implying the route was injected into BGP; I (IGP), E (EGP), or ? (incomplete information).	Outbound
Multi-Exit Discrim (MED)	Set and advertised by routers in an AS, impacting the BGP decision of routers in the other AS. Smaller is better.	Inbound

Four categories of attributes exist as follows:
- Well-known mandatory: Must be recognized by all BGP routers, present in all BGP updates, and passed on to other BGP routers. For example, **AS path, origin, and next hop.**
- Well-known discretionary: Must be recognized by all BGP routers and passed on to other BGP routers but need not be present in an update, for example, **local preference.**
- Optional transitive: Might or might not be recognized by a BGP router but is passed on to other BGP routers. If not recognized, it is marked as partial, for example, **aggregator, community.**
- Optional nontransitive: Might or might not be recognized by a BGP router and is not passed on to other routers, for example, **Multi-Exit Discriminator (MED), originator ID.**

BGP Path Selection Criteria
1. Highest weight (most local).
2. Highest LOCAL_PREF (inside an AS this is global).
3. Choose routes that this router originated (next-hop = 0.0.0.0)
4. Shortest AS_PATH
5. Lowest origin type/code - (i)iBGP is lowest, (e)eBGP is next, and ? is last
6. Lowest MED (if the same AS advertises the possible routes)
7. eBGP wins over an iBGP route.
8. Nearest IGP neighbor (lowest IGP metric)
9. Oldest route (for eBGP to minimize the effects of route flapping)
10. Lowest router ID.
11. Lowest IP address.

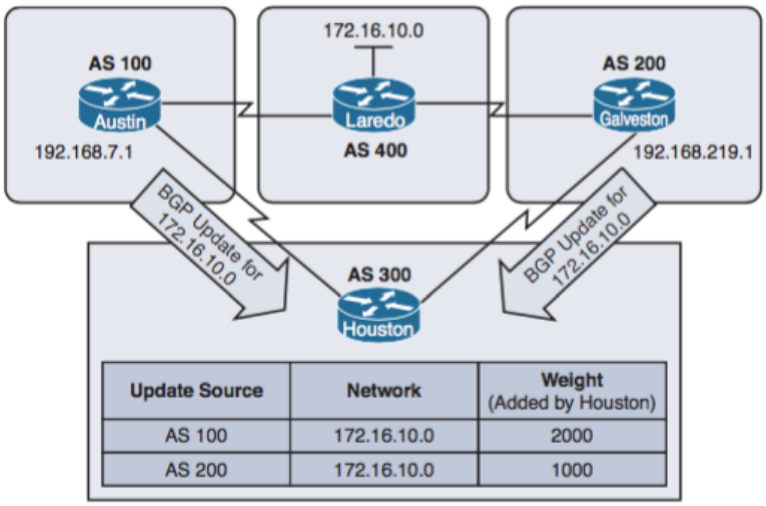
BGP only chooses one route as the best route (thus, no load balancing). Step 1, 2, and 4 are typically used to influence outbound routes, while MED for outbound routes. 9, 10 and 11 aren't used except in a tie
Mnemonic: N WLLA OMNI

Step	Mnemonic Letter	Short Phrase	Which Is Better?
0	N	Next hop: reachable?	If no route to reach Next_Hop, router cannot use this route.
1	W	Weight	Bigger weight.
2	L	LOCAL_PREF	Larger preference.
3	L	Locally injected routes	Locally injected ‘network’ is better than iBGP/eBGP learned.
4	A	AS_PATH length	Smaller paths win
5	O	ORIGIN	Prefer I over E. Prefer E over ?
6	M	MED	Smaller Multi-Exit Discriminator (MED)
7	N	Neighbor Type	Prefer eBGP over iBGP.
8	I	IGP metric to Next_Hop	Smaller metric wins. If no IGP is used, consider tied.
9	A	Age equals seniority	Oldest eBGP route
10	O	think not OSPF	Route with lowest BGP RID
11	S	think not OSPF	Route with lowest neighbor IP address

Weight Attribute - not a BGP PA! Cisco proprietary

The weight attribute is a proprietary Cisco attribute that is used in the path selection process when there is more than one route to the same destination. It's configured locally on a router and is not propagated to any other routers. This attribute applies when one router is used with multiple exit points out of an autonomous system, as opposed to the local preference attribute, which is used when two or more routers provide multiple exit points.

By default, the weight attribute is 32,768 for paths that the router originates, and 0 for other paths. Routes with a *higher weight are preferred* when there are multiple routes to the same destination.
[Weight value range from 0 to 65,535, 0 for learned routes, and 32,768 for locally injected routes]



Update Source	Network	Weight (Added by Houston)
AS 100	172.16.10.0	2000
AS 200	172.16.10.0	1000

Houston(config)#**router bgp 300**	Starts the BGP routing process
Houston(config-router)#**neighbor 192.168.7.1 remote-as 100**	Identifies a peer router at 192.168.7.1
Houston(config-router)#**neighbor 192.168.7.1 weight 2000**	Sets the weight of all updates from neighbor 192.168.7.1 to 2000
Houston(config-router)#**neighbor 192.168.219.1 remote-as 200**	Identifies a peer router at 192.168.219.1
Houston(config-router)#**neighbor 192.168.219.1 weight 1000**	Sets the weight of all updates from neighbor 192.168.219.1 to 1000

The result of this configuration will have Houston forward traffic to the 172.16.10.0 network through AS 100, because the route entering AS 300 from AS 100 had a higher weight attribute set compared to that same route advertised from AS 200. When a router receives a BGP Update, that router can set the Weight either selectively, per route, using a route map, or for all routes learned from a single neighbor

Local Preference Attribute

- Local preference tells routers inside the AS about the path that is preferred for exiting the AS.
- The **local-preference** value can be between 0 and 429,496,729. Higher is preferred, default is 100.
- It is local to the AS; it is exchanged between iBGP peers - ***not*** *advertised to eBGP peers*.
- Use the **local-preference** attribute to force BGP routers to prefer one exit point over another.
- Updates received from eBGP peers do not include the Local_Pref PA- a null value for Local_Pref for eBGP-learned routes by default (column will be blank). However, Updates from iBGP peers do include Local_Pref
- Recall also that BGP does not allow a router to advertise iBGP-learned routes to iBGP peers
- R1(config-router)#**bgp default local-preference 150**

Houston(config)#router bgp 256	
Houston(config-router)#neighbor 172.16.1.1 remote-as 100	Identifies a peer router at 172.16.1.1.
Houston(config-router)#neighbor 10.1.1.2 remote-as 256	Identifies a peer router at 10.1.1.2.
Houston(config-router)#bgp default local-preference 150	Sets the local preference attribute on this router.
Galveston(config)#router bgp 256	Starts the BGP routing process.
Galveston(config-router)#neighbor 172.17.1.1 remote-as 300	Identifies a peer router at 172.17.1.1.
Galveston(config-router)#neighbor 10.1.1.1 remote-as 256	Identifies a peer router at 10.1.1.1.
Galveston(config-router)#bgp default local-preference 200	Sets the local preference attribute on this router.

Based on these two configurations, traffic destined for a remote network that can be reached through autonomous system 256 will be routed through Galveston.

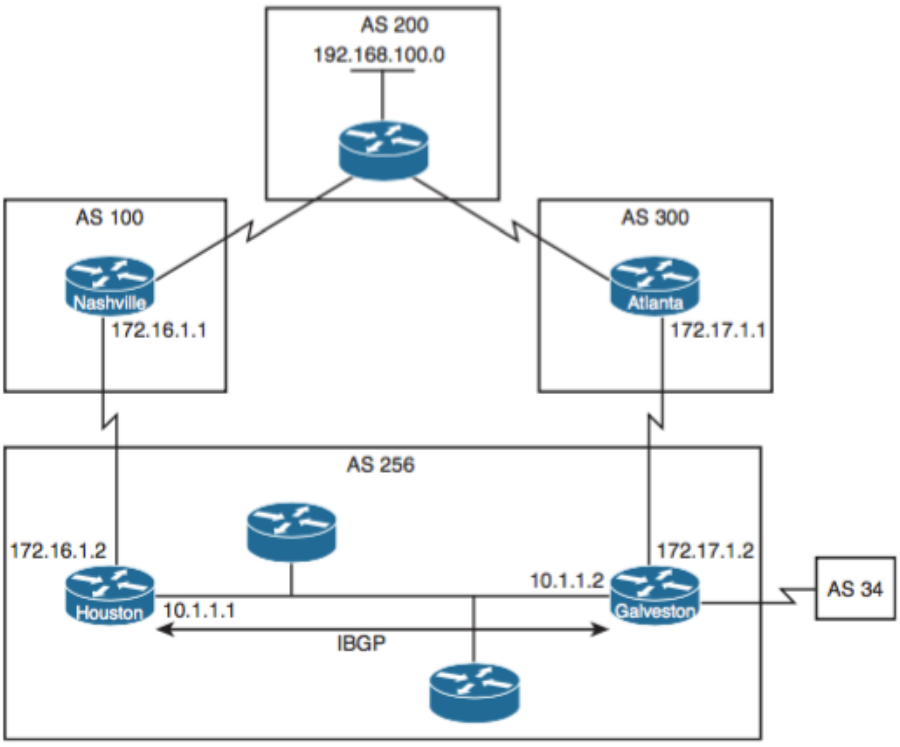


Figure 6-12

Using AS_PATH Access Lists with Route Maps to Manipulate the Local Preference Attribute
Route maps provide more flexibility than the **bgp default local-preference** router configuration command.

Galveston(config)#**router bgp 256**	Starts the BGP routing process.
Galveston(config-router)#**neighbor 172.17.1.1 remote-as 300**	Identifies a peer router at 172.17.1.1.
Galveston(config-router)#**neighbor 172.17.1.1 route-map SETLOCAL in**	Refers to a route map called SETLOCAL.
Galveston(config-router)#**neighbor 10.1.1.1 remote-as 256**	Identifies a peer router at 10.1.1.1.
Galveston(config-router)#**exit**	Returns to global configuration mode.
Galveston(config)#**ip as-path access-list 7 permit ^300$**	Permits updates whose AS_PATH attribute starts and ends with 300 (represented by the ^ and $).
Galveston(config)#**route-map SETLOCAL permit 10**	Creates a route map called SETWEIGHT. A sequence number of 10 is assigned.
Galveston(config-route-map)#**match as-path 7**	Specifies the condition under which policy routing is allowed- matching the BGP ACL 7.
Galveston(config-route-map)#**set local-preference 200**	Assigns a local preference of 200 to any update coming from autonomous system 300 (ACL 7)
Galveston(config-route- map)#**route-map SETLOCAL permit 20**	Second statement of route map SETLOCAL. This instance will accept all other routes.

