

BGP (Border Gateway Protocol)

- CCNP Routing & Switching (prezentacje)
- Rick Graziani
- Homer Simpson
- Łukasz Sturgulewski

<http://www.inetdaemon.com/tutorials/internet/ip/routing/bgp/>

http://www.inetdaemon.com/tutorials/internet/ip/routing/bgp/autonomous_system_number.shtml

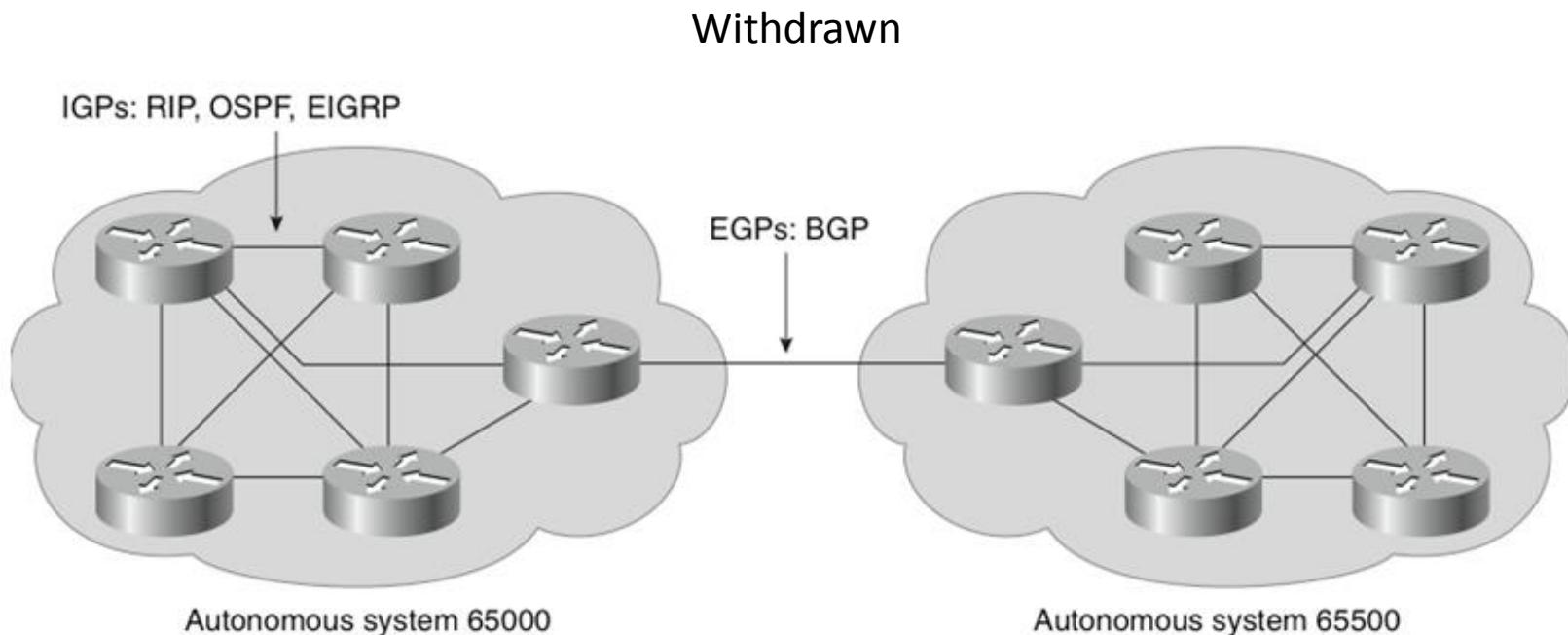
Terms

- **IGP** (Interior Gateway Protocol) - RIP, IGRP, EIGRP, OSPF = Routing protocol used to exchange routing information within an autonomous system.
- **EGP** (Exterior Gateway Protocol) - BGP = Routing protocol used to exchange routing information between autonomous systems.
- **Autonomous System** = (From RFC 1771) “A set of routers under the single technical administration, using an IGP and common metrics to route packets within the AS, and using an EGP to route packets to other AS’s.”
- **BGP** is a path vector or an advanced distance vector routing protocol.

IGP vs EGP

IGP versus EGP

- **Interior gateway protocol (IGP)**
 - A routing protocol operating within an Autonomous System (AS).
 - RIP, OSPF, and EIGRP are IGPs.
- **Exterior gateway protocol (EGP)**
 - A routing protocol operating between different AS.
 - BGP is an interdomain routing protocol (IDRP) and is an EGP.



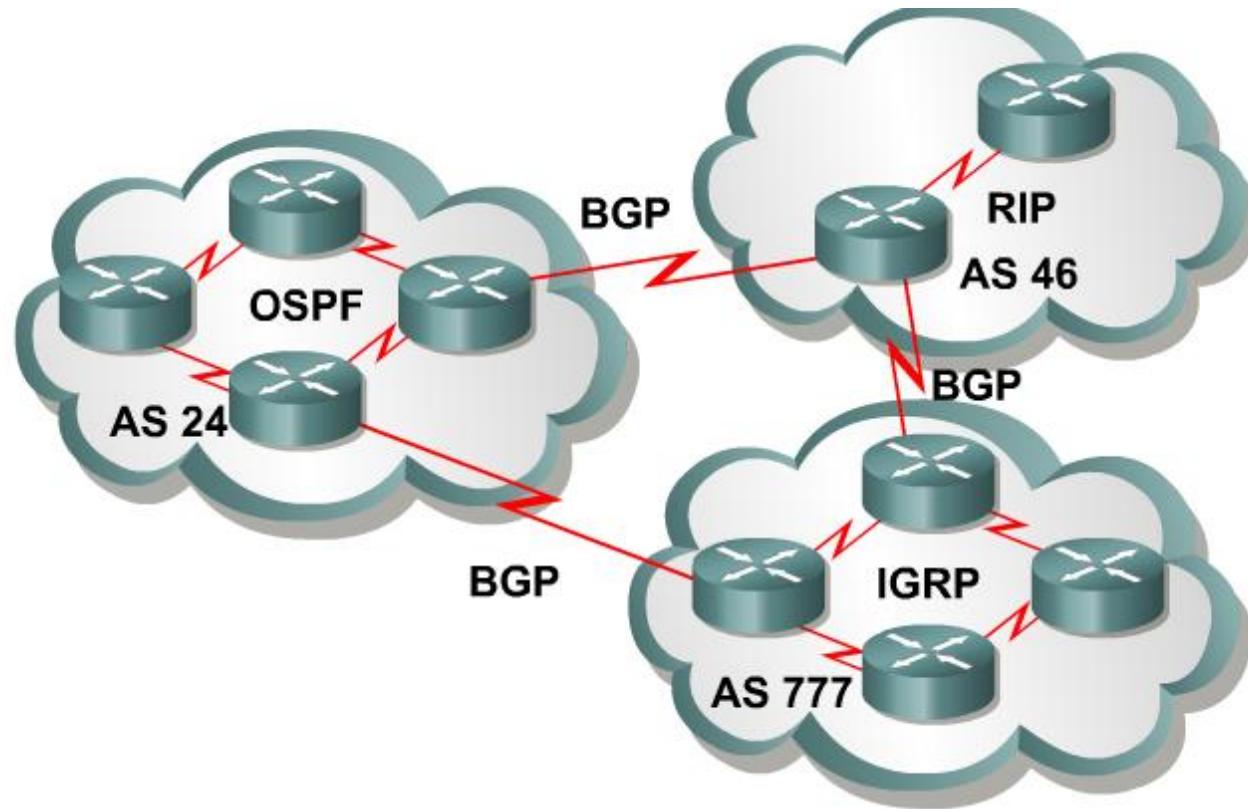
IGP versus EGP

- BGP works differently than IGPs because it does not make routing decisions based on best path metrics.
 - Instead, BGP is a policy-based routing protocol that allows an AS to control traffic flow using multiple BGP attributes.
- Routers running BGP exchange network attributes including a list of the full path of BGP AS numbers that a router should take to reach a destination network.
- BGP allows an organization to fully use all of its bandwidth by manipulating these path attributes.

IGP versus EGP

Protocol	Interior or Exterior	Type	Hierarchy Required?	Metric
RIP	Interior	Distance vector	No	Hop count
OSPF	Interior	Link state	Yes	Cost
IS-IS	Interior	Link state	Yes	Metric
EIGRP	Interior	Advanced distance vector	No	Composite
BGP	Exterior	Path vector	No	Path vectors (attributes)

Autonomous Systems (AS)



EGPs, such as BGP, are used to interconnect autonomous systems.

Autonomous Systems (AS)

- An AS is a group of routers that share similar routing policies and operate within a single administrative domain.
- An AS typically belongs to one organization.
 - A single or multiple interior gateway protocols (IGP) may be used within the AS.
 - In either case, the outside world views the entire AS as a single entity.
- If an AS connects to the public Internet using an exterior gateway protocol such as BGP, then it must be assigned a unique AS number which is managed by the Internet Assigned Numbers Authority (IANA).

IANA

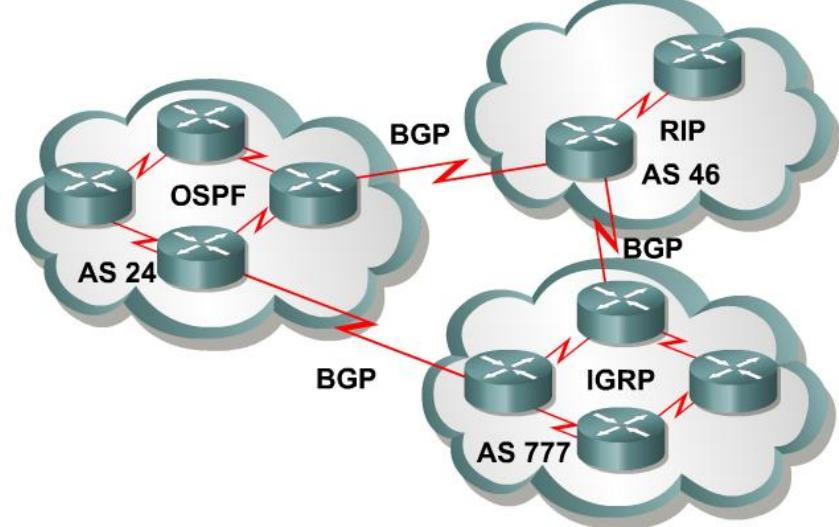
- The IANA is responsible for allocating AS numbers through five Regional Internet Registries (RIRs).
 - RIRs are nonprofit corporations established for the purpose of administration and registration of IP address space and AS numbers in key geographic locations.



Regional Internet Registries (RIRs)

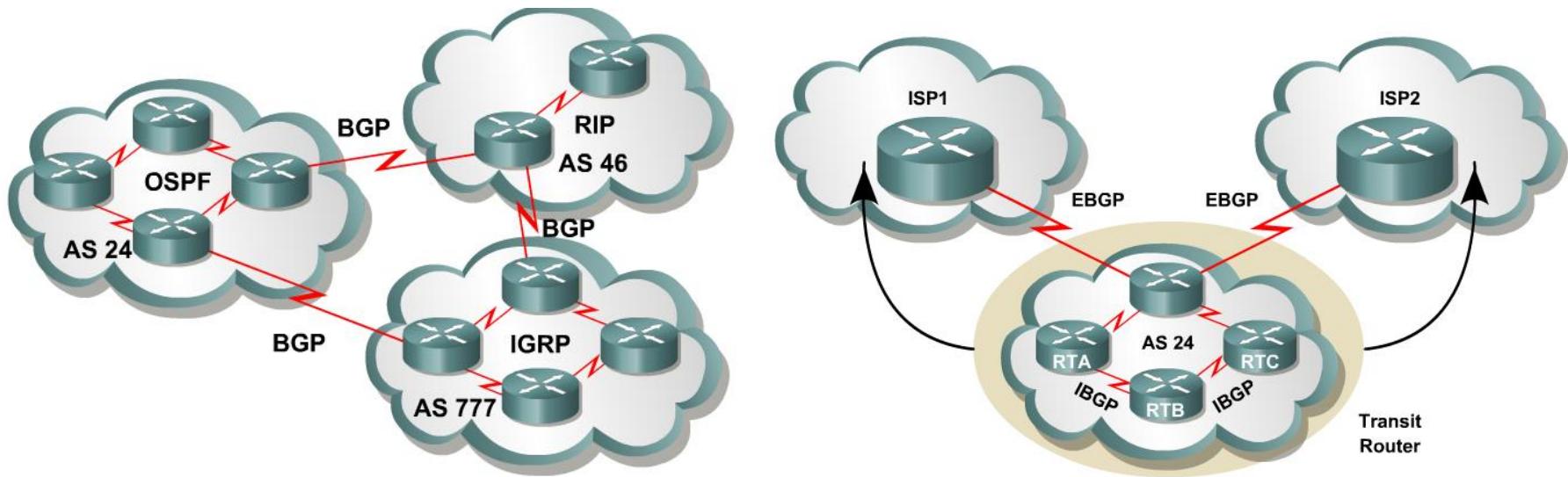
RIR Name	Geographic Coverage	Link
AfriNIC	Continent of Africa	www.afrinic.net
APNIC (Asia Pacific Network Information Centre)	Asia Pacific region	www.apnic.org
ARIN (American Registry for Internet Numbers)	Canada, the United States, and several islands in the Caribbean Sea and North Atlantic Ocean	www.arin.net
LACNIC (Latin America and Caribbean Internet Addresses Registry)	Central and South America and portions of the Caribbean	www.lacnic.net
RIPE (Réseaux IP Européens)	Europe, the Middle East, and Central Asia	www.ripe.net

AS Numbers



- AS numbers can be between **1** to **65,535**.
 - RIRs manage the AS numbers between **1** and **64,512**.
 - The **64,512 - 65,535** numbers are reserved for private use (similar to IP Private addresses).
 - The IANA is enforcing a policy whereby organizations that connect to a single provider use an AS number from the private pool.
- **Note:**
 - The current AS pool of addresses is predicted to run out by 2012.
 - For this reason, the IETF has released RFC 4893 and RFC 5398.
 - These RFCs describe BGP extensions to increase the AS number from the two-octet (16-bit) field to a four-octet (32-bits) field, increasing the pool size from **65,536** to **4,294,967,296** values.

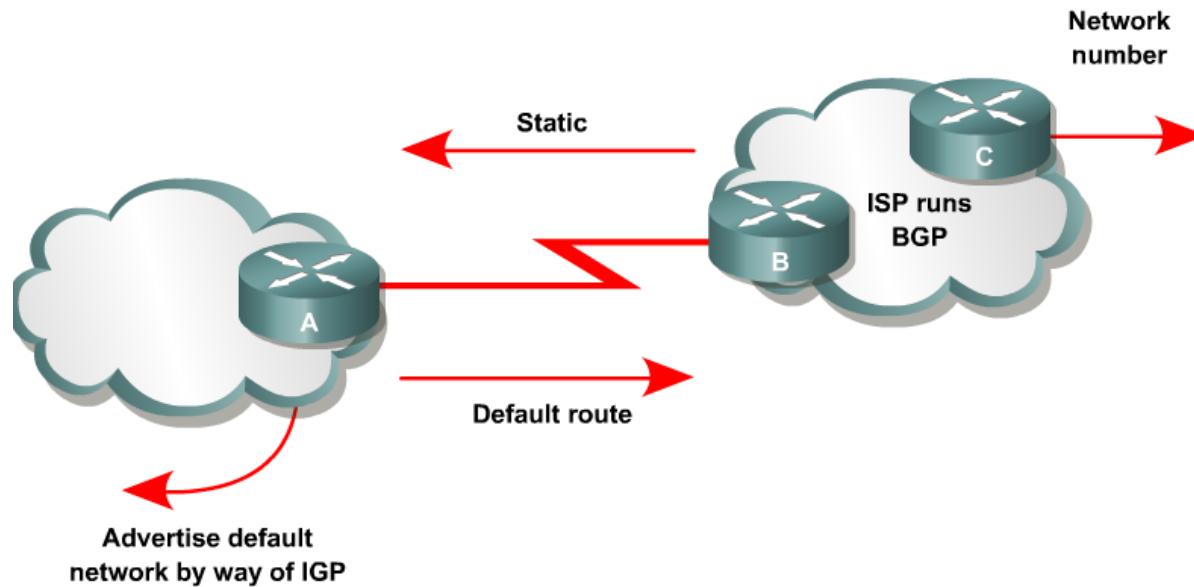
When to use BGP and when not to use BGP



Use BGP when the effects of BGP are well understood and one of the following conditions exist:

- The AS allows packets to transit through it to reach another AS (transit AS).
- The AS has multiple connections to other AS's.
- The flow of traffic entering or exiting the AS must be manipulated.
This is policy based routing and based on attributes.

When to use BGP and when not to use BGP



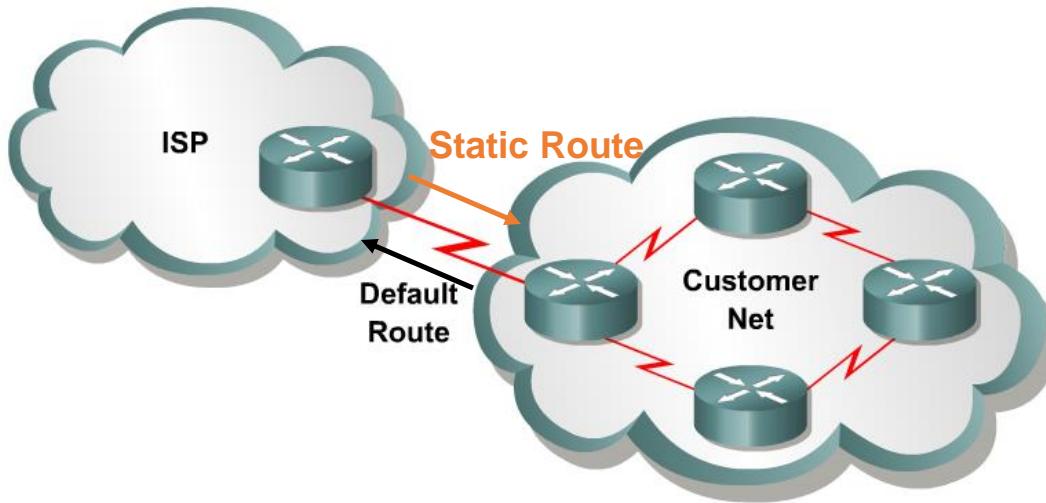
Do not use BGP if you have one or more of the following conditions:

- A single connection to the Internet or another AS
- No concern for routing policy or routing selection
- A lack of memory or processing power on your routers to handle constant BGP updates
- A limited understanding of route filtering and BGP path selection process
- Low bandwidth between AS's

Connection Redundancy

Connecting to One ISP	Connecting to Two or more ISPs
<p>Single-homed</p> <p>Company A</p> <p>ISP</p> <p>Internet</p>	<p>Multihomed</p> <p>Company A</p> <p>ISP 1</p> <p>ISP 2</p> <p>Internet</p>
<p>Dual-homed</p> <p>Option 1</p> <p>Company A</p> <p>ISP</p> <p>Internet</p> <p>Option 2</p> <p>Company A</p> <p>ISP</p> <p>Internet</p>	<p>Dual-multihomed</p> <p>Company A</p> <p>ISP 1</p> <p>ISP 2</p> <p>Internet</p>

Single-homed autonomous systems



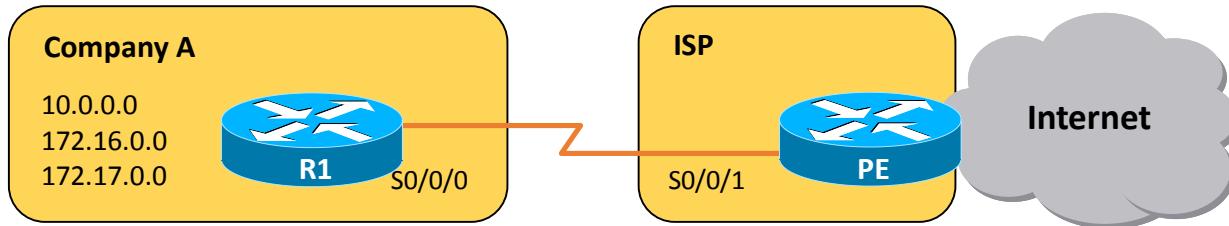
A single-homed AS can be configured with a default route to reach outside networks.

- If an AS has only **one exit point** to outside networks, it is considered a **single-homed system**.
- Single-homed autonomous systems are often referred to as **stub** networks or stubs.
- Stubs can rely on a **default route** to handle all traffic destined for non-local networks.
- BGP is **not** normally needed in this situation.

Using Static Routes Example

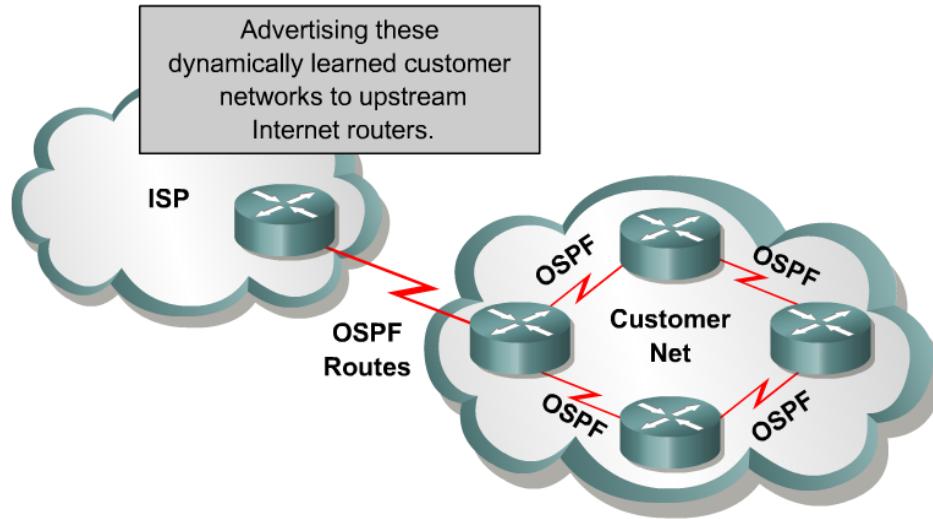
- Static routes are the simplest way to implement routing with an ISP.
 - Typically a customer has a single connection to an ISP and the customer uses a default route toward the ISP while the ISP deploys static routes toward the customer.

```
R1(config)# router eigrp 110
R1(config-router)# network 10.0.0.0
R1(config-router)# exit
R1(config)# ip default-network 0.0.0.0
R1(config)# ip route 0.0.0.0 0.0.0.0 serial 0/0/0
```



```
PE(config)# ip route 10.0.0.0 255.0.0.0 serial 0/0/1
PE(config)# ip route 172.16.0.0 255.255.0.0 serial 0/0/1
PE(config)# ip route 172.17.0.0 255.255.0.0 serial 0/0/1
```

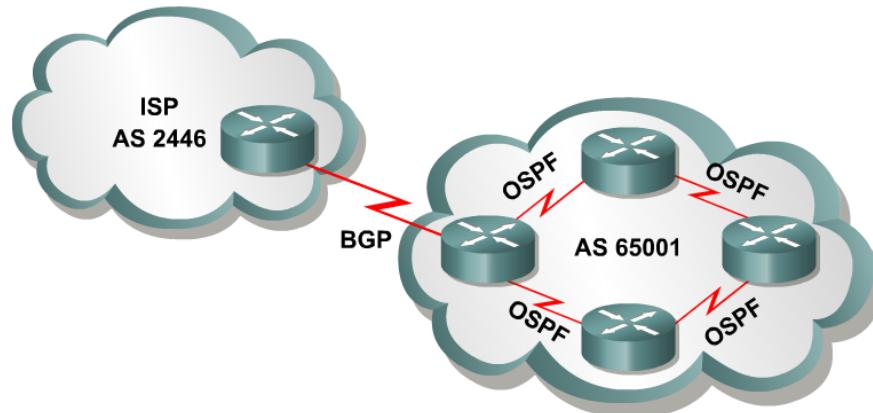
Single-homed autonomous systems



A provider may choose to dynamically learn customer routes using an IGP, such as OSPF.

- Use an IGP – Both the provider and the customer use an **IGP** to share information regarding the customer's networks.
- This provides the benefits associated with dynamic routing.
- BGP is not normally needed in this situation.

Single-homed autonomous systems

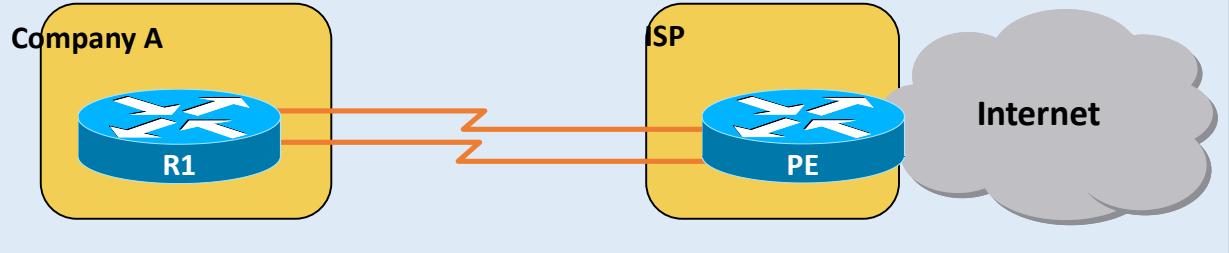


A provider may also choose to dynamically learn a customer's routes using BGP, which typically runs between the ISP router and the customer's boundary router.

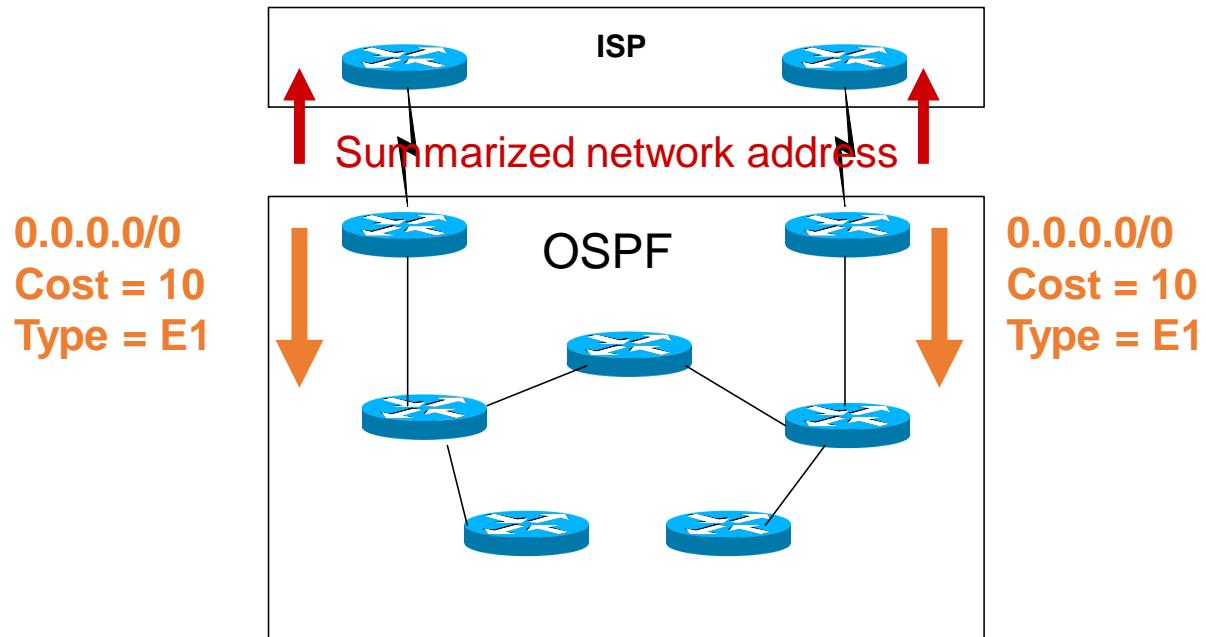
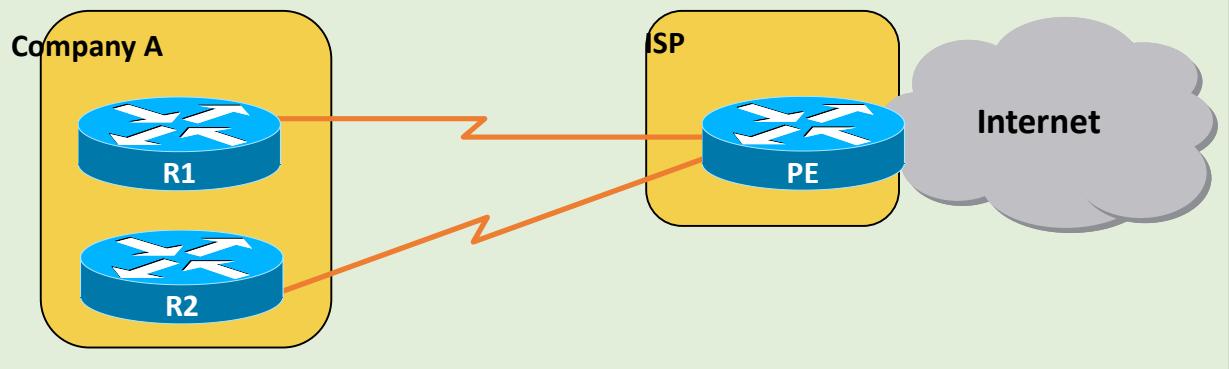
- **Use an EGP** – The third method by which the ISP can learn and advertise the customer's routes is to use an EGP such as BGP.
- In a **single-homed autonomous system** the *customer's routing policies are an extension of the policies of the provider*.
 - For this reason the Internet number registries are unlikely to assign an AS number.
 - Instead, the provider can give the customer an AS number from the **private** pool of AS numbers, 64,512 to 65,535.
 - The provider will strip off these numbers when advertising the customer's routes towards the core of the Internet.