In the previous example, using the **bgp default local-preference** command on Galveston, the local preference attribute of *all* routing updates received by Galveston would be set to 200. This would include updates from AS 34. In this example, using the **route-map** command, only updates received from AS 300, as specified in the **ip as_path access-list** command, will have a local preference set to 200.
Through the **neighbor route-map** command; **in** option is required for updates from an eBGP peer

[Some BGP features outside the scope of this book actually impact the AS_Path length calculation as well. However, for the purposes of this book, AS_Path length is simply the number of ASNs listed in the AS_Path]

[- increase the length of an AS_Path by adding ASNs to the AS_Path, while not impacting the loop-prevention role of the AS_Path PA. By increasing the length of an AS_Path, a route is less likely to become the best route. By adding ASNs that already exist inside a particular route's AS_Path, the fea- ture does not inadvertently prevent a route from being ignored because of AS_Path loop prevention.]

```
route-map add-two-asns permit 10
 set as-path prepend 3 3
router bgp 11
 neighbor 192.168.1.6 route-map add-two-asns in
```

AS_PATH Attribute Prepending

Autonomous system paths can be manipulated by prepending, or adding, extra autonomous system numbers to the AS_PATH attribute. Assuming that the values of all other attributes are the same, routers will pick the shortest AS_PATH attribute; therefore, prepending numbers to the path will manipulate the decision as to the best path. Normally, AS_PATH prepending is performed on outgoing eBGP updates over the undesired return path.

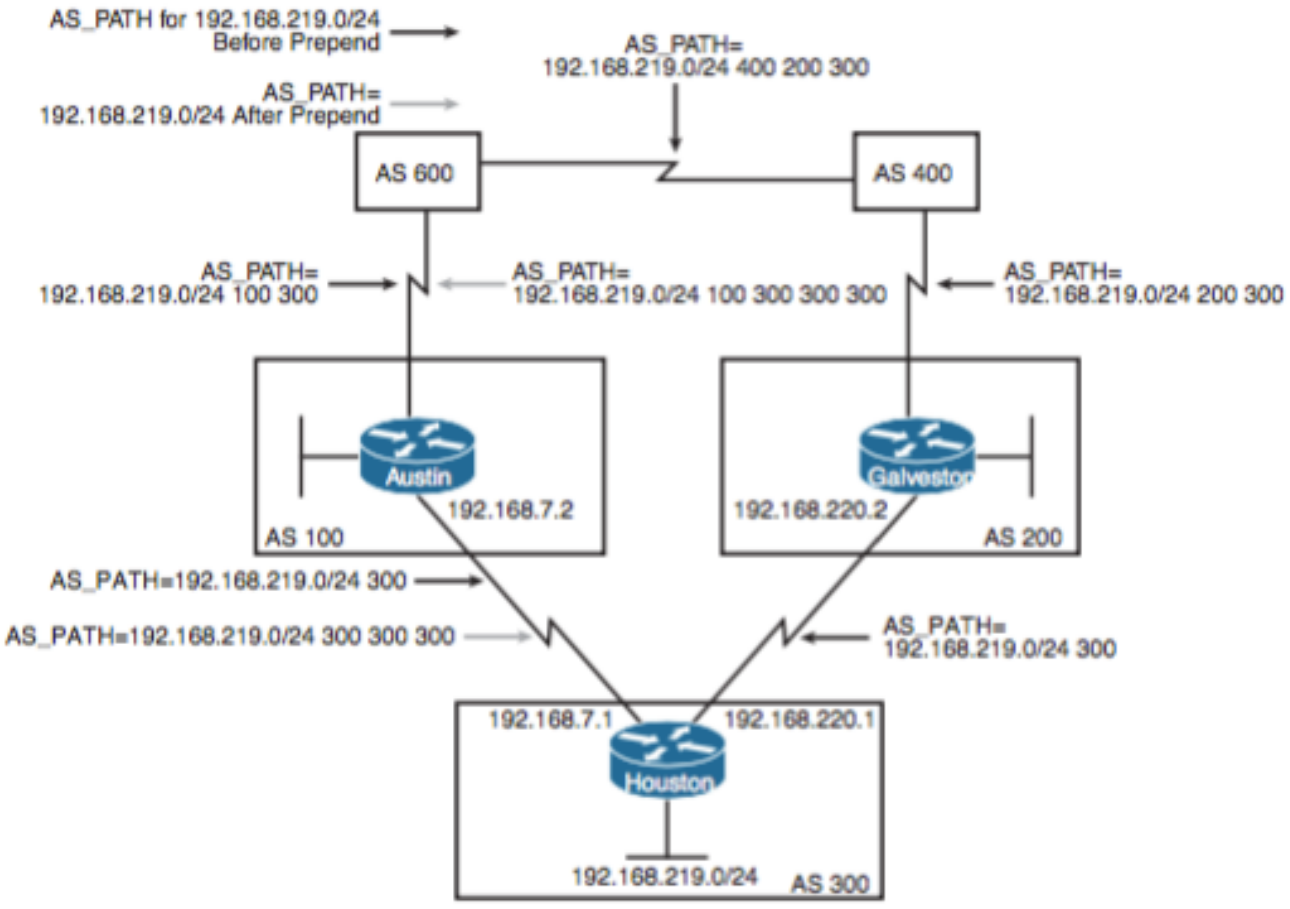


Figure 6-13 AS_PATH Attribute Prepending

In this scenario, you want to use the configuration of Houston to influence the choice of paths in autonomous system 600. Currently, the routers in autonomous system 600 have reachability information to the 192.168.219.0/24 network via two routes: via autono- mous system 100 with an AS_PATH attribute of (100, 300), and via autonomous system 400 with an AS_PATH attribute of (400, 200, 300). Assuming that the values of all other attributes are the same, the routers in autonomous system 600 will pick the short- est AS_PATH attribute: the route through autonomous system 100. You will prepend, or add, extra autonomous system numbers to the

AS_PATH attribute for routes that Hous- ton advertises to autonomous system 100 to have autonomous system 600 select autono- mous system 400 as the preferred path of reaching the 192.168.219.0/24 network.

Houston(config)#**router bgp 300**	Starts the BGP routing process.
Houston(config-router)#**network 192.168.219.0**	Tells the BGP process what locally learned networks to advertise.
Houston(config-router)#**neighbor 192.168.220.2 remote-as 200**	Identifies a peer router at 192.168.220.2.
Houston(config-router)#**neighbor 192.168.7.2 remote-as 100**	Identifies a peer router at 192.168.7.2.
Houston(config-router)#**neighbor 192.168.7.2 route-map SETPATH out**	Read this command to say, "All routes destined for neighbor 192.168.7.2 will have to follow the conditions laid out by the SETPATH route map."
Houston(config-router)#**exit**	Returns to global configuration mode.
Houston(config)#**route-map SETPATH permit 10**	Creates a route map named SETPATH. This route map will permit traffic based on subsequent criteria. A sequence num- ber of 10 is assigned.
Houston(config-route-map)#**set as-path prepend 300 300**	Read this command to say, "The local router will add (prepend) the autonomous system number 300 twice to the AS_PATH attribute before sending it out to its neighbor at 192.168.7.2."

The result of this configuration is that the AS_PATH attribute of updates for network 192.168.219.0 that autonomous system 600 receives via autonomous system 100 will be (100, 300, 300, 300), which is longer than the value of the AS_PATH attribute of updates for network 192.168.219.0 that autonomous system 600 receives via autono- mous system 400 (400, 200, 300).
Autonomous system 600 will choose autonomous system 400 (400, 200, 300) as the bet- ter path. This is because BGP is a path vector routing protocol that chooses the path with the least number of autonomous systems that it has to cross.

AS_PATH: Removing Private Autonomous Systems
Private autonomous system numbers (64,512 to 65,535) cannot be passed on to the Internet because they are not unique. Cisco has implemented a feature, **remove-private-as**, to strip private autonomous system numbers out of the AS_PATH list before the routes get propagated to the Internet. The **remove-private-as** command is available for eBGP neighbors only

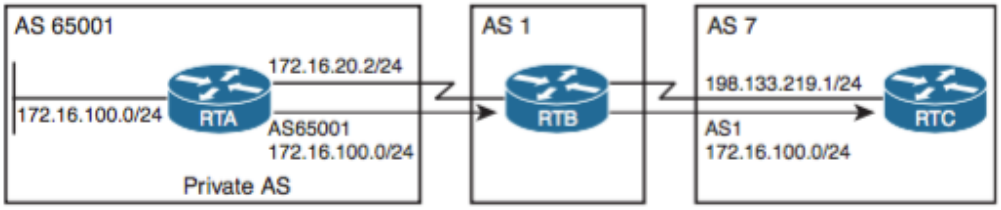


Figure 6-14 AS_PATH: Removing Private Autonomous Systems

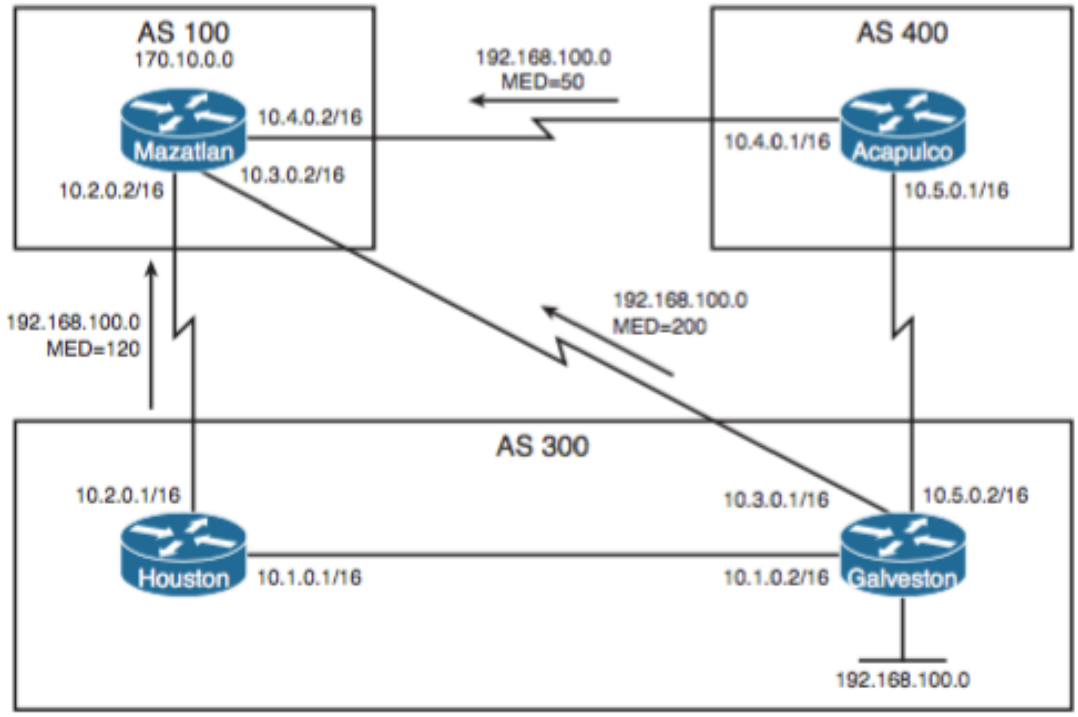
RTB(config)#**router bgp 1**	Starts the BGP routing process.
RTB(config-router)#**neighbor 172.16.20.2 remote-as 65001**	Identifies a peer router at 172.16.20.2.
RTB(config-router)#**neighbor 198.133.219.1 remote-as 7**	Identifies a peer router at 198.133.219.1.
RTB(config-router)#**neighbor 198.133.219.1 remove-private-as**	Removes private ASNs from the path in outbound routing updates.