Dual-Homed

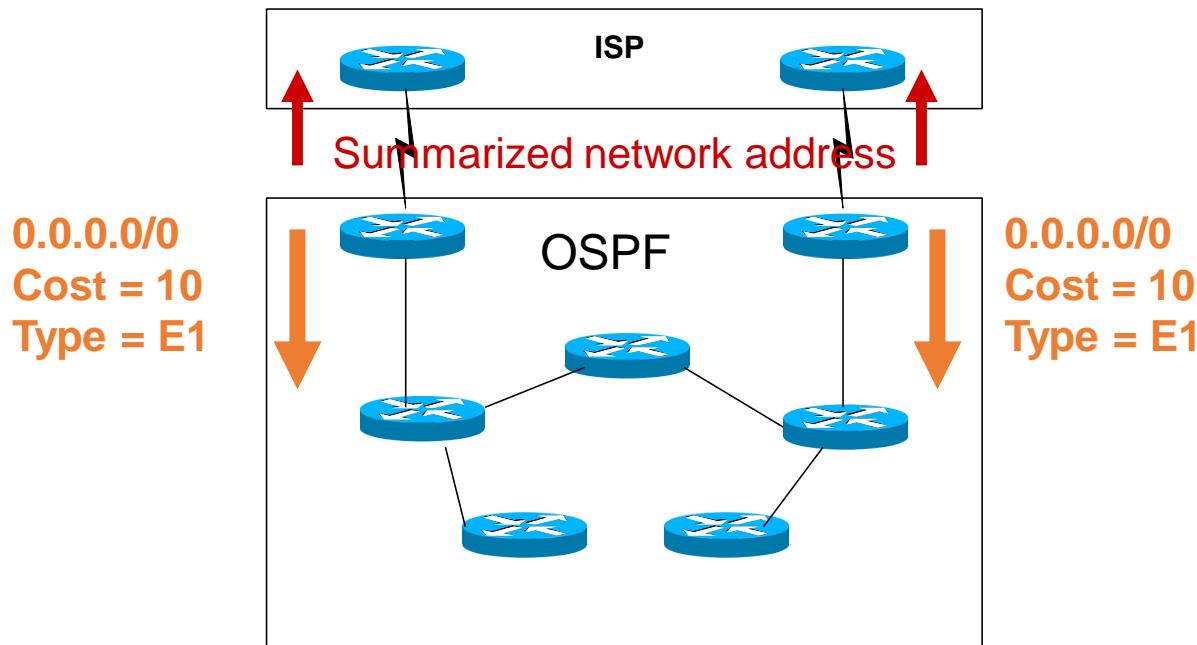
Option 1:



Option 2:

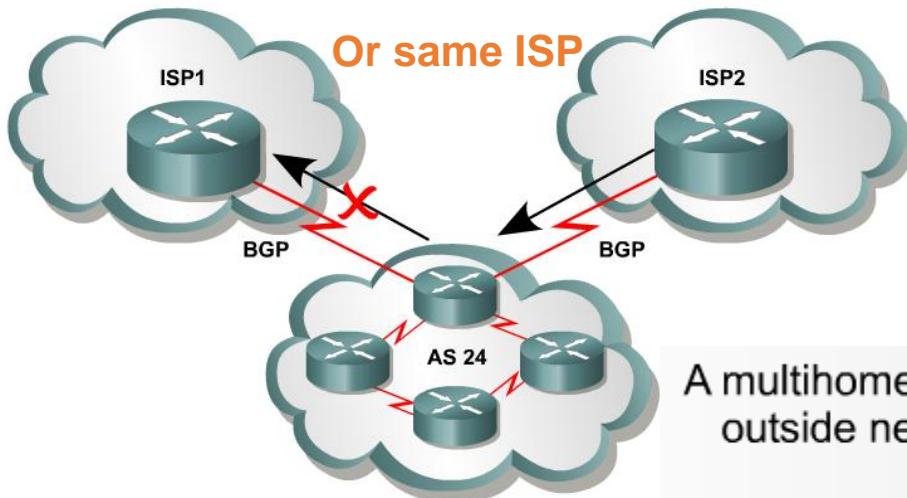


Dual-Homed to a Single Autonomous Systems



- This is an improved topology over Single-Home AS, providing for **redundancy**.
- One option may be to use one link as the **primary** link and the other as a **backup** link.
- A better design would be to **use both paths**, with each one providing backup for the other in the event of link or router failure.
- In most cases this will be sufficient for good internetwork performance.

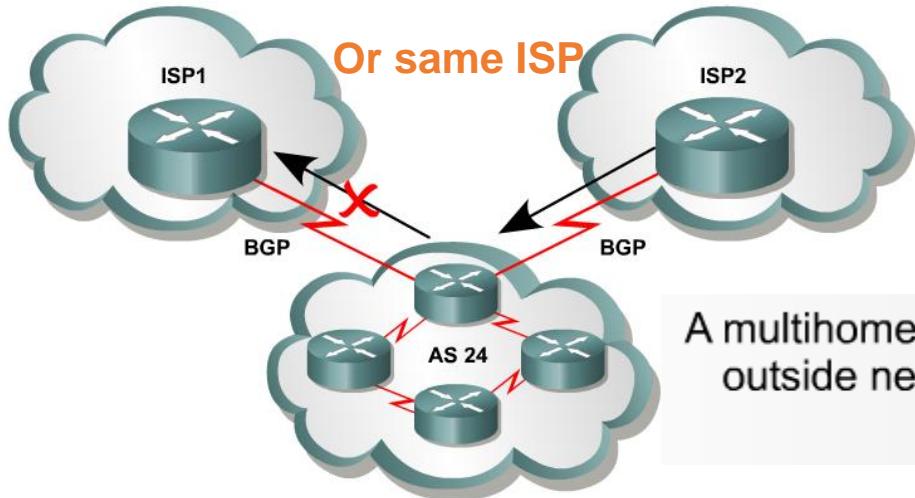
Multihomed nontransit autonomous systems



A multihomed nontransit AS features more than one exit point to outside networks, but does not allow traffic to pass from one outside connection to another.

- An AS is a **multihomed system** if it has *more than one exit point* to outside networks.
- A **non-transit AS** does not allow transit traffic—that is, any traffic that has a source and destination outside the AS—to pass through it.
- A **non-transit AS** would advertise only its *own* routes to both the providers it connects to—it would not advertise routes it learned from one provider to another.
- This makes certain that ISP1 will not use AS 24 to reach destinations that belong to ISP2, and ISP2 would not use AS 24 to reach destinations that belong to ISP1.

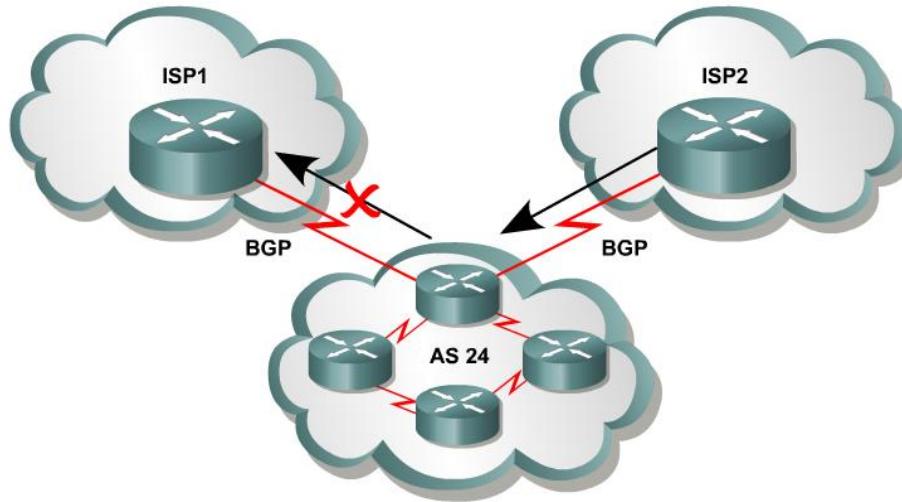
Multihomed nontransit autonomous systems



A multihomed nontransit AS features more than one exit point to outside networks, but does not allow traffic to pass from one outside connection to another.

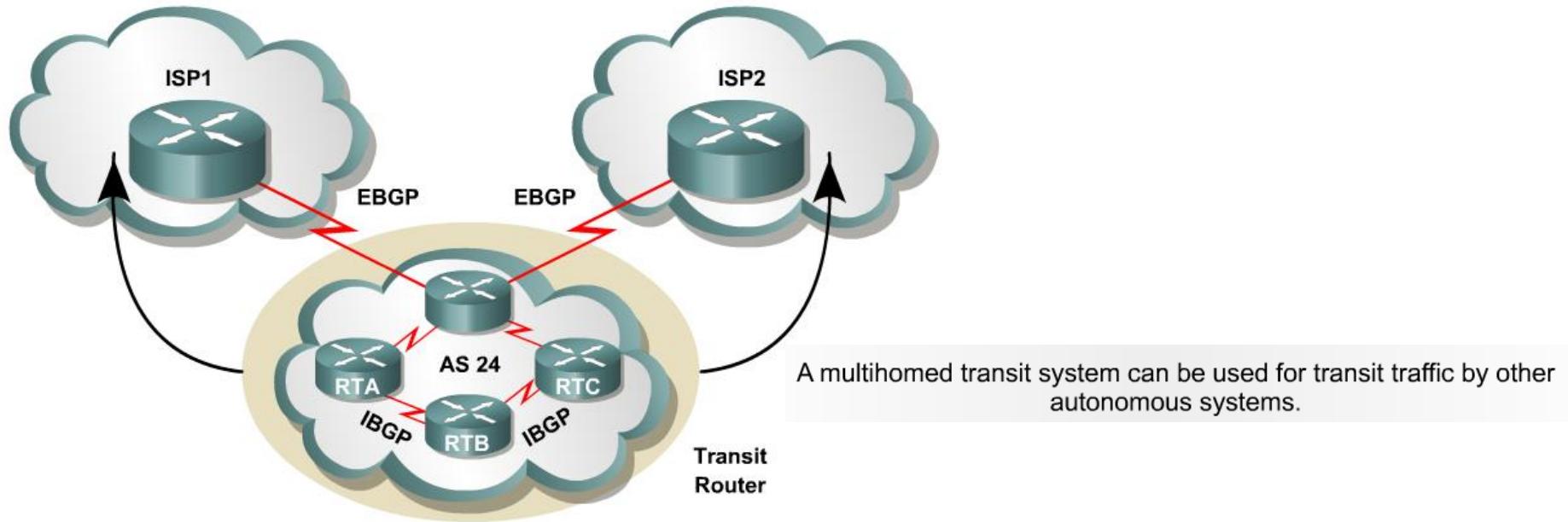
- Multihomed nontransit autonomous systems do not really need to run BGP4 with their providers.
- It is usually **recommended** and **often required** by ISPs.
- As it will be seen later in this module, BGP4 offers numerous advantages, including increased control of route propagation and filtering.

Multihomed nontransit autonomous systems



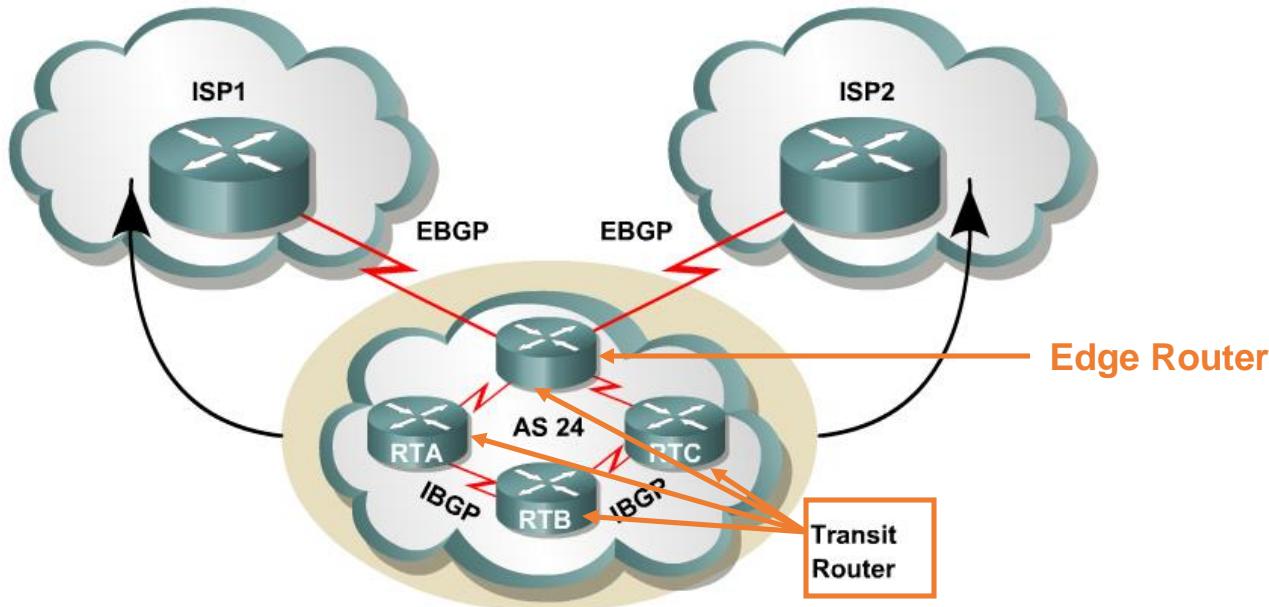
- ***Incoming route advertisements influence your outgoing traffic, and outgoing advertisements influence your incoming traffic.***
- If the provider advertises routes into your AS via BGP, your internal routers have more accurate information about external destinations.
 - BGP also provides tools for setting routing policies for external destinations.
- If your internal routes are advertised to the provider via BGP, you have influence over which routes are advertised at which exit point.
 - BGP also provides tools for your influencing (to some degree) the choices the provider makes when sending traffic into your AS.

Multi-homed Transit Autonomous Systems



- A multi-homed **transit system** has more than one connection to the outside world and can be used for transit traffic by other autonomous systems.
 - From the point of view of the multi-homed AS, transit traffic is any traffic originating from outside sources bound for outside destinations

Multi-homed Transit Autonomous Systems

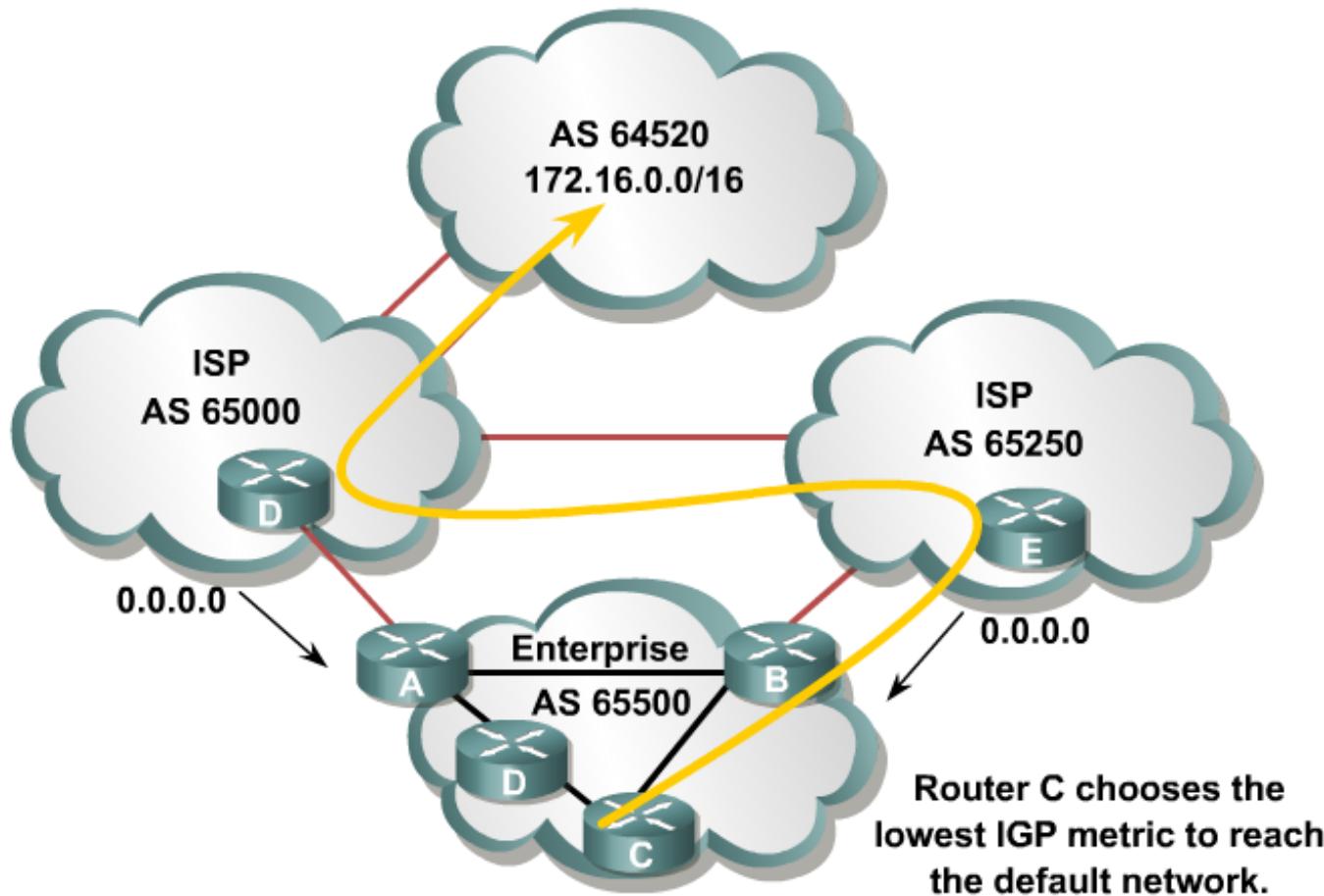


- When BGP is running inside an AS, it is referred to as **Internal BGP (IBGP)**.
- When BGP runs between autonomous systems, it is called **External BGP (EBGP)**.
- If the role of a BGP router is to route IBGP traffic, it is called a **transit router**.
- Routers that sit on the boundary of an AS and that use EBGP to exchange information with the ISP are called **border or edge routers**.

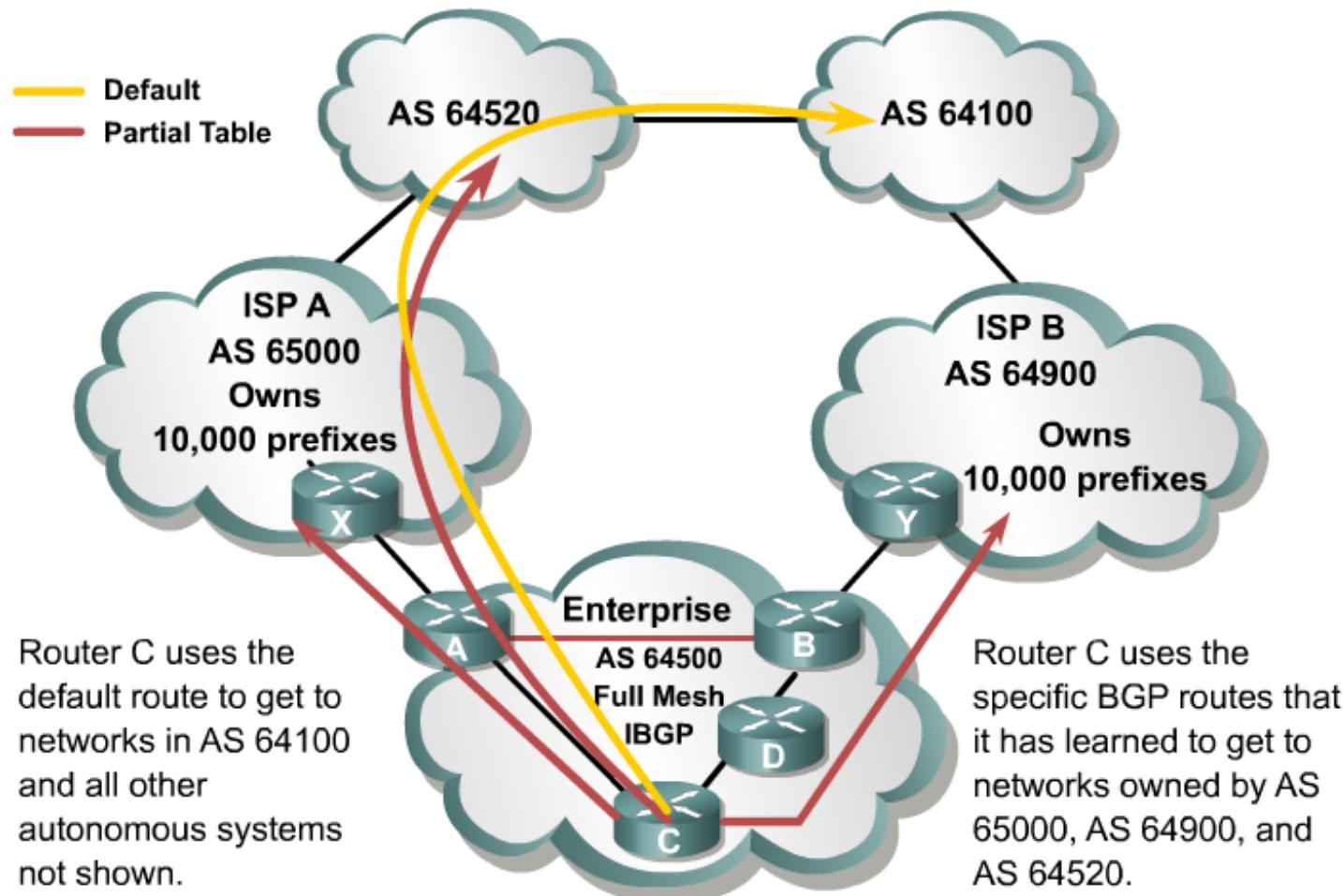
Three Multihoming Connection Options

1. Each ISP passes only a default route to the AS.
 - The default route is passed on to internal routers.
2. Each ISP passes only a default route and provider-owned specific routes to the AS.
 - These routes may be propagated to internal routers, or all internal routers in the transit path can run BGP to exchange these routes.
3. Each ISP passes all routes to the AS.
 - All internal routers in the transit path run BGP to exchange these routes.

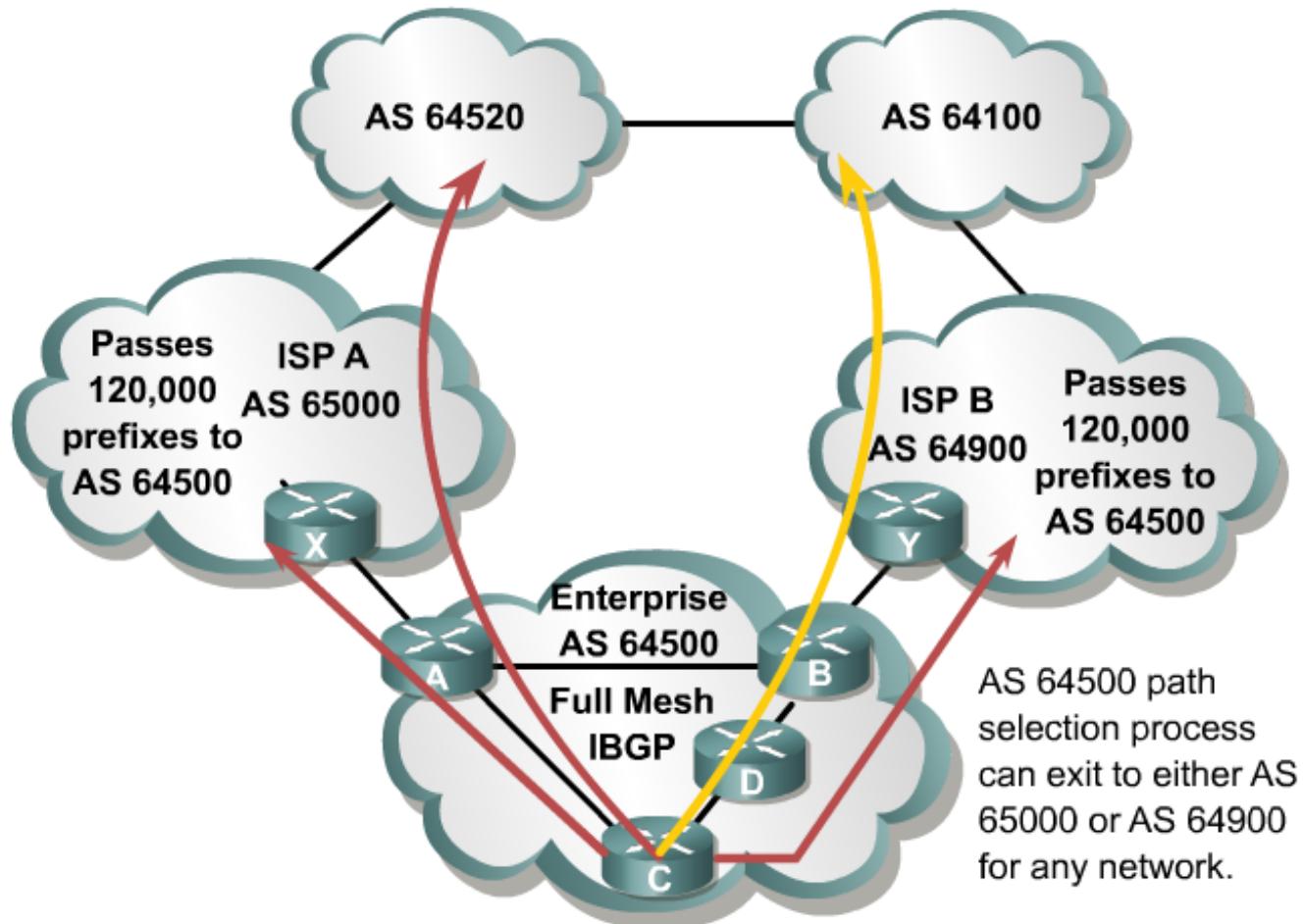
Default Routes from All Providers



Default Routes and Partial Updates



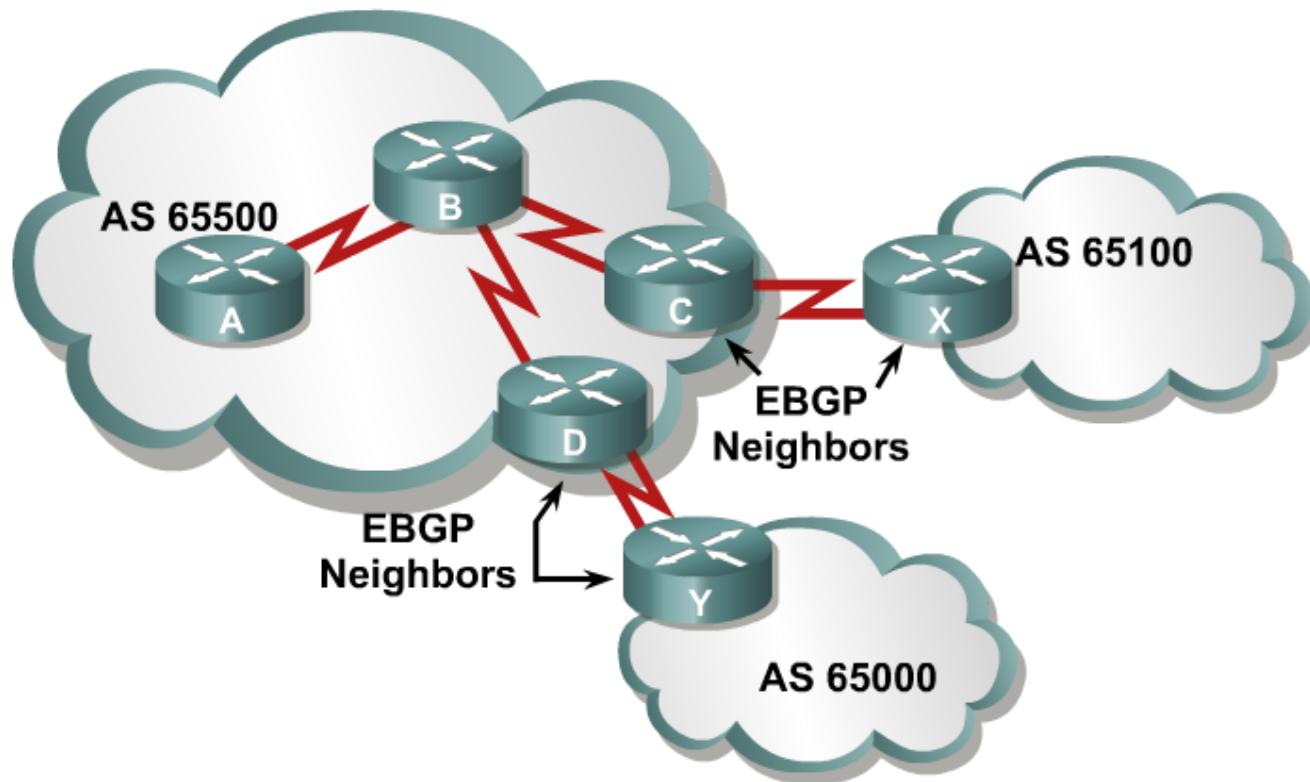
Full Routes from All Providers



EBGP vs IBGP

External BGP

- EBGP neighbors are in different autonomous systems.
 - EBGP neighbors need to be directly connected.

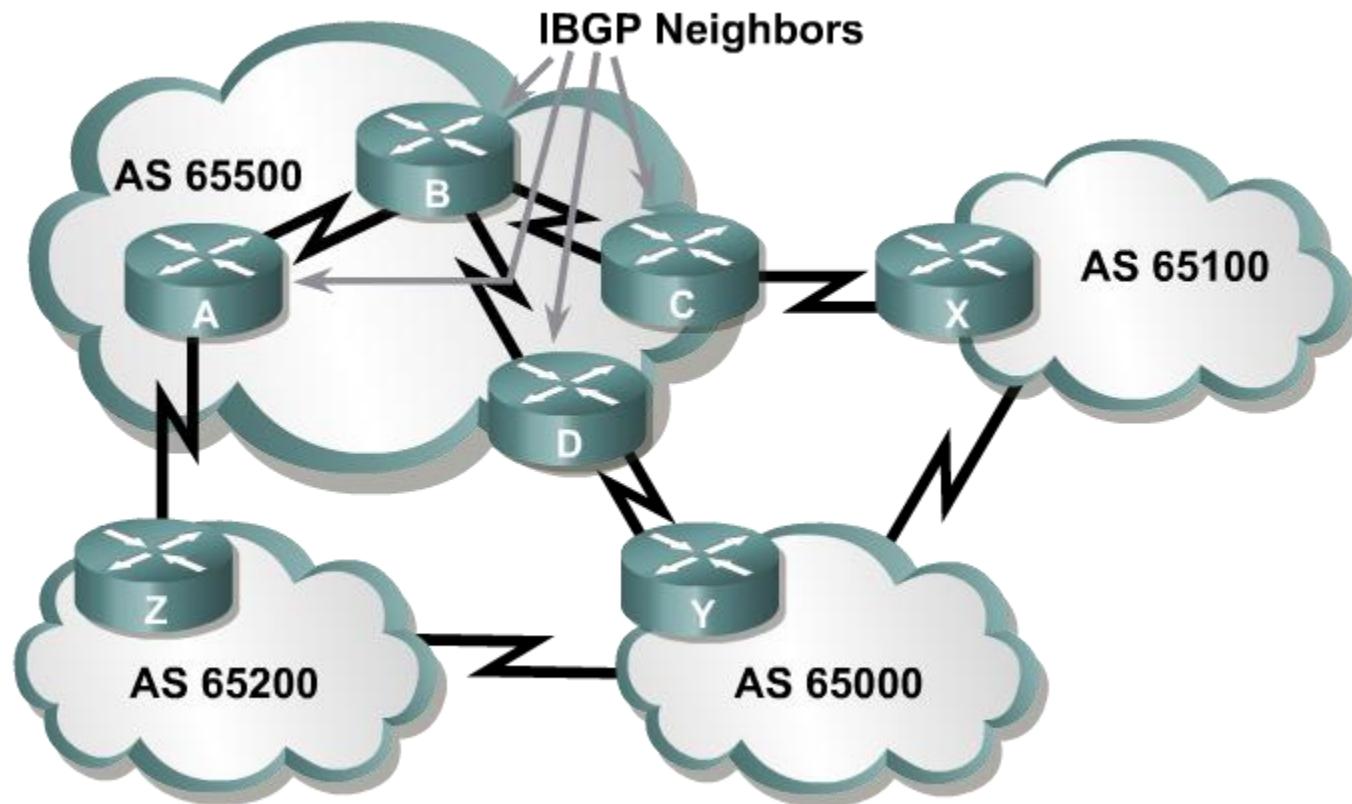


EBGP Neighbors Relationship Requirements

- Define neighbors:
 - A TCP session (three-way handshake) must be established before starting BGP routing update exchanges.
- Reachability:
 - EBGP neighbors are usually directly connected.
- Different AS number:
 - EBGP neighbors must have different AS numbers.