MED Attribute

Also called ***the BGP metric***, indicates the preferred path is into an AS between two eBGP neighbors.

- It is an AS to AS specific value describing that hop only- is not later handed to the next peer.
- The **metric** command is used to configure the MED attribute.
- A lower MED value is preferred over a higher MED value.
- The default value is 0. Change the default value of the MED using the **default-metric** command
- By default, a router compares the MED attribute only for paths from neighbors in the same AS.
- If you want MED from neighbors in other AS'es to be compared, use the **bgp always-compare-med** command.

By default, BGP *only* compares the MED attributes of routes coming from neighbors in the same AS. Below, the routes for AS 300 to choose from are from Houston and Galveston. This is where MED is used by default (what it is intended for), so it won't be looked at in the case of AS 400 (Acapulco) unless it is specifically told to



Example: influence Mazatlan router to choose Houston as the entry point for AS 300 to reach 192.168.100.0.

Mazatlan(config)#**router bgp 100**	Starts the BGP routing process.
Mazatlan(config-router)#**neighbor 10.2.0.1 remote-as 300**	Identifies a peer router at 10.2.0.1.
Mazatlan(config-router)#**neighbor 10.3.0.1 remote-as 300**	Identifies a peer router at 10.3.0.1.
Mazatlan(config-router)#**neighbor 10.4.0.1 remote-as 400**	Identifies a peer router at 10.4.0.1.
Acapulco(config)#**router bgp 400**	Starts the BGP routing process.
Acapulco(config-router)#**neighbor 10.4.0.2 remote-as 100**	Identifies a peer router at 10.4.0.2.
Acapulco(config-router)#**neighbor 10.4.0.2 route-map SETMEDOUT out**	Refers to a route map named SETMEDOUT.
Acapulco(config-router)#**neighbor 10.5.0.2 remote-as 300**	Identifies a peer router at 10.5.0.2.
Acapulco(config-router)#**exit**	Returns to global configuration mode.
Acapulco(config)#**route-map SETMEDOUT permit 10**	Creates a route map named SETMEDOUT. A sequence number of 10 is assigned.
Acapulco(config-route-map)#**set metric 50**	Sets the metric value for BGP.
Houston(config)#**router bgp 300**	Starts the BGP routing process.
Houston(config-router)#**neighbor 10.2.0.2 remote-as 100**	Identifies a peer router at 10.2.0.1.
Houston(config-router)#**neighbor 10.2.0.2 route-map SETMEDOUT out**	Refers to a route map named SETMEDOUT.
Houston(config-router)#**neighbor 10.1.0.2 remote-as 300**	Identifies a peer router at 10.1.0.2.
Houston(config-router)#**exit**	Returns to global configuration mode.
Houston(config)#**route-map SETMEDOUT permit 10**	Creates a route map named SETMEDOUT. A sequence number of 10 is assigned.

Houston(config-route-map)#**set metric 120**	Sets the metric value for BGP.
Galveston(config)#**router bgp 300**	Starts the BGP routing process.
Galveston(config-router)#**neighbor 10.3.0.2 remote-as 100**	Identifies a peer router at 10.3.0.2.
Galveston(config-router)#**neighbor 10.3.0.2 route-map SETMEDOUT out**	Refers to a route map named SETMEDOUT.
Galveston(config-router)#**neighbor 10.1.0.1 remote-as 300**	Identifies a peer router at 10.1.0.1.
Galveston(config-router)#**neighbor 10.5.0.1 remote-as 400**	Identifies a peer router at 10.5.0.1
Galveston(config-router)#**exit**	Returns to global configuration mode.
Galveston(config)#**route-map SETMEDOUT permit 10**	Creates a route map named SETMEDOUT. A sequence number of 10 is assigned.
Galveston(config-route-map)#**set metric 200**	Sets the metric value for BGP.

Mazatlan can only compare the MED attribute coming from Houston (120) to the MED attribute coming from Galveston (200) even though the update coming from Acapulco has the lowest MED value. Mazatlan will choose Houston as the best path for reaching network 192.168.100.0.

To force Mazatlan to include updates for network 192.168.100.0 from Acapulco in the comparison, use the **bgp always-compare-med** router configuration command on Mazatlan:
Mazatlan(config)#**router bgp 100**
Mazatlan(config-router)#**neighbor 10.2.0.1 remote-as 300**
Mazatlan(config-router)#**neighbor 10.3.0.1 remote-as 300**
Mazatlan(config-router)#**neighbor 10.4.0.1 remote-as 400**
Mazatlan(config-router)#**bgp always-compare-med**
Assuming that all other attributes are the same, Mazatlan will choose Acapulco as the best next hop for reaching network 192.168.100.0.

The most recent IETF decision about BGP MED assigns a value of infinity to a missing MED, making that route the least preferred. The Cisco IOS default is the opposite - treating routes without the MED attribute as having a MED of 0, making the route that is lacking the MED variable the **most** preferred.
- To configure the router to conform to the IETF standard, use **bgp bestpath missing-as-worst**.

An enterprise has much less control over inbound routes: routes for packets coming back toward the enterprise. First, these inbound routes exist on routers that the enterprise does not own. Even if an ISP or set of ISPs can be convinced by engineers at the enterprise to make their routes toward an enterprise take a particular path, technical issues can prevent the design from being implemented. In particular, if the enterprise's public IP address range is summarized, the companies that use addresses in that range might have compet- ing goals. As a result, no policy can be applied to influence the best route.
However, several tools exist that allow some control over the last ASN hop between an ISP and its enterprise customer. This book examines one such tool, called Multi-Exit Discriminator (MED), which originally worked for a dual-homed design—that is, with a single ISP but with multiple links to that ISP. MED was later expanded to support dual- multihomed designs (2+ ASNs, 2+ links), relying on the concept that ISPs would work together. This section examines the dual-homed case, with a single ISP.

The name Multi-Exit Discriminator actually describes its function to a great degree. With a dual-homed design, at least two links exist between an enterprise and its ISP. The enterprise can announce to the ISP a value (MED) that tells the ISP which path into the enterprise is best. As a result, the ISP can discriminate between the multiple exit points from that ISP to the enterprise.
Because MED lets the enterprise ASN tell just the neighboring ASN which link into the enterprise to use, engineers typically use MED when advertising an enterprise's public IP address space. Those inbound routes into the enterprise from the ISP typically consist of either one, or a few, public IP address ranges.
For example, consider a new network design as shown in Figure 14-19. In this case, the enterprise uses the same 128.107.0.0/19 public address range used in Chapter 13 and in this chapter. The enterprise connects only to ASN 1 with a total of four physical links and three BGP neighbors.

MED uses smallest-is-best logic. As a result, the figure shows a design in which the enterprise engineer prefers the top BGP neighborship as the best path to use for inbound routes (MED 10), the middle link next (MED 20), and the bottom connection last (MED 30). Following the steps in the figure:

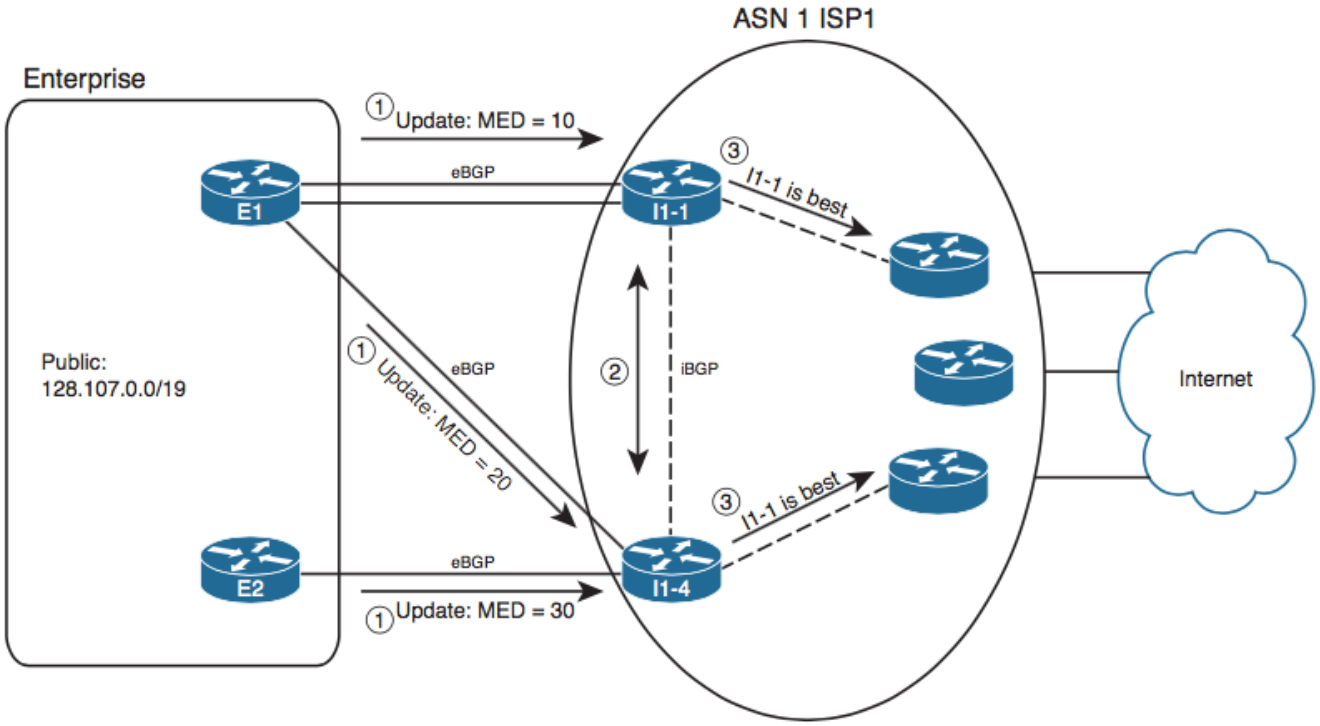


Figure 14-19 *Example of Using MED*

Step 1. E1 and E2 advertise 128.107.0.0/19, setting MED with an outbound route map, to various settings: MED 10 sent by E1 to I1-1, MED 20 sent by E1 to I1-4, and MED 30 sent by E2 to I1-4.
Step 2. I1-1 and I1-4 have an iBGP connection, so they learn each other's routes and agree as to which route wins based on MED.