Internal BGP

- IBGP neighbors are in the same autonomous systems.
 - IBGP neighbors do not need to be directly connected.

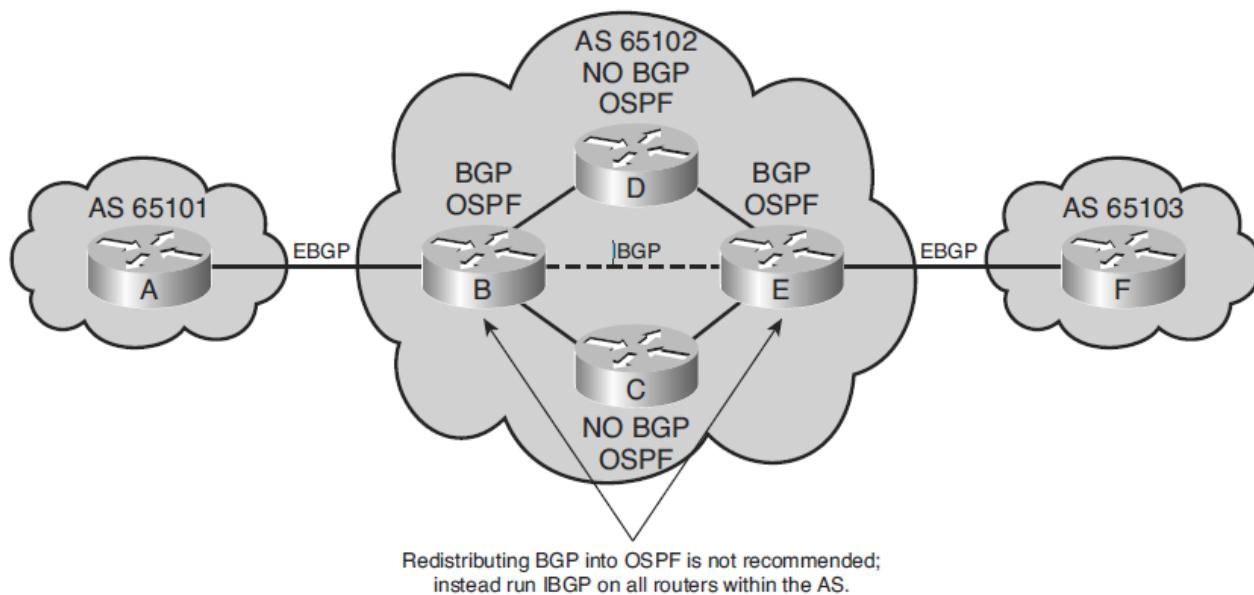


IBGP Neighbors Relationship Requirements

- **Define neighbors:**
 - A TCP session (three-way handshake) must be established before starting BGP routing update exchanges.
- **Reachability:**
 - IBGP neighbors must be reachable usually by using an IGP.
 - Loopback IP addresses are typically used to identify IBGP neighbors.
- **Same AS number:**
 - IBGP neighbors must have the same AS number.

IIGP in a Transit AS

- A transit AS is an AS that routes traffic from one external AS to another external AS.
 - Transit AS network are typically ISPs.
- All routers in a transit autonomous system must have complete knowledge of external routes.

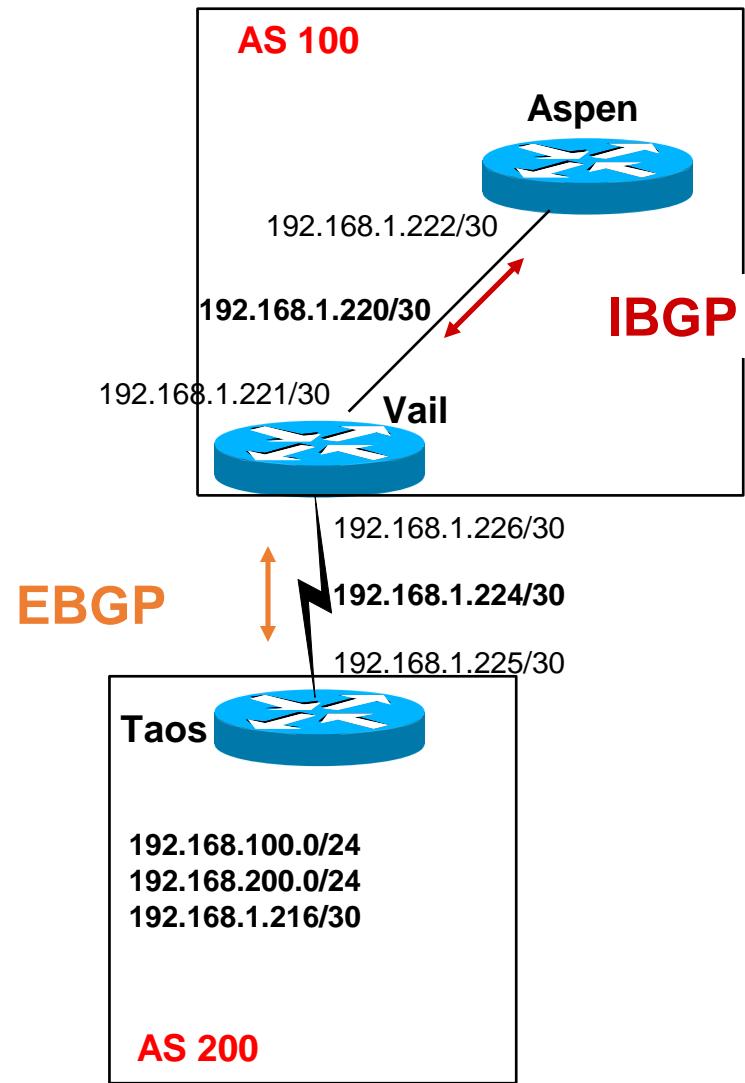


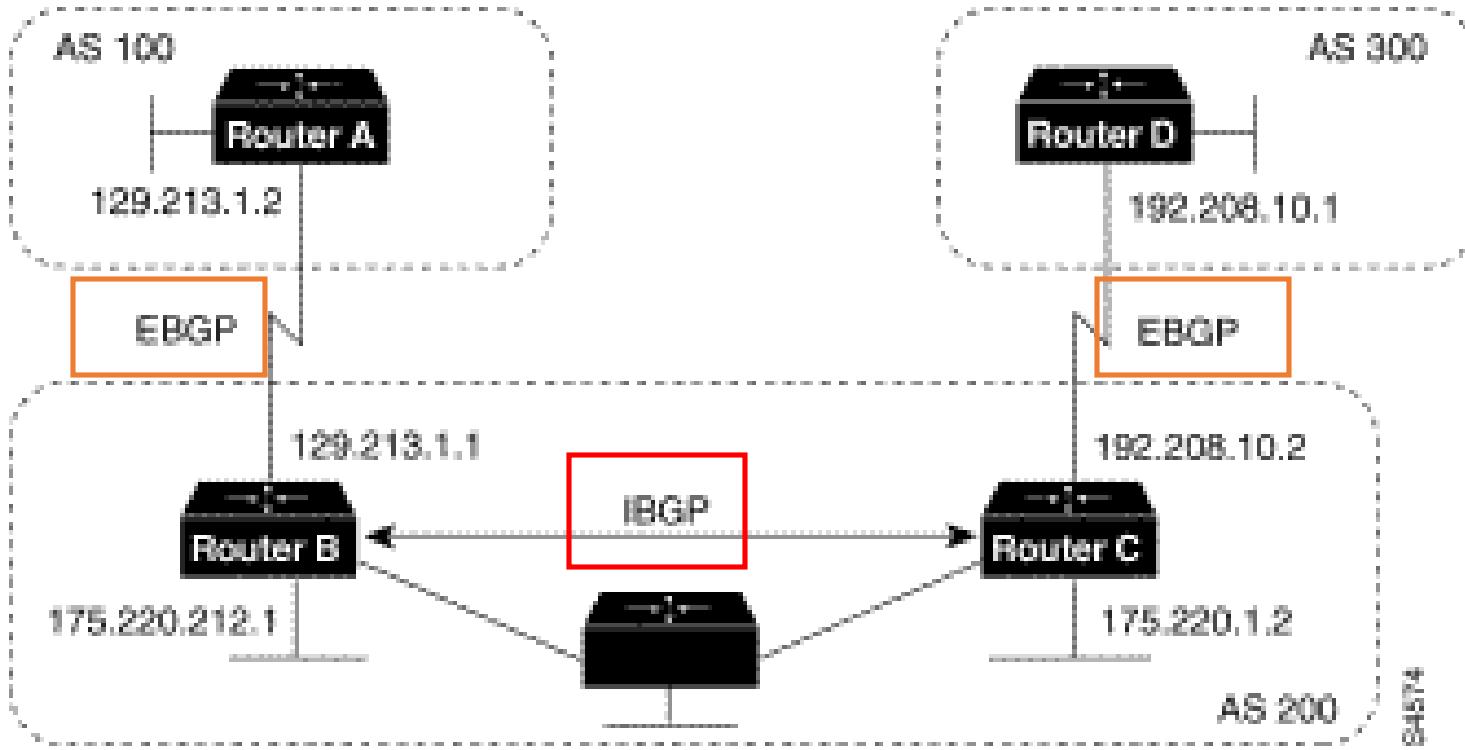
IBGP in a Nontransit AS

- A nontransit AS is an AS that does not route traffic from one external AS to another external AS.
 - Nontransit AS networks are typically enterprise networks.
- All routers in a nontransit autonomous system must still have complete knowledge of external routes.
- To avoid routing loops within an AS, BGP specifies that routes learned through IBGP are never propagated to other IBGP peers.
 - It is assumed that the sending IBGP neighbor is fully meshed with all other IBGP speakers and has sent each IBGP neighbor the update.

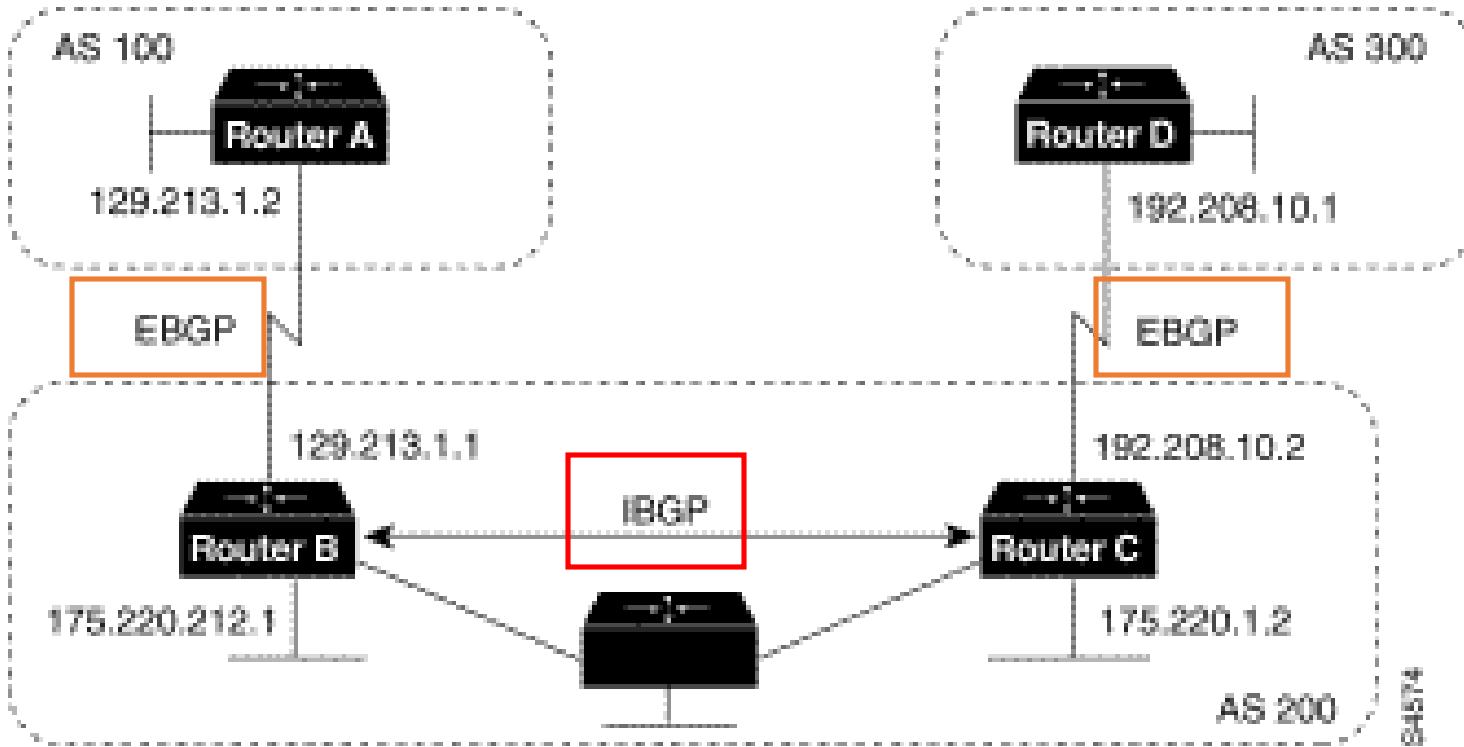
IBGP vs EBGP

- When BGP is running inside an AS, it is referred to as **Internal BGP (IBGP)**.
 - If a BGP router's role is to route IBGP traffic, it is called a transit router.
- When BGP runs between autonomous systems, it is called **External BGP (EBGP)**.
 - Routers that sit on the boundary of an AS and use EBGP to exchange information with the ISP are called border routers.
- “With very few exceptions, interior BGP (IBGP) – BGP between peers in the same AS – is used only in multihomed scenarios.” – Doyle





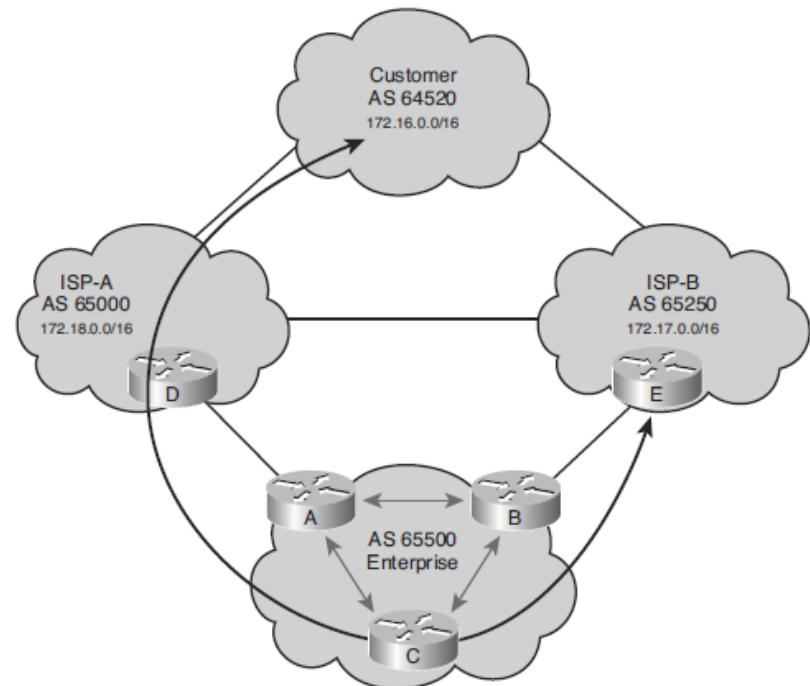
- Routers A and B are running **EBGP (BGP)**, and Routers B and C are running **IBGP**.
- Note that the **EBGP (BGP)** peers are directly connected and that the **IBGP** peers are not. (They can be.)
- As long as there is an **IGP** running that allows the two neighbors to reach one another, IBGP peers do not have to be directly connected.
- ***More later!***