Step 3. I1-1 and I1-4 also tell the other routers inside ISP1, causing all inbound traffic to funnel toward Router I1-1.
Note that Routers I1-1 and I1-4 in this example could have chosen a better route based on all the earlier best-path steps. However, a brief analysis of the steps tells us that unless someone makes an effort to override the effects of MED, these routers' best-path algo- rithms will use MED. Assuming that the enterprise and ISP agree to rely on MED, the earlier best-path steps should not matter. Here's why:

Weight: Needs to be set locally. Therefore, if relying on MED, the ISP simply chooses to not set the Weight for received Updates from the enterprise.
Local_Pref: Again, this takes overt effort to match and set the Local_Pref. If relying on MED, the ISP simply chooses to not set the Local_Pref.
Locally injected? All these public routes from the enterprise will be learned with eBGP and not locally injected.
AS_Path length: All such routes on the ISP routers should list one ASN—the enterprise's ASN—so all should tie on this point.
Origin: Whatever the Origin is (i, e, or ?), it should tie.
MED: None of the other steps determined the best route. Therefore, MED
now takes effect.

Allows an AS to tell a neighboring AS the best way to forward packets into the first AS.
Advertised by one AS into another, propagated inside the AS, but not sent to any other autonomous systems.

```
route-map set-med-to-I1-1 permit 10
  match ip address prefix-list only-public
  set metric 10
 !
```

```
route-map set-med-to-I1-4 permit 10
 match ip address prefix-list only-public
 set metric 20
!
ip prefix-list only-public permit 128.107.0.0/19
!
router bgp 11
 neighbor 1.1.1.1 route-map set-med-I1-1 out
 neighbor 192.168.1.2 route-map set-med-I1-4 out
```

The show ip bgp longer-prefixes command's briefer output, and the show ip bgp 185.0.0.0/8 commands more verbose output, both identify the Local_Pref value. However, the longer command output does not list the Weight value

---

When the BGP best-path algorithm has gotten through this complexity and chosen a best route for a prefix, the router then tries to add that route to the IP routing table. However, rather than add the BGP route to the IP routing table directly, BGP actually gives that best BGP route to another process for consideration: the Cisco IOS Routing Table Manager (RTM).

The Cisco IOS RTM chooses the best route among many competing sources. For exam- ple, routes can be learned by an IGP, BGP, or even as connected or static routes. Cisco IOS collects the best such route for each prefix and feeds those into the RTM function. The RTM then chooses the best route

For the most part, an enterprise router should not see cases in which a prefix learned with BGP has also been learned as a connected or IGP-learned route. (Conversely, these issues occur more often when implementing MPLS VPNs with BGP/IGP redistribution.) However, it can happen, and when it does, the show ip bgp rib-failures command can be helpful. This command lists routes for which BGP has chosen the route as best, but the RTM function has not placed the route into the Routing Information Base (RIB), which is simply another name for the IP routing table.

Example of a BGP RIB Failure

To show an example of a RIB failure, imagine that an enterprise engineer needs to do some testing, so the engineer just picks an IP address range to use. The engineer tries
to avoid problems by not using network 10.0.0.0, which is used throughout the enter- prise. Rather than choosing another private network, the engineer chooses public range 185.0.0.0/8. After changing the lab configuration repeatedly, a route for 185.0.0.0/8 leaks into the OSPF topology database.

Keep in mind that at the end of the previous example, E1 had chosen its eBGP route for 185.0.0.0/8 as its best route, and E2 had chosen its iBGP route as its best route for 185.0.0.0/8. Example 14-21 shows the results, based on RTM's comparisons of the AD values.

Example 14-21 *Example with the RTM and RIB Failures*

```
! First, E1's IP Routing table for 185.0.0.0/8
E1# show ip route 185.0.0.0 255.0.0.0 longer-prefixes
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 1.1.1.1 to network 0.0.0.0

B    185.0.0.0/8 [20/0] via 1.1.1.1, 00:25:11
```

```
! Next, E2's IP Routing table
E2# show ip route 185.0.0.0 255.0.0.0 longer-prefixes
! Legend omitted for brevity

Gateway of last resort is 192.168.1.6 to network 0.0.0.0

O    185.0.0.0/8 [110/2] via 10.1.1.77, 00:15:44, FastEthernet0/0

E2# show ip bgp rib-failure
Network            Next Hop                  RIB-failure   RIB-NH Matches
185.0.0.0/8        10.100.1.1          Higher admin distance            n/a
```

The first command shows that E1, with an eBGP route, actually adds its route to the IP routing table. The route lists a code of B, meaning BGP. The output lists the eBGP default AD of 20, which is a better default AD than OSPF's 110. RTM added this BGP route to the IP routing table on E1 because of eBGP's better AD.

E2 currently lists its iBGP route through E1 as its current best BGP route for 185.0.0.0/8 because of the higher Local_Pref configured in Example 14-20. However, after giving this route to the RTM, RTM instead chose the lower-AD OSPF route (AD 110) rather than the higher-AD iBGP route (AD 200).

Finally, the show ip bgp rib-failure command lists one line for each best BGP route that the RTM does not place into the IP routing table. In this case, this command on Router E2 lists the route for 185.0.0.0/8, with the reason listed.

Using Access Lists to Manipulate the Weight Attribute

Refer to Figure 6-11 for the config that follows to configure the weight attribute using AS_PATH access lists.

Houston(config)#**router bgp 300**	Starts the BGP routing process.
Houston(config-router)#**neighbor 192.168.7.1 remote-as 100**	Identifies a peer router at 192.168.7.1.
Houston(config-router)#**neighbor 192.168.7.1 filter-list 5 weight 2000**	Assigns a **weight** attribute of 2000 to updates from the neighbor at 192.168.7.1 that are permitted by ACL 5. (defined below in global configuration mode)
Houston(config-router)#**neighbor 192.168.219.1 remote-as 200**	Identifies a peer router at 192.168.219.1.
Houston(config-router)#**neighbor 192.168.219.1 filter-list 6 weight 1000**	Assigns a **weight** attribute of 1000 to updates from the neighbor at 192.168.219.1 that are permitted by ACL 6 (defined below in global configuration mode)
Houston(config-router)#**exit**	Returns to global configuration mode.
Houston(config)#**ip as-path access- list 5 permit ^100$**	Permits updates whose AS_PATH attribute starts and ends with 100 (represented by the ^ and $).
	The ^ and $ symbols are used for regular expressions. (See section on regular expressions in IOS)
Houston(config)#**ip as-path access- list 6 permit ^200$**	Permits updates whose AS_PATH attribute starts and ends with 200 (represented by the ^ and $).

The result of this configuration will have Houston forward traffic for the 172.16.10.0 network through autonomous system 100, because it has a higher weight attribute set as compared to the weight attribute set for the same update from autonomous system 200. We call the ACL with the filter-list, then declare the ACL contents with an AS_PATH ACL in the global config.

Using Route Maps to Manipulate the Weight Attribute (do the same as the last)

Houston(config)#router bgp 300	Starts the BGP routing process.
Houston(config-router)#neighbor 192.168.7.1 remote-as 100	Identifies a peer router at 192.168.7.1.
Houston(config-router)#neighbor 192.168.7.1 route-map SETWEIGHT in	Identifies that the route map named SETWEIGHT will be used to assign weights to route updates. [All routes originating from 192.168.7.1 must meet conditions in this route map]
Houston(config-router)#neighbor 192.168.219.1 remote-as 200	Identifies a peer router at 192.168.219.1.
Houston(config-router)#neighbor 192.168.219.1 route-map SETWEIGHT in	Identifies that the route map named SETWEIGHT will be used to assign weights to route updates.
Houston(config-router)#exit	Returns to global configuration mode.
Houston(config)#ip as-path access-list 5 permit ^100$	Permits updates whose AS_PATH attribute starts and ends with 100 (represented by the ^ and $).
Houston(config)#route-map SETWEIGHT permit 10	Creates a route map called SETWEIGHT to permit traffic based on criteria. A sequence number of 10 is assigned.
Houston(config-route-map)#match as-path 5	Specifies the condition under which policy routing is allowed: matching the BGP ACL 5.
Houston(config-route-map)#set weight 2000	Assigns a weight of 2000 to any route update that meets the condition of ACL 5 (AS_PATH that meets "100")
Houston(config-route-map)#exit	Returns to global configuration mode.
Houston(config)#route-map SETWEIGHT permit 20	Creates the second statement for SETWEIGHT to permit traffic. A sequence number of 20 is assigned.
Houston(config-route-map)#set weight 1000	Assigns a weight of 1000 to route updates from any AS other than 100. AS 100 will be assigned a weight of 2000 due to the first statement the route map.

Here, we tag the neighbor statements with the route-map assignment, then later, in global config mode, use an as-path ACL to match the AS in the AS_PATH with a regexp match, refer to that ACL to set the route-map to first match it, then catch the rest in the second statement (for those that didn't meet the first criteria)

Using Prefix Lists AND Route Maps to Manipulate the Weight Attribute (yet another way to do the same)

Houston(config)#**ip prefix-list AS400_ROUTES permit 172.16.10.0/24**	Creates a prefix list that matches the 172.16.10.0/24 network belonging to AS 400.
Houston(config)#**route-map SETWEIGHT permit 10**	Creates a route map called SETWEIGHT. A sequence number of 10 is assigned.
Houston(config-route-map)#**match ip address prefix-list AS400_ ROUTES**	Specifies the condition under which policy routing is allowed, matching the AS400_ROUTES prefix list.
Houston(config-route-map)#**set weight 200**	Assigns a weight of 200 to any route update that meets the condition of prefix list AS400_ROUTES.
Houston(config-route-map)#**route- map SETWEIGHT permit 20**	Creates the second statement for the route map A sequence number of 20 is assigned.
Houston(config-route-map)#**set weight 100**	Assign weight of 100 to all route updates/networks learned from outside AS400_ROUTES (didn't meet crit 1)
Houston(config-route-map)#**exit**	Returns to global configuration mode.
Houston(config)#**router bgp 300**	Starts the BGP routing process.
Houston(config-router)#**neighbor 192.168.7.1 route-map SETWEIGHT in**	Uses the route map SETWEIGHT to filter all routes learned from neighbor 192.168.7.1.

Here, a prefix-list is set to match specific traffic (network/subnet), then the route-map is set up with it's first rule to match that prefix-list item, and the second one to catch all of the rest.