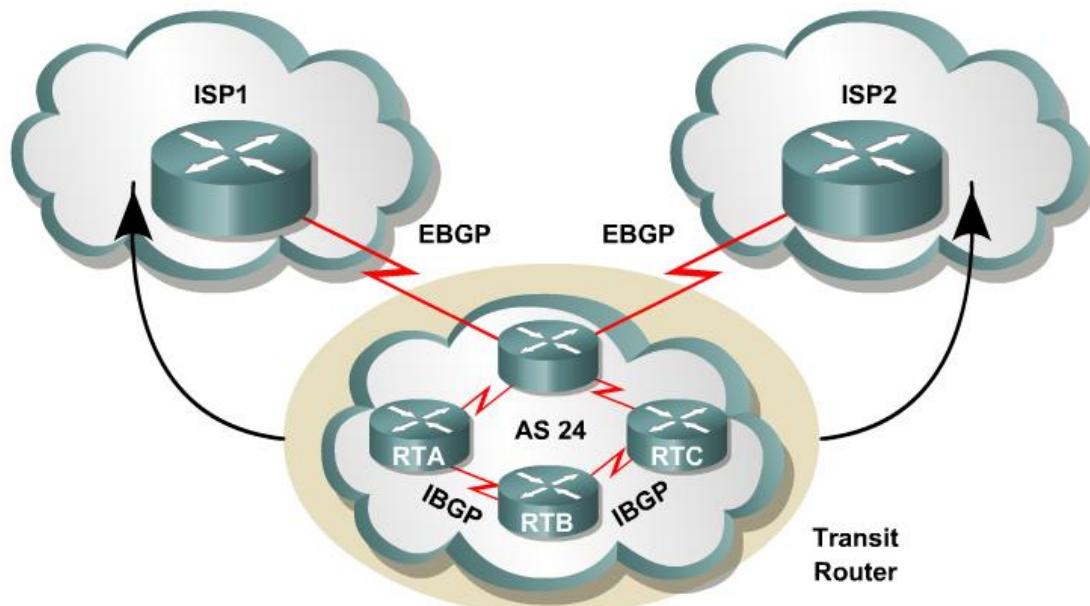
- All **BGP** speakers within an AS must establish a peer relationship with each other, that is, the **BGP** speakers within an AS must be fully meshed logically. (later)
- BGP4 provides two techniques that alleviate the requirement for a logical full mesh: confederations and route reflectors. (later)
- AS 200 is a ***transit AS*** for AS 100 and AS 300---that is, AS 200 is used to transfer packets between AS 100 and AS 300.

BGP in an Enterprise Example

- Enterprise AS 65500 is learning routes from both ISP-A and ISP-B via EBGP and is also running IBGP on all of its routers.
 - If one of the connections to the ISPs goes down, traffic will be sent through the other ISP.
- An undesirable situation could occur if the enterprise AS is configured as a transit AS.
 - For example, AS 65500 learns the 172.18.0.0/16 route from ISP-A.
 - If router B advertises that route to ISP-B, then ISP-B may decide to use it.
 - This undesirable configuration could be avoided through careful BGP configuration.



BGP Hazards – Doyle, Routing TCP/IP



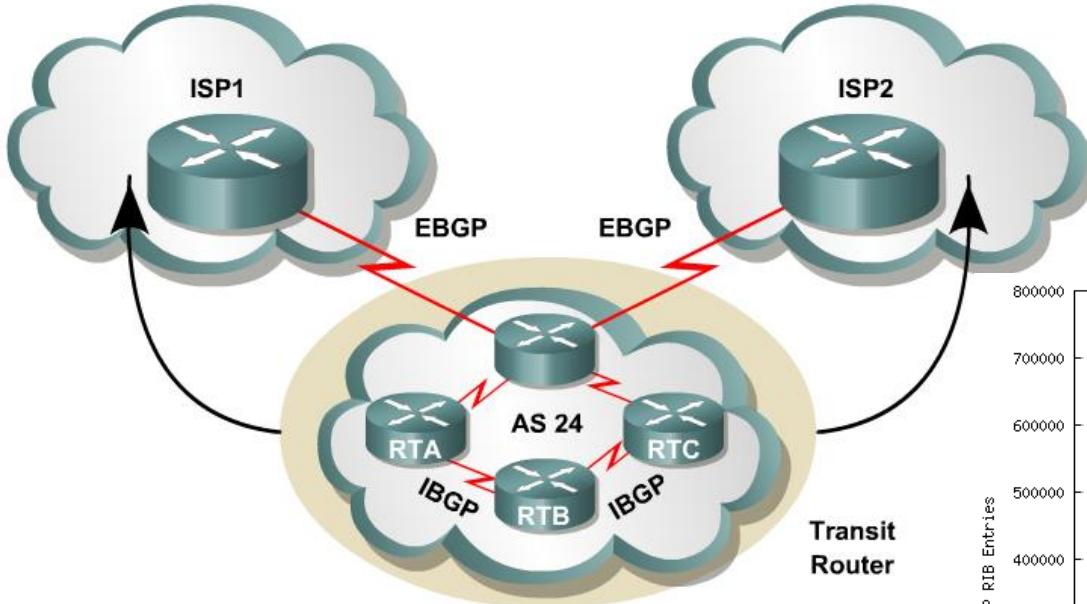
- Creating a BGP “peering” relationship involves an interesting combination of trust and mistrust.
- You must trust the network administrator on that end to know what they are doing.
- At the same time, if you are smart, you will take every practical measure to protect yourself in the event that a mistake is made on the other end.
- “Paranoia is your friend.”

BGP Hazards – Doyle, Routing TCP/IP

- Your ISP will show little patience with you if you make mistakes in your BGP configuration.
- Suppose, for example, that through some misconfiguration you advertise 207.46.0.0/16 to your ISP.
- On the receiving side, the ISP does not filter out this incorrect route, allowing it to be advertised to the rest of the Internet.
- This particular CIDR block belongs to Microsoft, and you have just claimed to have a route to that destination.
- A significant portion of the Internet community could decide that the best path to Microsoft is through your domain.
- You will receive a flood of unwanted packets across your Internet connection and, more importantly, you will have black-holed traffic that should have gone to Microsoft.
- They will be neither amused nor understanding.

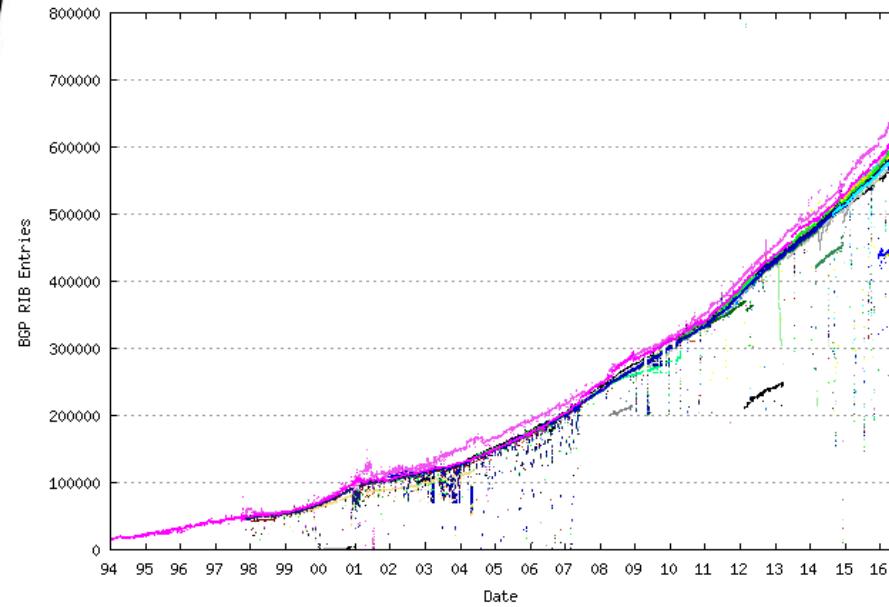
BGP Basics

BGP Basics

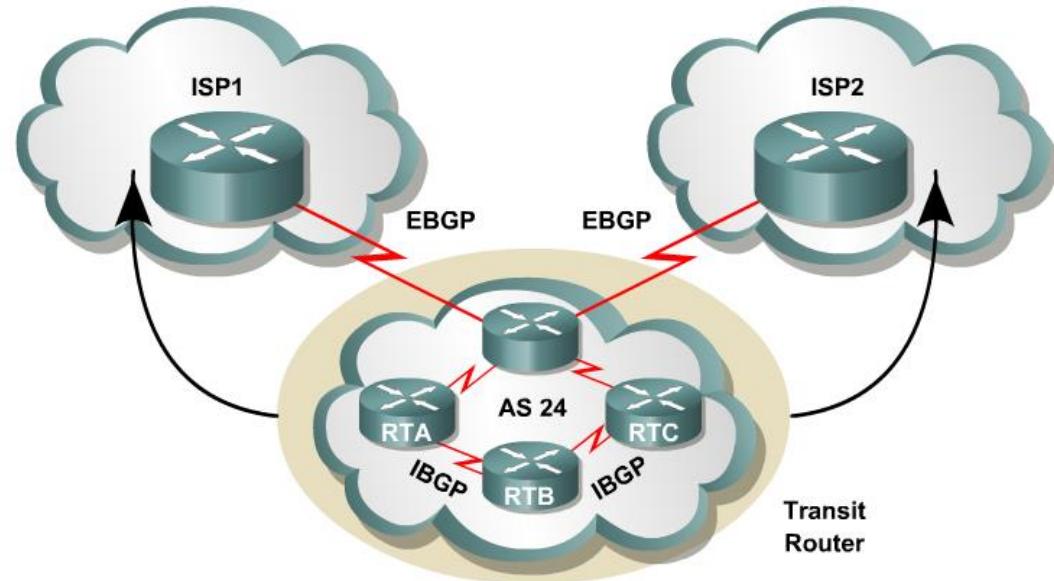


- BGP is a **path vector** routing protocol.
- BGP version 4 (BGP-4) is the latest version of BGP.
 - Defined in RFC 4271.
 - Supports supernetting (CIDR) and VLSM .
- BGP4 and CIDR prevent the Internet routing table from becoming too large.
 - Without CIDR, the Internet would have few millions entries.
 - With CIDR, Internet core routers manage around 600,000 entries.
 - <http://bgp.potaroo.net/>
- BGP is a distance vector routing protocol, in that it relies on downstream neighbors to pass along routes from their routing table.
- BGP uses a list of AS numbers through which a packet must pass to reach a destination.

- Internal routing protocols announce a list of networks and the metrics to get to each network.
- In contrast, BGP routers exchange network reachability information, called **path vectors**, made up of **path attributes**.

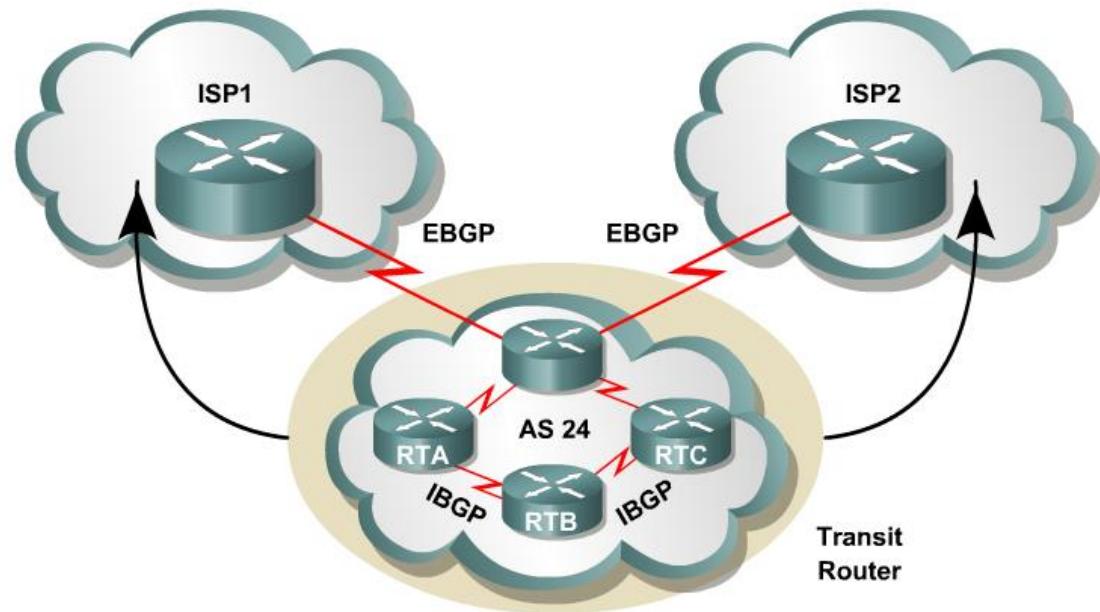


BGP Basics



- The function of BGP is to:
 - Exchange routing information between autonomous systems
 - Guarantee the selection of a loop free path.
- BGP4 is the first version of BGP that supports CIDR and route aggregation.
- Common IGPs such as RIP, OSPF, and EIGRP use technical metrics.
 - BGP does not use technical metrics.
- BGP makes routing decisions based on network policies, or rules (later)
- BGP does not show the details of topologies within each AS.
- BGP sees only a tree of autonomous systems.
- Cisco routers maintain a separate routing table to hold BGP routes:
show ip bgp

BGP Basics



- BGP updates are carried using TCP on port 179.
 - In contrast, RIP updates use UDP port 520
 - OSPF, IGRP, EIGRP does not use a Layer 4 protocol
- Because BGP requires TCP, IP connectivity must exist between BGP peers.
- TCP connections must also be negotiated between them before updates can be exchanged.
- Therefore, BGP inherits those reliable, connection-oriented properties from TCP.

of Current BGP Routes

As of May 25, 2016, there were 619.795 routes in the routing tables of the Internet core routers.

<http://bgpupdates.potaroo.net/instability/bgpupd.html>

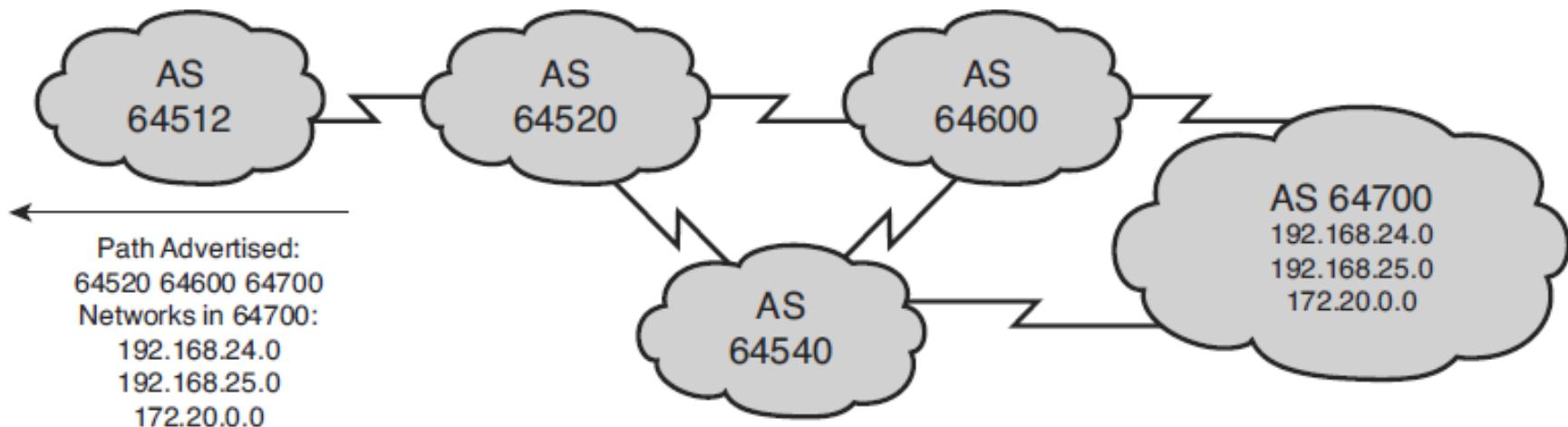
7 Day BGP Profile: 18-May-2016 00:00 - 24-May-2016 23:59 (UTC+1000)

Number of BGP Update Messages:	1876653	
Number of Prefix Updates:	4143042	
Number of Prefix Withdrawals:	204703	
Average Prefixes per BGP Update:	2.32	
Average BGP Update Messages per second:	2.72	
Average Prefix Updates per second:	6.29	
Peak BGP Update Message Rate per second:	6141	(06:48:01 Tue, 24-May-2016)
Peak Prefix Update Rate per second:	4412	(17:05:17 Tue, 17-May-2016)
Peak Prefix Withdraw Rate per second:	30330	(06:48:02 Tue, 24-May-2016)
Prefix Count:	619796	
Updated Prefix Count:	619795	
Stable Prefix Count:	1	
Origin AS Count:	54098	
Updated Origin AS Count:	54068	
Stable Origin AS Count:	30	
Unique Path Count:	289501	
Updated Path Count:	257356	
Stable Path Count:	32145	

BGP Basics

BGP Path Vector Characteristics

- The path vector information includes:
 - A list of the full path of BGP AS numbers (hop by hop) necessary to reach a destination network.
 - Other attributes including the IP address to get to the next AS (the next-hop attribute) and how the networks at the end of the path were introduced into BGP (the origin code attribute).



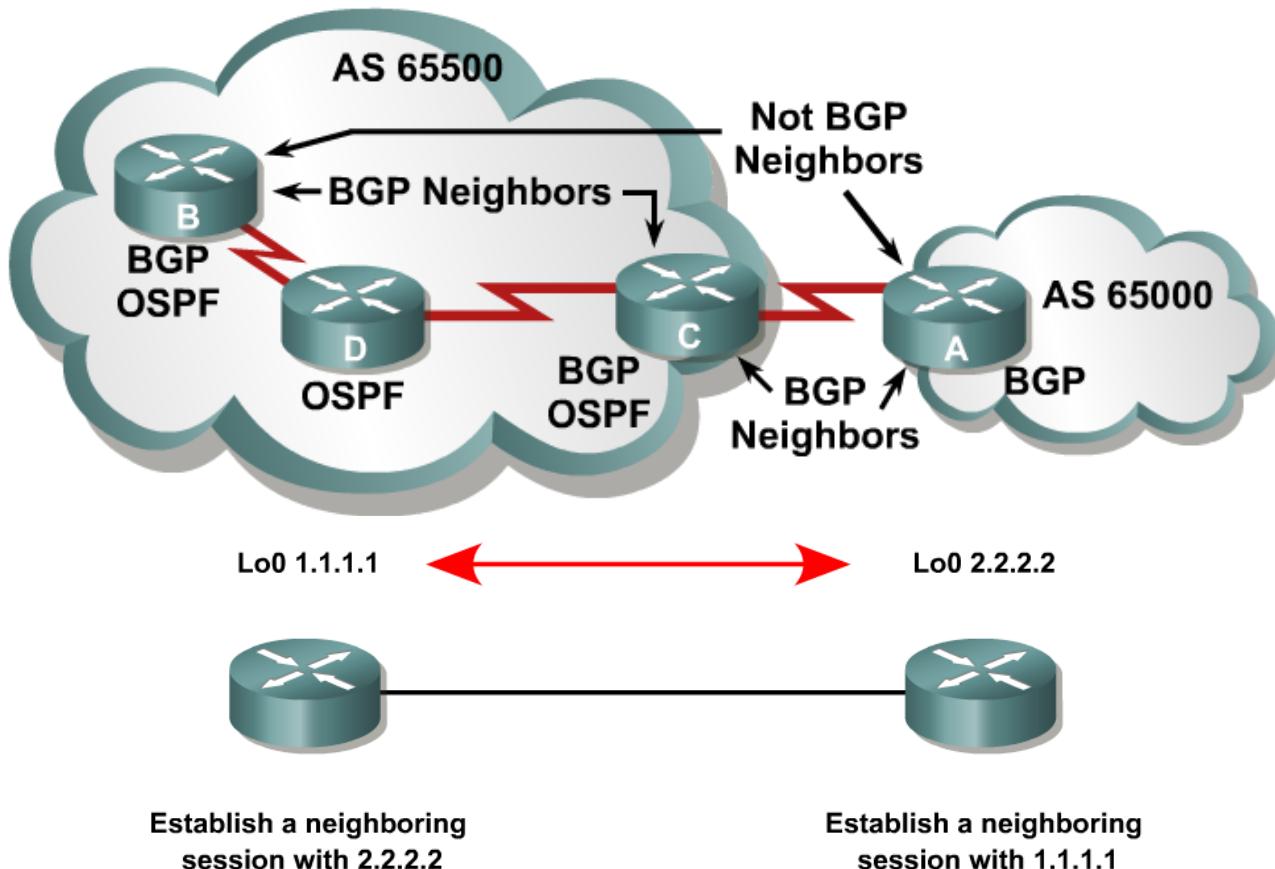
BGP Basics

Peers = Neighbors

- A “BGP peer,” also known as a “BGP neighbor,” is a specific term that is used for BGP speakers that have established a neighbor relationship.
- Any two routers that have formed a TCP connection to exchange BGP routing information are called BGP peers or BGP neighbors.

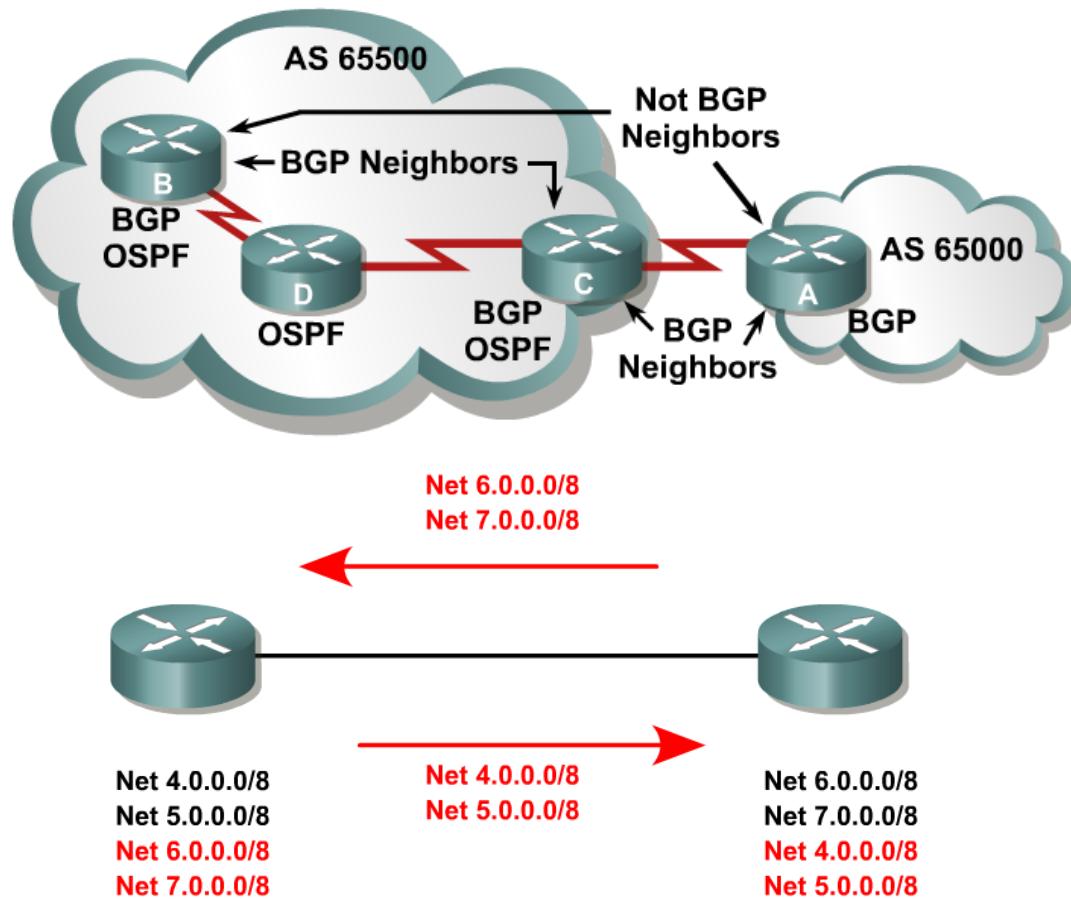
BGP Operational Overview

- When two routers establish a TCP enabled BGP connection, they are called **neighbors** or **peers**.
 - Peer routers exchange multiple connection messages.
- Each router running BGP is called a **BGP speaker**.

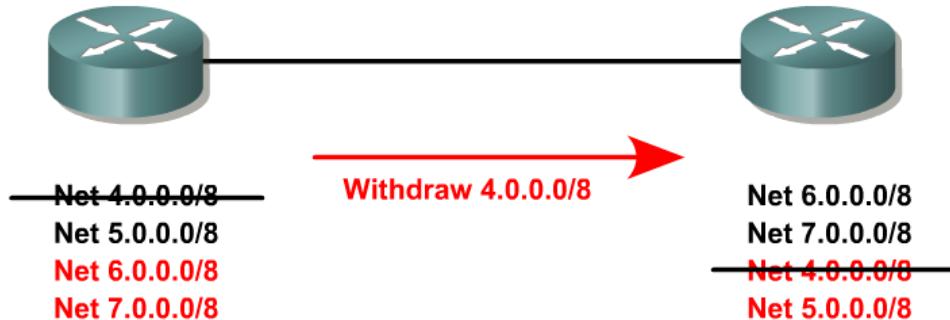


BGP Operational Overview

- When BGP neighbors first establish a connection, they exchange all candidate BGP routes.
- After this initial exchange, incremental updates are sent as network information changes.

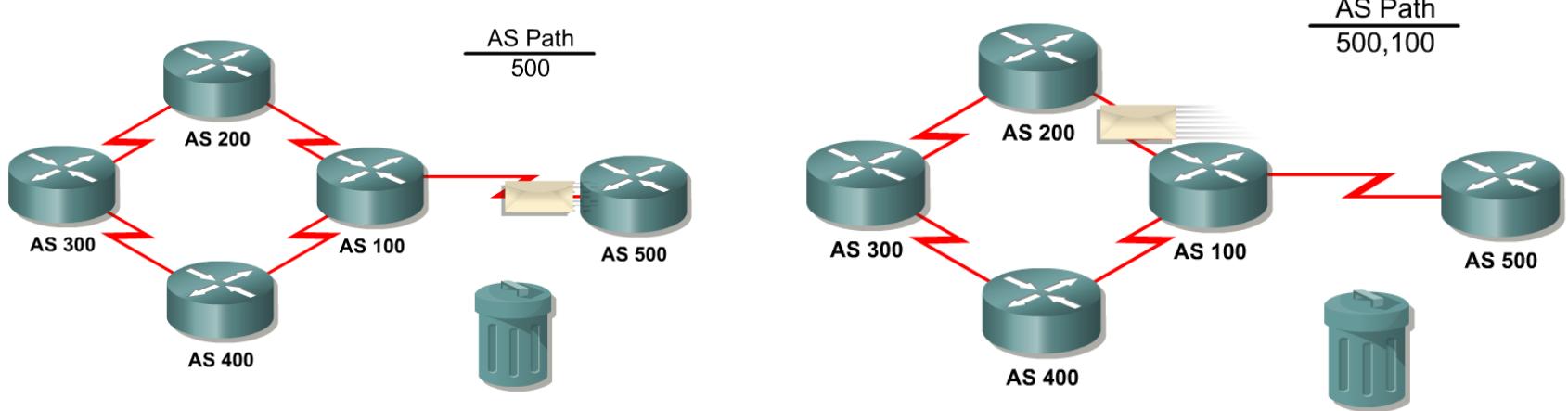


Withdrawn Routes



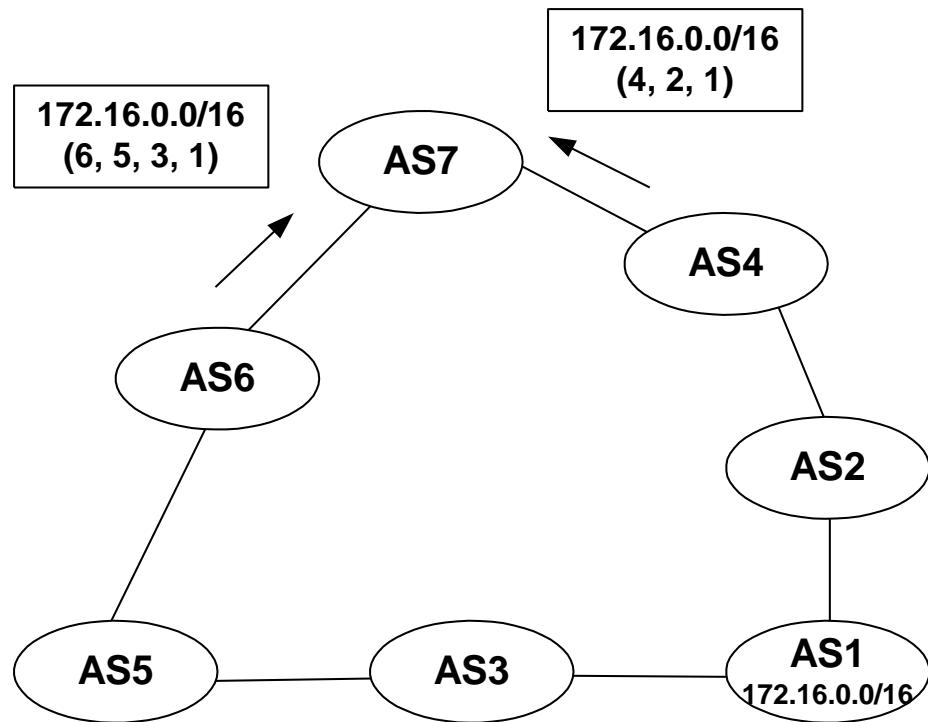
- The information for network reachability can change, such as when a route becomes unreachable or a better path becomes available.
- BGP informs its neighbors of this by withdrawing the invalid routes and injecting the new routing information.
- Withdrawn routes are part of the update message. BGP routers keep a table version number that tracks the version of the BGP routing table received from each peer.
- If the table changes, BGP increments the table version number.
- A rapidly incrementing table version is usually an indication of instabilities in the network, or a misconfiguration.