Local Preference Attribute

- Local preference is a BGP attribute that tells routers in the AS about the path that is preferred for exiting the AS.
- The **local-preference** value can be between 0 and 429,496,729. Higher is preferred, default is 100.
- It is local to the AS; it is exchanged between iBGP peers - not advertised to eBGP peers.
- Use the **local-preference** attribute to force BGP routers to prefer one exit point over another.
- R1(config-router)#**bgp default local-preference 150**

Houston(config)#router bgp 256	Starts the BGP routing process.
Houston(config-router)#neighbor 172.16.1.1 remote-as 100	Identifies a peer router at 172.16.1.1.
Houston(config-router)#neighbor 10.1.1.2 remote-as 256	Identifies a peer router at 10.1.1.2.
Houston(config-router)#bgp default local-preference 150	Sets the local preference attribute on this router.
Galveston(config)#router bgp 256	Starts the BGP routing process.
Galveston(config-router)#neighbor 172.17.1.1 remote-as 300	Identifies a peer router at 172.17.1.1.
Galveston(config-router)#neighbor 10.1.1.1 remote-as 256	Identifies a peer router at 10.1.1.1.
Galveston(config-router)#bgp default local-preference 200	Sets the local preference attribute on this router.

Based on these two configurations, traffic destined for a remote network that can be reached through autonomous system 256 will be routed through Galveston.

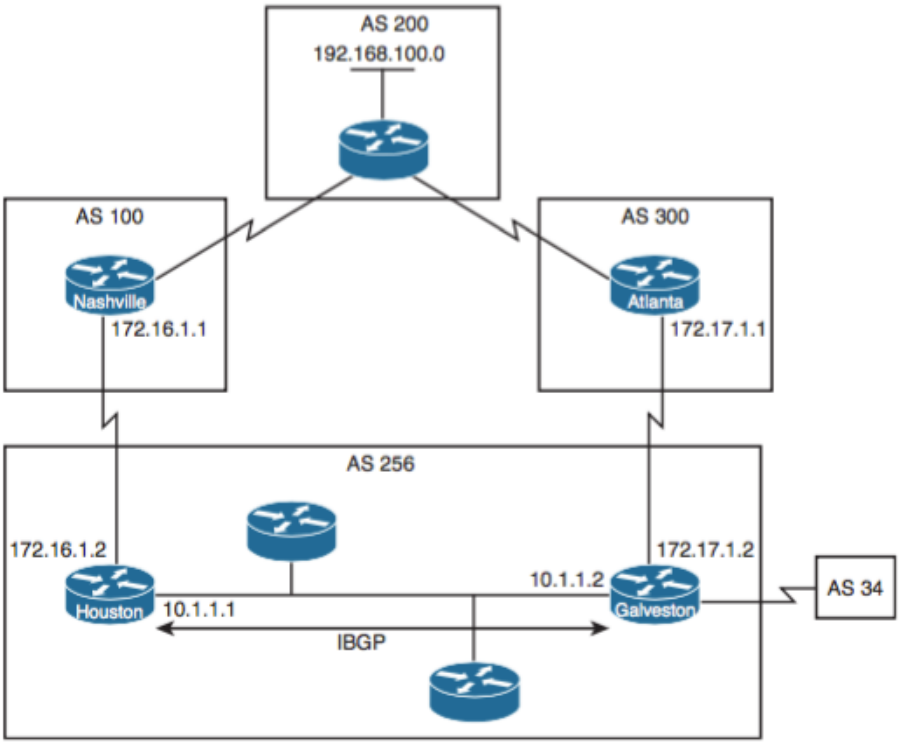


Figure 6-12

Using AS_PATH Access Lists with Route Maps to Manipulate the Local Preference Attribute

Route maps provide more flexibility than the **bgp default local-preference** router configuration command.

Galveston(config)#**router bgp 256**	Starts the BGP routing process.
Galveston(config-router)#**neighbor 172.17.1.1 remote-as 300**	Identifies a peer router at 172.17.1.1.
Galveston(config-router)#**neighbor 172.17.1.1 route-map SETLOCAL in**	Refers to a route map called SETLOCAL.
Galveston(config-router)#**neighbor 10.1.1.1 remote-as 256**	Identifies a peer router at 10.1.1.1.
Galveston(config-router)#**exit**	Returns to global configuration mode.
Galveston(config)#**ip as-path access-list 7 permit ^300$**	Permits updates whose AS_PATH attribute starts and ends with 300 (represented by the ^ and $).
Galveston(config)#**route-map SETLOCAL permit 10**	Creates a route map called SETWEIGHT. A sequence number of 10 is assigned.
Galveston(config-route-map)#**match as-path 7**	Specifies the condition under which policy routing is allowed- matching the BGP ACL 7.
Galveston(config-route-map)#**set local-preference 200**	Assigns a local preference of 200 to any update coming from autonomous system 300 (ACL 7)
Galveston(config-route- map)#**route-map SETLOCAL permit 20**	Second statement of route map SETLOCAL. This instance will accept all other routes.

In the previous example, using the **bgp default local-preference** command on Galveston, the local preference attribute of *all* routing updates received by Galveston would be set to 200. This would include updates from autonomous system 34. In this example, using the **route-map** command, only updates received from autonomous system 300, as specified in the **ip as_path access-list** command, will have a local preference set to 200.

Route Aggregation - Setting the Atomic Aggregate attribute

- A BGP router can transmit overlapping routes (nonidentical routes that point to the same destination)
- When making a best path decision, a router always chooses the more specific path.

R1(config-router)#**aggregate- address 172.16.0.0 255.255.0.0**	Creates an aggregate entry in the BGP routing table to cover specific BGP routes are in that range. More specific routes will still be sent unless **summary-only** is used.
R1(config-router)#**aggregate- address 172.16.0.0 255.255.0.0 summary-only**	Creates aggregate route AND suppresses advertisement of more-specific routes. Specific AS_PATH info on those subnets are lost.
R1(config-router)#**aggregate- address 172.16.0.0 255.255.0.0 as-set**	Creates an aggregate entry but the path advertised will be an AS_SET or list of AS_PATHs where individual subnets originated.

Below is a use of aggregate route as a precaution: to send an aggregate address, we only need one of the more specific routes configured. But by configuring all of them, and *not* using **summary-only**, they will all be sent, and the aggregate will be sent in case one of the networks goes down.

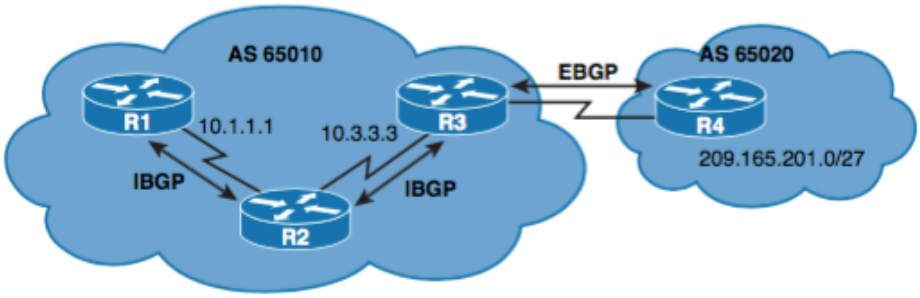
Lubbock(config)#router bgp 1	Starts the BGP routing process.
Lubbock(config-router)#neighbor 10.1.1.2 remote-as 2	Identifies a peer router at 10.1.1.2.
Austin(config)#router bgp 2	Starts the BGP routing process.
Austin(config-router)#neighbor 10.1.1.1 remote-as 1	Identifies a peer router at 10.1.1.1.
Austin(config-router)#network 172.16.0.0 mask 255.255.255.0	Advertises a specific subnet.
Austin(config-router)#network 172.16.1.0 mask 255.255.255.0	Advertises a specific subnet.
Austin(config-router)#network 172.16.2.0 mask 255.255.255.0	Advertises a specific subnet.
Austin(config-router)#aggregate- address 172.16.0.0 255.255.252.0	Advertises the aggregate address.

Both Lubbock and Austin will have all the specific routes *and* the aggregate address in its BGP table

```
Lubbock#show ip bgp 172.16.0.0 255.255.252.0
BGP routing table entry for 172.16.0.0/22, version 18
Paths: (1 available, best #1)
<text omitted>
 Origin IGP, localpref 100, valid, external, atomic-aggregate,
  best
```

Route Reflectors

By default, a router that receives an eBGP route advertises it to its eBGP and iBGP peers.
If it receives it through iBGP, it won't advertise it to its iBGP peers, as loop-prevention (split horizon).
One way for all iBGP routers to receive a route after it is originated into the AS is to have a full mesh of iBGP peers, which can get complex with a large number of peers.
Designating route reflectors eliminates resorting to that.



The objective is to allow R2 to advertise to R1 the 209.165.201.0/27 network learned from R3. Without these commands, R1 would never learn the 209.165.201.0/27 network without a full-mesh iBGP topology.

R2(config)#**router bgp 65010**	Enters BGP routing configuration mode
R2(config-router)#**neighbor 10.1.1.1 route-reflector-client**	Designates a BGP route reflector and the specified neighbor as a client
R2(config-router)#**neighbor 10.3.3.3 route-reflector-client**	Designates a BGP route reflector and the specified neighbor as a client

Syntax for BGP Filtering Methods

Traditional access lists not great for granular BGP filters
The old-school IOS ACLs we know might be seem like a good way to match traffic by prefix/subnet, but it can get to be a mess. Cisco IOS 12.0 came up with a better way with **prefix-list**, but it helps to know why this was done.

Here is an easy scenario: declare BGP neighbors, make a simple access list to match one subnet, and apply it with **distribute-list** to the updates going **out**, so we can say "don't send routing updates from that **access-list 1** to this neighbor". This works fine.

```
R1(config)# router bgp 3
R1(config-router)# neighbor 172.16.1.2 remote-as 3
R1(config-router)# neighbor 172.16.20.1 remote-as 1
R1(config-router)# neighbor 172.16.20.1 distribute-list 1 out
R1(config-router)# exit
R1(config)# access-list 1 deny 192.168.10.0 0.0.0.255
R1(config)# access-list 1 permit any
```

Problems come up when advertising the aggregate address of 172.16.0.0/16 but not the individual subnet- you need an extended ACL. The way those work with BGP route filters, the ACL will first match the network address and *then* match the subnet mask of the prefix. To do this, both network and netmask are paired with their own wildcard bitmask and it looks like this confusing, ugly mess:

access-list 101 permit ip 172.16.0.0 0.0.255.255 255.255.0.0 0.0.0.0

That is why IOS 12.0 gave us the **prefix-list** option.

Prefix-List Syntax

This example limits updates sent to the peer to only be routing info for this specific matching subnet:

```
R1(config)# ip prefix-list UPDATE172 permit 172.16.0.0/16
R1(config)# router bgp 100
R1(config-router)# neighbor 192.168.1.1 remote-as 200
R1(config-router)# neighbor 192.168.1.1 prefix-list UPDATE172 out
```

Applies UPDATE172 to routing updates sent to 192.168.1.1. Permits update of 172.16.0.0/16, but has implicit deny at end of the list for others like 172.16.0.0/17 or 172.16.20/24

ip prefix-list list-name **[seq** seq-value**] deny | permit** network/cidr **[ge** ge-value**] [le** le-value**]**

Parameter	Description
list-name	The name of the prefix list.
seq	(Optional) Applies a sequence number to the entry being created or deleted.
seq-value	(Optional) Specifies the sequence number.
deny	Denies access to matching conditions.
permit	Permits access for matching conditions.
network/cidr	(Mandatory) The network number and length (in bits) of the netmask.
ge	(Optional) Applies *ge-value* to the range specified.
ge-value	(Optional) Specifies the beginning or "from" value of a range
le	(Optional) Applies *le-value* to the range specified.
le-value	(Optional) Specifies the end or "to" value of a range

- There is an **implicit deny** statement at the end of each prefix list.
- The range of sequence numbers that can be entered is from 1 to 4,294,967,294.
- A router tests for prefix list matches from the lowest sequence number to the highest.
- By numbering your **prefix-list** statements, you can add new entries at any point in the list.
- If no seq # is given, default is applied: 5 is applied to the first, next unnumbered entries are incremented by 5.

ip prefix-list NY_ROUTES permit 192.0.0.0/8 le 24
Permit routes with a netmask of up to 24 bits in 192.0.0.0/8 - No seq number given- gets the default 5

ip prefix-list NY_ROUTES deny 192.0.0.0/8 ge 25
Deny routes with netmask of 25 bits or greater in 192.0.0.0/8 - No seq num given- 10 is applied (+5)

ip prefix-list RENO permit 10.0.0.0/8 ge 16 le 24
Permit routes 10.0.0.0/8 with a netmask between 16 to 24 bits - No seq number given- gets the default 5

ip prefix-list HOUSTON seq 3 deny 0.0.0.0/0
Assigns a sequence number of 3 to this statement.

no ip prefix-list TORONTO seq 10
Removes sequence number 10 from the TORONTO list.

AS_PATH Access Lists

The AS_PATH attribute can be searched with regular expressions to match routing updates, and slap with an access-list label to filter or modify. Most of these should look familiar. They can also be tested with **show ip bgp**

^	Matches the beginning of the input string.
$	Matches the end of the input string.
^$	Matches an empty string in the AS_PATH field (the local device's AS is blank, so this matches that)
_	Matches a space, comma, left brace, right brace, beginning or end of an input string
.	Matches any single character.
*	Matches 0 or more single- or multiple-character patterns.

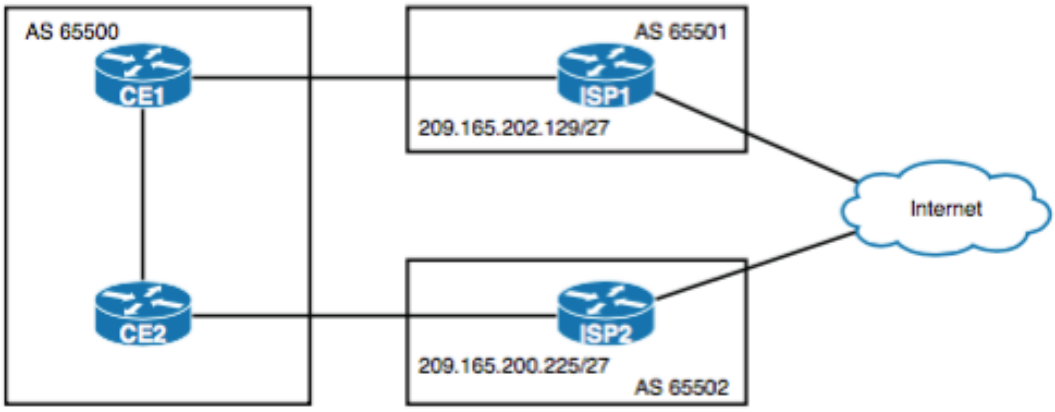
To find all subnets reachable via autonomous system 65002 (AS_PATH begins with 65002):

R1#**show ip bgp regexp ^65002_**

Network	Next Hop	Metric	LocPrf	Weight	Path	
*>i172.16.0.0	192.168.28.1			100	0	65002 65003 i
* i172.24.0.0	192.168.28.1			100	0	65002 65003 65004 65005 i

show ip bgp regexp _65004$	... originating from autonomous system 65004 (AS_PATH ends with 65004):
show ip bgp regexp ^65002_	... reachable via autonomous system 65002 (AS_PATH begins with 65002):
show ip bgp regexp _65005_	... transiting through autonomous system 65005 (AS_PATH contains 65005):
show ip bgp regexp 2150	Will match 2150, 12150 or 21507. This is where "_" helps
show ip bgp ^$	Originates from THIS autonomous system (AS_PATH is blank)

This simple example demonstrates using prefix lists and AS_PATH access-lists together. Notice how the AS_PATH list is referred to with the **filter-list** keyword. The first job is to allow CE1 and CE2 to only learn ISP routes with a mask greater than /15 (ge 16) and less than /25 (le 24). The second job is to ensure that AS 65000 does not become a transit autonomous system for ISP1 to reach ISP2 (and vice versa).



C1 Configuration	
ip prefix-list ISP1 permit 0.0.0.0 ge 16 le 24	Prefix list which only permits routes with a mask between 16 and 24
ip as-path access- list 1 permit ^$	Creates AS_PATH ACL matching routes that only originate from within autonomous system 65500
router bgp 65000	
neighbor 209.165.202.129 prefix-list ISP1 in	Assigns ISP1 prefix list to neighbor 209.165.202.129 (ISP1) for all routes learned from that neighbor
neighbor 209.165.202.129 filter-list 1 out	Assigns AS_PATH ACL to neighbor 209.165.202.129 (ISP1) for all routes sent to that neighbor

C2 Configuration	
ip prefix-list ISP2 permit 0.0.0.0 ge 16 le 24	Creates a prefix list that only permits routes with a mask between 16 and 24
ip as-path access- list 1 permit ^$	Creates an AS_PATH access list matching routes that only originate from within AS 65500
router bgp 65000	
neighbor 209.165.200.225 prefix-list ISP2 in	Assigns ISP2 prefix list to neighbor 209.165.200.225 (ISP2) for all routes learnt from that neighbor
neighbor 209.165.200.225 filter-list 1 out	Assigns AS_PATH ACL to neighbor 209.165.200.225 (ISP2) for all routes sent to that neighbor

Route Maps
This example uses a prefix-list to catch updates matching 172.16.10.0/24 and label with "AS400_ROUTES". Then a route-map is created so we can basically make a conditional statement that says, if it sees those, make a change to the weight attribute. If it doesn't match and falls through to match the second rule, set a different weight. This same mechanism can be used to make other changes, or specify what updates to use or drop.

Finally, at the end make the BGP process, add the neighbors and directs to check the route-map for the rules. Adding the keyword "in" refers to routing updates from 192.168.7.1, and the keyword "out" is used for the routing updates we send out 192.168.7.1

Houston(config)#**ip prefix-list AS400_ROUTES permit 172.16.10.0/24**
Creates a prefix list that matches the 172.16.10.0/24 network belonging to AS 400.
Houston(config)#**route-map SETWEIGHT permit 10**
Creates a route map called SETWEIGHT. A sequence number of 10 is assigned.
Houston(config-route-map)#**match ip address prefix-list AS400_ ROUTES**
Specifies the condition under which policy routing is allowed, matching the AS400_ROUTES prefix list.
Houston(config-route-map)#**set weight 200**
Assigns a weight of 200 to any route update that meets the condition of prefix list AS400_ROUTES.
Houston(config-route-map)#**route- map SETWEIGHT permit 20**
Creates a second statement for the route map. A sequence number of 20 is assigned.
Houston(config-route-map)#**set weight 100**
Assign weight of 100 to all route updates/networks learned from outside (didn't match) AS400_ROUTES
Houston(config-route-map)#**exit**
Returns to global configuration mode.
Houston(config)#**router bgp 300**
Starts the BGP routing process.
Houston(config-router)#**neighbor 192.168.7.1 route-map SETWEIGHT in**
Uses the route map SETWEIGHT to filter all routes learned from neighbor 192.168.7.1.

Route-map can employ regular and extended ACLs, prefix-lists, and AS_PATH access-lists.

BGP Subcommand	Commands to create	What Can Be Matched
neighbor distribute-list (standard ACL)	**access-list, ip access-list**	Prefix, with WC mask
neighbor distribute-list (extended ACL)	**access-list, ip access-list**	Prefix and prefix length, with WC mask for each
neighbor prefix-list	**ip prefix-list**	Exact or "first N" bits of prefix, plus range of prefix lengths
neighbor filter-list	**ip as-path access-list**	AS_Path contents; all NLRI whose AS_Paths are matched considered to be a match
neighbor route-map	**route-map**	Prefix, prefix length, AS_ Path, and/or any other PA matchable within a BGP route map

E1(config)# **ip prefix-list only-public permit 128.107.0.0/19**
E1(config)# **router bgp 11**
E1(config-router)# **neighbor 1.1.1.1 prefix-list only-public out**
E1(config-router)# **end**
E1# **show ip bgp neighbor 1.1.1.1 advertised-routes**
--- *Will show change didn't take effect- BGP process needs a reset for change to take effect*
E1# **clear ip bgp 1.1.1.1**

Applicative syntax: prefix-list <LABEL> [in | out]
--- in means routes FROM updates matching IPs and out means updates sent TO those matching IPs
--- is separate from neighbor declaration line (neighbor 192.168.1.1 remote-as 200)
Example1: neighbor 192.168.7 prefix-list AS400_ROUTES in
Example2 (in Route-Map): match ip address prefix-list AS400_ROUTES

AS_PATH Access-List Syntax
Declarative syntax: as-path access-list <#> permit <matching-criteria>
----- assign a numerical identifier to the access list, it's items match criteria in AS_PATH
Example: as-path access-list 5 permit ^100$
--- makes an access list "5" that matches occurrences of simply 100 (with regexp for begin and end of line)

Applicative syntax: filter-list (#) (action)
Example1: neighbor 192.168.7.1 filter-list 5 weight 2000
Example2 (in Route-Map): match as-path 5

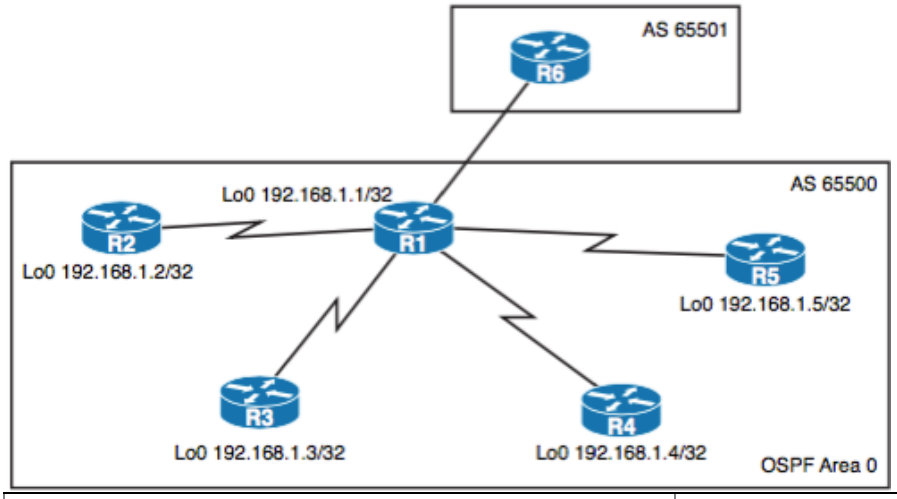
Using a regular ACL with distribute-list to apply
Set:
access-list 1 deny 192.168.21.2 0.0.0.255
access-list permit any
Apply:
neighbor 172.12.20.1 distribute-list 1 out
In this context, when sending updates out to 172.12.20.1, updates from 192.168.21.1 are left out (deny)

Simple changes without route maps, prefix lists etc:
E1(config)# router bgp 11
E1(config-router)# neighbor 1.1.1.1 weight 60
E1(config-router)# end
E1# clear ip bgp 1.1.1.1 soft

The neighbor weight command does not use an in or out direction, because Weight can only be set on input.
The configuration results in all routes learned from 1.1.1.1

BGP and the maximum-paths Command
Like the IGP protocols, BGP supports the maximum-paths number-of-paths subcom- mand, but BGP uses significantly different logic than the IGPs. Unlike the IGP routing protocols, BGP truly needs to pick one route, and only one route, as the best path for
a given prefix/length. In effect, the BGP best-path algorithm already breaks the ties for “best” route for each prefix. Therefore, from BGP’s perspective, one route for each prefix is always best.
BGP does allow multiple BGP routes for a prefix to be considered to tie, at least for the purpose of adding multiple routes to the IP routing table. The conditions are as follows:
If the BGP best-path algorithm does not choose a best path by Step 8 (per the num- bering in this book), the routes which still tie for being best path will be allowed into the IP routing table, up to and including the number defined by the BGP maximum- paths number-of-paths router subcommand.
The section “Overview of the BGP Best-Path Algorithm,” earlier in this chapter, lists the best-path steps, including the tiebreaker steps that allow routes to be considered by the maximum-paths command.

BGP Peer Groups
To ease the burden of configuring a large number of neighbors with identical or similar parameters (for example, route maps, filter lists, or prefix lists), the concept of peer groups was introduced. The administrator configures the peer group with all the BGP parameters that are to be applied to many BGP peers. Actual BGP neighbors are bound to the peer group, and the network administrator applies the peer group configuration on each of the BGP sessions. The result below, all four iBGP neighbors have the same basic BGP configuration assigned to them.



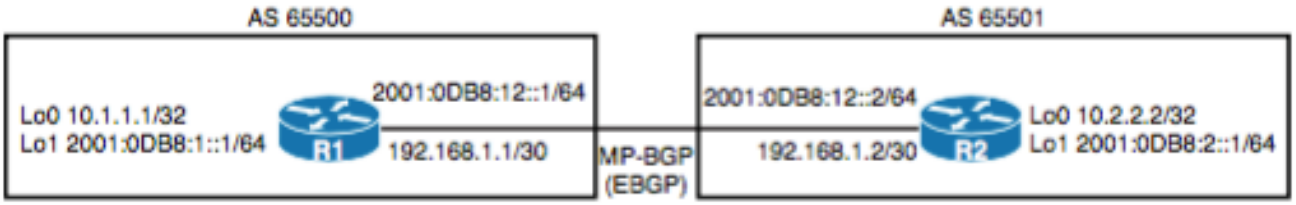
router bgp 65500	Starts the BGP routing process
neighbor INTERNAL peer-group	Creates a BGP peer group called INTERNAL
neighbor INTERNAL remote-as 65500	Assigns a first parameter to the peer group
neighbor INTERNAL next-hop-self	Assigns a second parameter to the group
neighbor INTERNAL update-source loopback0	Assigns a third parameter to the peer group
neighbor INTERNAL route-reflector-client	Assigns a fourth parameter to the peer group
neighbor 192.168.1.2 peer-group INTERNAL	Assigns the peer group to neighbor R2
neighbor 192.168.1.3 peer-group INTERNAL	... to neighbor R3
neighbor 192.168.1.4 peer-group INTERNAL	... to neighbor R4
neighbor 192.168.1.5 peer-group INTERNAL	... to neighbor R5

A peer group can be, among others, configured to do the following:
- Use IP of a specific interface as source address when opening the TCP session or use next-hop-self feature
- Use, or not use, things like the eBGP multihop function, or MD5 authentication on the BGP sessions.
- Filter out any incoming or outgoing routes using a prefix list, a filter list, and a route map.
- Assign a particular weight value to the routes that are received.

[[
Community
BGP communities are used to group networking devices that share common properties, regardless of network, autonomous system, or any physical boundaries. In large networks applying a common routing policy through prefix lists or access lists requires individual peer statements on each networking device. Using the BGP community attribute BGP neighbors, with common routing policies, can implement inbound or outbound route filters based on the community tag rather than consult large lists of individual permit or deny statements.
None of this mentions communities and this is from
IP Routing: BGP Configuration Guide, Cisco IOS XE Release 3S Dec 04, 2014
Chapter: Connecting to a Service Provider Using External BGP

]]

MP-BGP
Original BGP was designed for only IPv4. MP-BGP (Multiprotocol BGP - RFC 2858) can run over as IPv6, multicast IPv4, and MPLS - can exchange routes for IPv4, IPv6, or both. The extensions enable NEXT_HOP to carry IPv6 addresses and NLRI (Network Layer Reachability Information) to an IPv6 prefix, thanks to TCP.
.
Configure MP-BGP Using Address Families to Exchange IPv4 and IPv6 Routes
MP-BGP is used to exchange IPv4 and IPv6 routes. The IPv4 and IPv6 routes use separate TCP connections. Below is demonstrated how to configure MP-BGP using address families to exchange both IPv4 and IPv6 routes.

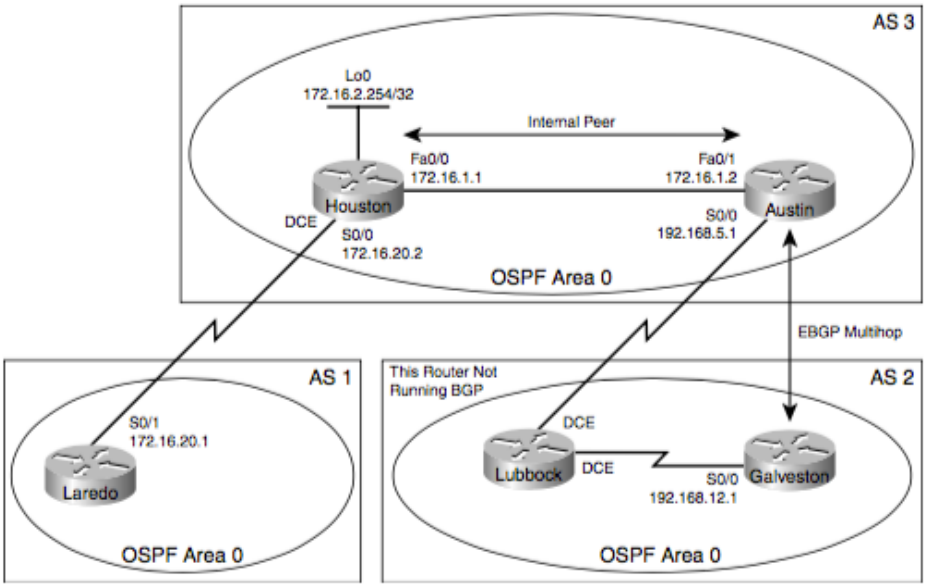


R1(config)#**ipv6 unicast-routing**	Enables the forwarding of IPV6 unicast datagrams globally on the router.
R1(config)#**router bgp 65500**	Starts the BGP routing process.
R1(config-router)#**neighbor 2001:0DB8:12::2 remote-as 65501**	Configures R2 as an IPv6 BGP neighbor.
R1(config-router)#**neighbor 192.168.1.2 remote-as 65501**	Configures R2 as an IPv4 BGP neighbor.
R1(config-router)#**address-family ipv4 unicast**	Enters IPv4 address family configuration mode for unicast prefixes (default IPv4)
R1(config-router-af)#**neighbor 192.168.1.2 activate**	Enables the exchange of IPv4 BGP info with R2. IPv4 neighbors auto-activate, so keyword is optional.
R1(config-router-af)#**network 10.1.1.1 mask 255.255.255.255**	Advertises an IPv4 network into BGP.
R1(config-router-af)#**exit**	Exits the IPv4 address family config mode.
R1(config-router)#**address-family ipv6 unicast**	Enters IPv6 address family configuration mode for unicast address prefixes.
R1(config-router-af)#**neighbor 2001:0DB8:12::2 activate**	Enables the exchange of IPv6 BGP infor- mation with R2.
R1(config-router-af)#**network 2001:0DB8:1::1/64**	Advertises an IPv6 network into BGP.
R2(config)#**ipv6 unicast-routing**	Enables the forwarding of IPv6 unicast datagrams globally on the router.
R2(config)#**router bgp 65501**	Starts the BGP routing process.
R2(config-router)#**neighbor 2001:0DB8:12::1 remote-as 65500**	Configures R1 as an IPv6 BGP neighbor.
R2(config-router)#**neighbor 192.168.1.1 remote-as 65500**	Configures R1 as an IPv4 BGP neighbor.
R2(config-router)#**address-family ipv4 unicast**	Enters IPv4 address family configuration mode for unicast address prefixes.
R2(config-router-af)#**neighbor 192.168.1.1 activate**	Enables the exchange of IPv4 BGP info with R1. The IPv4 neighbors will be automatically activated, so this is optional.
R2(config-router-af)#**network 10.2.2.2 mask 255.255.255.255**	Advertises an IPv4 network into BGP.
R2(config-router-af)#**exit**	Exits the IPv4 address family configura- tion mode.
R2(config-router)#**address-family ipv6 unicast**	Enters IPv6 address family configuration mode for unicast address prefixes.
R2(config-router-af)#**neighbor 2001:0DB8:12::1 activate**	Enables the exchange of IPv6 BGP information with R1.
R2(config-router-af)#**network 2001:0DB8:2::1/64**	Advertises an IPv6 network into BGP.

Recall that the router ID is set on IPv4 addresses, so a router running BGP over IPv6will at least have it manually set using the **bgp router-id** *IPv4_address* BGP configuration command.

Troubleshooting MP-BGP
show bgp ipv6 unicast [optional: summary | neighbors] These work just like the non-IPv6 versions
show ipv6 route bgp Displays the content of the IPv6 routing table

BGP Configuration Example



Houston Router

Router(config)#hostname Houston	Sets the router name to Houston.
Houston(config)#interface loopback 0	Moves to loopback interface mode.
Houston(config-if)#ip address 172.16.2.254 255.255.255.255	Assigns an IP address and netmask.
Houston(config-if)#interface fastethernet 0/0	Moves to interface configuration mode.
Houston(config-if)#ip address 172.16.1.1 255.255.255.0	Assigns an IP address and netmask.
Houston(config-if)#no shutdown	Enables the interface.
Houston(config-if)#interface serial 0/0/0	Moves to interface configuration mode.
Houston(config-if)#ip address 172.16.20.2 255.255.255.0	Assigns an IP address and netmask.
Houston(config-if)#clock rate 56000	Assigns the clock rate.
Houston(config-if)#no shutdown	Activates the interface.
Houston(config-if)#exit	Returns to global configuration mode.
Houston(config)#router ospf 1	Starts the OSPF routing process.
Houston(config-router)#network 172.16.0.0 0.0.255.255 area 0	Any interface with 172.16.*x.x* to be placed into OSPF area 0.
Houston(config-router)#exit	Returns to global configuration mode.
Houston(config)#router bgp 3	Starts the BGP routing process.
Houston(config-router)#no synchronization	Turns off route synchronization.
Houston(config-router)#neighbor 172.16.1.2 remote-as 3	Identifies a peer router at 172.16.1.2.
Houston(config-router)#neighbor 172.16.1.2 update-source loopback 0	Use any interface for TCP connections, as long as Loopback0 is configured.
Houston(config-router)#neighbor 172.16.20.1 remote-as 1	Identifies a peer router at 172.16.20.1.
Houston(config-router)#no auto- summary	Disables auto-summarization.
Houston(config-router)#exit	Returns to global configuration mode.
Houston(config)#exit	Returns to privileged mode.
Houston#copy running-config startup- config	Saves the configuration to NVRAM.

Laredo Router

Router(config)#hostname Laredo	Sets the router name to Laredo.
Laredo(config)#interface serial 0/0/1	Moves to interface configuration mode.
Laredo(config-if)#ip address 172.16.20.1 255.255.255.0	Assigns an IP address and netmask.
Laredo(config-if)#no shutdown	Activates the interface.
Laredo(config-if)#exit	Returns to global configuration mode.
Laredo(config)#router bgp 1	Starts the BGP routing process.
Laredo(config-router)#no synchronization	Turns off route synchronization.
Laredo(config-router)#neighbor 172.16.20.2 remote-as 3	Identifies a peer router at 172.16.20.2.
Laredo(config-router)#no auto-summary	Disables auto-summarization.

Laredo(config-router)#exit	Returns to global configuration mode.
Laredo(config)#exit	Returns to privileged mode.
Laredo#copy running-config startup- config	Saves the configuration to NVRAM.

Galveston Router

Router(config)#hostname Galveston	Sets router name to Galveston.
Galveston(config)#interface serial 0/0/0	Moves to interface configuration mode.
Galveston(config-if)#ip address 192.168.12.1 255.255.255.0	Assigns an IP address and netmask.
Galveston(config-if)#no shutdown	Activates the interface.
Galveston(config-if)#exit	Returns to global configuration mode.
Galveston(config)#router ospf 1	Starts the OSPF routing process.
Galveston(config-router)#network 192.168.12.0 0.0.0.255 area 0	Any interface with 192.168.12.x to be placed into OSPF Area 0.
Galveston(config-router)#exit	Returns to global configuration mode.
Galveston(config)#router bgp 2	Starts the BGP routing process.
Galveston(config-router)#neighbor 192.168.5.1 remote-as 3	Identifies a peer router at 192.168.5.1.
Galveston(config-router)#neighbor 192.168.5.1 ebgp-multihop 2	Two routers not directly connected need an eBGP session.
Galveston(config-router)#no auto-summary	Disables auto-summarization.
Galveston(config-router)#exit	Returns to global configuration mode.
Galveston(config)#exit	Returns to privileged mode.
Galveston#copy running-config startup- config	Saves the configuration to NVRAM.

Austin Router

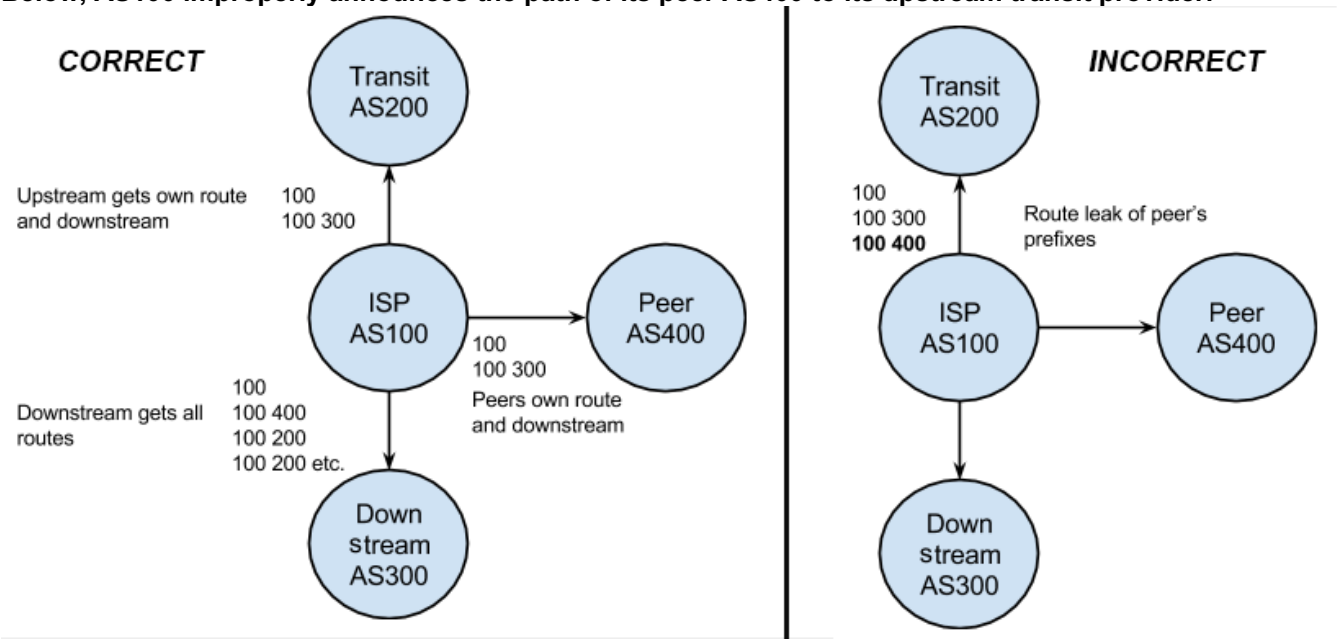
Router(config)#hostname Austin	Sets the router name to Austin.
Austin(config)#interface serial 0/0/0	Moves to interface configuration mode.
Austin(config-if)#ip address 192.168.5.1 255.255.255.0	Assigns an IP address and netmask.
Austin(config-if)#no shutdown	Activates the interface.
Austin(config-if)#interface fastethernet 0/1	Moves to interface configuration mode.
Austin (config-if)#ip address 172.16.1.2 255.255.255.0	Assigns an IP address and netmask.
Austin(config-if)#no shutdown	Activates the interface.
Austin(config-if)#exit	Returns to global configuration mode.
Austin(config)#router ospf 1	Starts the OSPF routing process.
Austin(config-router)#network 172.16.0.0 0.0.255.255 area 0	Any interface with 172.16.*x*.*x* to be placed into OSPF area 0.
Austin(config-router)#network 192.168.5.0 0.0.0.255 area 0	Any interface with 192.168.5.*x* to be placed into OSPF area 0.
Austin(config-router)#exit	Returns to global configuration mode.
Austin(config)#router bgp 3	Starts the BGP routing process.
Austin(config-router)#no synchronization	Turns off route synchronization.
Austin(config-router)#neighbor 172.16.2.254 remote-as 3	Identifies a peer router at 172.16.2.254.
Austin(config-router)#neighbor 192.168.12.1 remote-as 2	Identifies a peer router at 192.168.12.1.
Austin(config-router)#neighbor 192.168.12.1 ebgp-multihop 2	Two routers not directly connected need an eBGP session.
Austin(config-router)#no auto-summary	Turns off auto-summarization.
Austin(config-router)#exit	Returns to global configuration mode.
Austin(config)#exit	Returns to privileged mode.
Austin#copy running-config startup- config	Saves the configuration to NVRAM.

BGP Route Leaks and BGP Hijacking

Route leaks and hijacks are both where illegitimate prefixes/address blocks are wrongly propagated
BGP is vulnerable since built on a system of trust (and the nature of TCP)

- illegitimate advertisement of prefixes, blocks of IP addresses
- propagate across networks and lead to incorrect or suboptimal routing.
- Can happen from an AS originating a prefix that it does not actually own
- Also happens when AS announces it can deliver traffic through a route that should not exist
- Are particularly prone to propagation when
    - a more specific prefix is advertised (BGP prefers the most specific block of addresses)
    - a path is advertised that is shorter than the currently available paths (BGP prefers shortest AS Path).
- Usually happen when BGP advertisements are not properly filtered using the no-export community.

Below, AS100 improperly announces the path of its peer AS400 to its upstream transit provider.



Bigger real-world examples: https://blog.thousandeyes.com/finding-and-diagnosing-bgp-route-leaks/

Remediation

- Maintain communication and work with administrators of neighboring AS'es to collaborate on fixes
- BGP-advertise routes more preferable than the leaked route (a more specific prefix length and/or shortest path)
- For example, if the problematic route advertises a /17, counter with advertising an /18
- Advertising a shorter path is not as effective
- Last resort: consider changing your prefixes entirely by changing DNS records. Only feasible if alternate locations are available (i.e., other data center). The TTL on DNS records also becomes an issue.

Preventative measures

- publish Route Origin Authorizations (ROAs) in the various regional Internet registries (RIRs)
- verifies that a given origin AS is authorized to announce its prefixes, (incl. max prefix lengths)
- networks using Resource Certification (RPKI) can validate the origin AS and verify routes legitimacy

Best practices: Proper route filtering uses a set of robust filtering rules, likely including:

- Filter out Bogon prefixes and routes with Bogon ASNs anywhere in the AS path. Reserved or unallocated IP spaces should never be advertised.
- Filter out routes with more than two Tier 1 ("transit-free") networks in the AS path. When more are present, at least one of the networks is providing transit to another.
- If you don't sell transit to large networks (like Tier 1 networks), filter out routes from them in the AS path. Use with a whitelist of prefixes that each of your customers may announce to you.
- Use peer locking- ask peers for all possible upstream networks, and only allow those as intermediate networks (See Snijders NANOG67 - https://www.nanog.org/sites/default/files/Snijders_Everyday_Practical_Bgp.pdf
- Use BGP Maximum-Prefix to set the maximum number of prefixes that can be announced from your peers. This helps during an episode of rapid propagation of problematic AS'es.