Loop Free Path



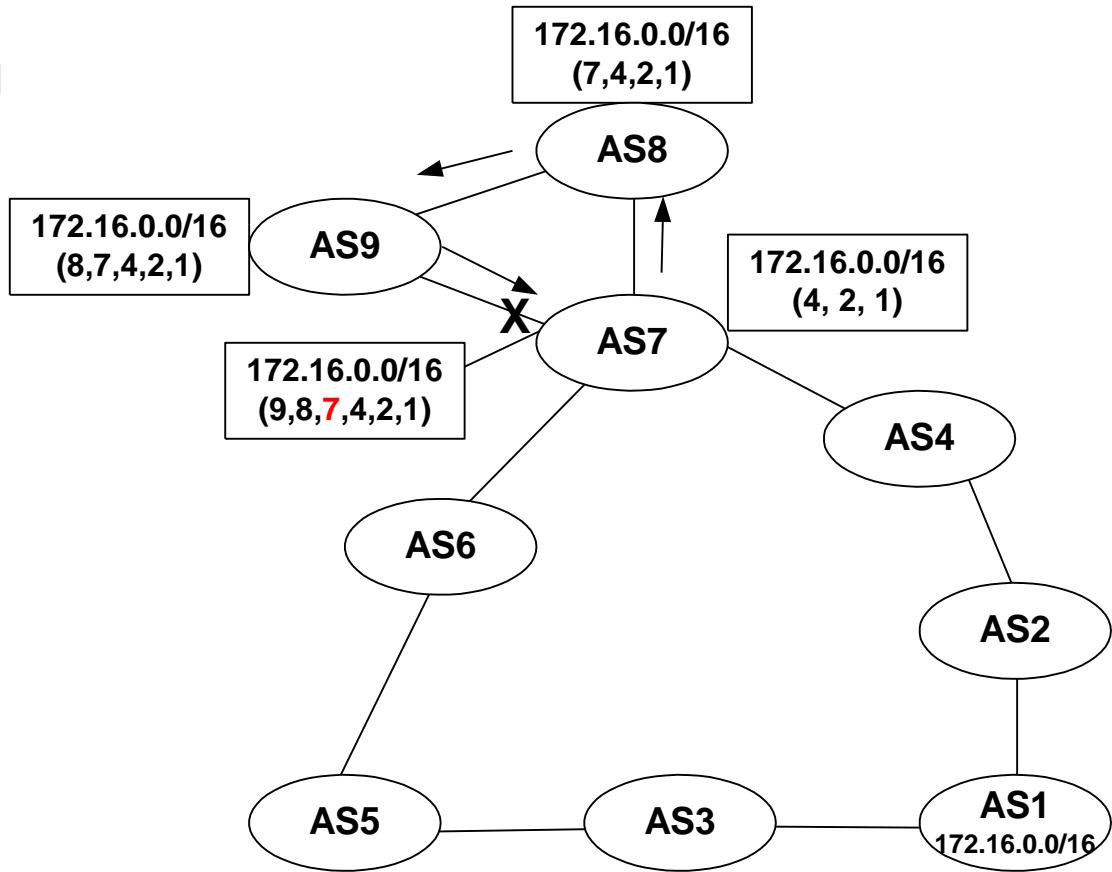
- To guarantee loop free path selection, BGP constructs a graph of autonomous systems based on the information exchanged between BGP neighbors.
- BGP views the whole internetwork as a graph, or tree, of autonomous systems.
- The connection between any two systems forms a path.
- The collection of path information is expressed as a sequence of AS numbers called the AS Path.
- This sequence forms a route to reach a specific destination

Loop Free Path



- The list of AS numbers associated with a BGP route is called the **AS_PATH** and is one of several path attributes associated with each route.
- Path attributes will be discussed in much more detail later.
- The shortest inter-AS path is very simply determined by the least number of AS numbers.
- All things being equal, BGP prefers routes with shorter AS paths.
- In this example, AS7 will choose the shortest path (4, 2, 1).
- We will see later what happens with equal cost paths.

Loop Free Path



Routing Loop Avoidance

- Route loops can be easily detected when a router receives an update containing its local AS number in the AS_PATH.
- When this occurs, the router will not accept the update, thereby avoiding a potential routing loop.

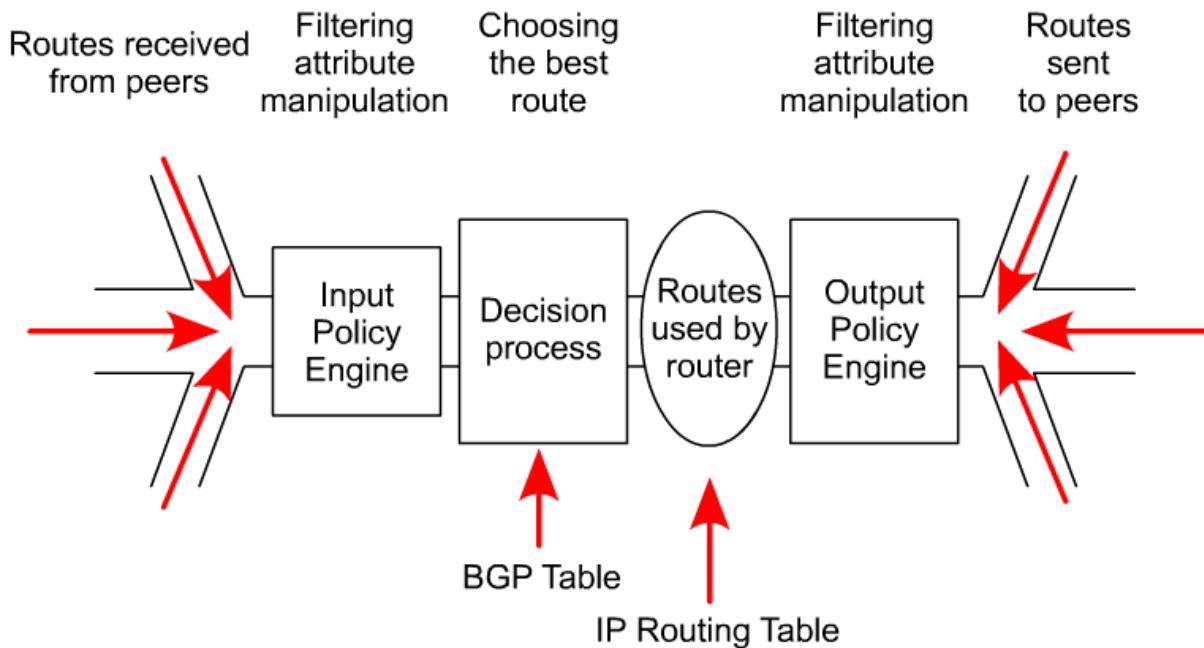
BGP Databases

- Neighbor table
 - List of BGP neighbors
- BGP table (forwarding database)
 - List of all networks learned from each neighbor
 - Can contain multiple paths to destination networks
 - Contains BGP attributes for each path
- IP routing table
 - List of best paths to destination networks

BGP Table

- BGP keeps its own table for storing BGP information received from and sent BGP neighbors.
 - This table is also known as the BGP table, BGP topology table, BGP topology database, BGP routing table, and the BGP forwarding database.
- The router offers the best routes from the BGP table to the IP routing table.

BGP Routing Process



- BGP is so flexible because it is a fairly simple protocol.
- Routes are exchanged between BGP peers via UPDATE messages.
- BGP routers receive the UPDATE messages, run some policies or filters over the updates, and then pass on the routes to other BGP peers.
- The Cisco implementation of BGP keeps track of all BGP updates in a BGP table separate from the IP routing table.

BGP Routing Process

BGP Routing Process

- Routes are exchanged between BGP peers by way of update messages
- BGP routers receive the update messages
- BGP routers run some policies or filters over the updates, and then pass the routes on to other BGP peers

- The Cisco implementation of BGP keeps track of all BGP updates in a BGP table separate from the IP routing table.
- In case multiple routes to the same destination exist, BGP does not flood its peers with all those routes. Instead, BGP picks only the best route and sends it to the peers.
- In addition to passing along routes from peers, a BGP router may originate routing updates to advertise networks that belong to its own AS.
- Valid local routes originated in the system and the best routes learned from BGP peers are then installed in the IP routing table.
- The IP routing table is used for the final routing decision.

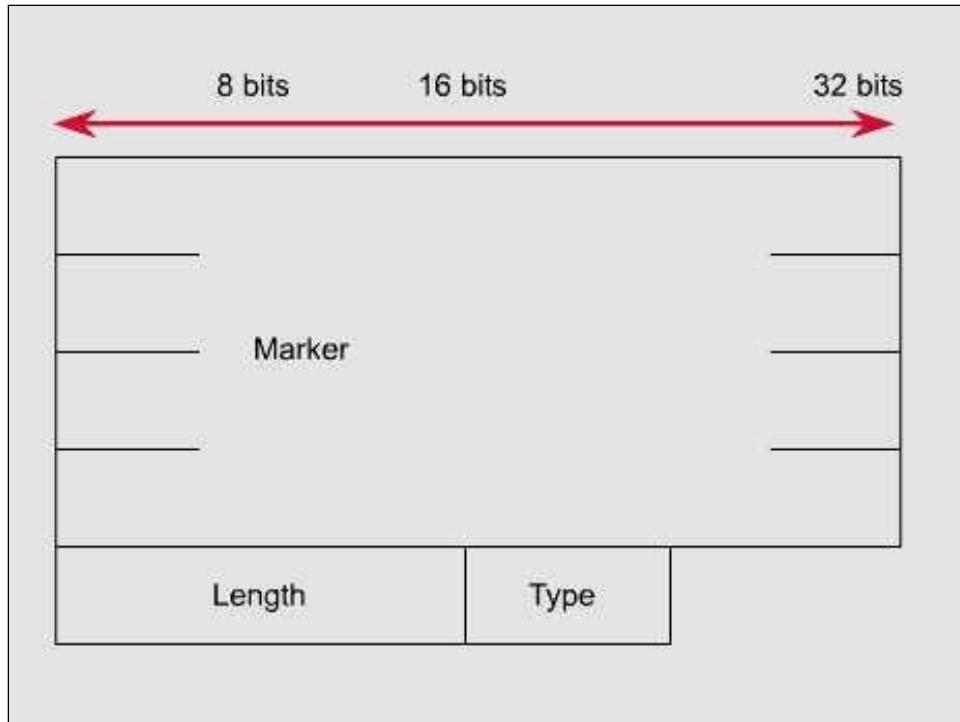
BGP Message Types

- Before establishing a BGP peer connection the two neighbors must perform the standard TCP three-way handshake and open a TCP connection to port 179.
- After the TCP session is established, BGP peers exchange several messages to open and confirm connection parameters and to send BGP routing information.
- All BGP messages are unicast to the one neighbor over the TCP connection.
- There are four BGP message types:
 - **Type 1: OPEN**
 - **Type 2: KEEPALIVE**
 - **Type 3: UPDATE**
 - **Type 4: NOTIFICATION**

BGP Message Types

Each BGP Message contains the following header:

- **Marker:** The marker field is used to either authenticate incoming BGP messages or to detect loss of synchronization between two BGP peers.
- **Length:** The length field indicates the total BGP message length, including the header (messages may be between 19 and 4096 bytes long).



BGP Message Header

Open Message

Octets	16	2	1	1	2	2	4	1	7
	Marker	Length	Type	Version	AS	Hold Time	BGP ID	Optional Length	Optional

Update Message

Octets	16	2	1	2	Variable	2	Variable	Variable
	Marker	Length	Type	Unfeasible Routes length	Withdrawn Routes	Attribute Length	Attributes	NLRI

Notification Message

Octets	16	2	1	1	1	Variable
	Marker	Length	Type	Error Code	Error Sub-code	Diagnostic Data

Keepalive Message

Octets	16	2	1
	Marker	Length	Type

Types of BGP Messages

Open Message

Octets	16	2	1	1	2	2	4	1	7
	Marker	Length	Type	Version	AS	Hold Time	BGP ID	Optional Length	Optional

Update Message

Octets	16	2	1	2	Variable	2	Variable	Variable
	Marker	Length	Type	Unfeasible Routes length	Withdrawn Routes	Attribute Length	Attributes	NLRI

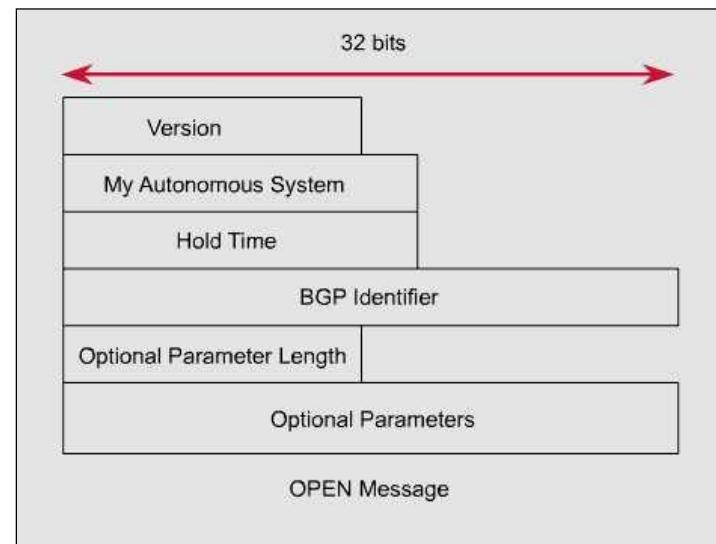
Notification Message

Octets	16	2	1	1	1	Variable
	Marker	Length	Type	Error Code	Error Sub-code	Diagnostic Data

Keepalive Message

Octets	16	2	1
	Marker	Length	Type

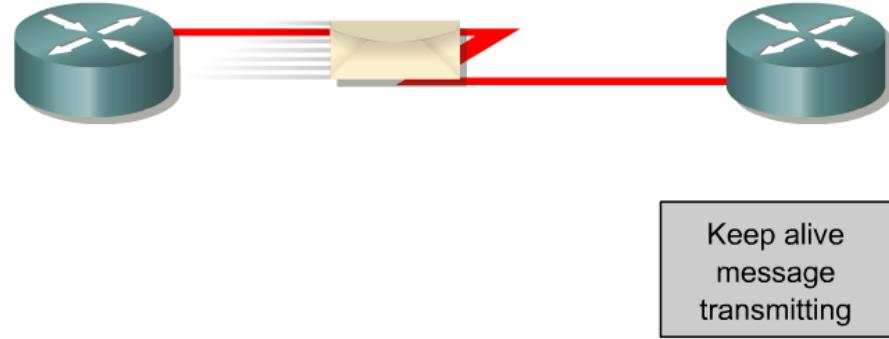
Type 1: BGP Open Message



- After the TCP session is established, both neighbors send Open messages which includes a set of parameters that have to be agreed upon before a full BGP adjacency can be established.
- This message is used to establish full connections with peers.
- Each neighbor uses this message to identify itself and to specify its BGP operational parameters including:
 - **BGP version number** (defaults to version 4)
 - **AS number**: AS number of the originating router, determines if BGP session is EBGP or IBGP.
 - **BGP identifier**: IP address that identifies the neighbor using the same method as OSPF router ID.
 - **Optional parameter**: authentication, multiprotocol support and route refresh.

16	2	1	1	2	2	4	1	7
Marker	Length	Type	Version	AS	Hold Time	BGP ID	Optional Length	Optional 67

Type 2: BGP Keepalive Message



- Keepalive messages are sent between peers every 60 seconds (by default) to maintain connections.
- The message consist of only a message header (19 bytes).
 - Hold time is three times the KEEPALIVE timer of 60 seconds.
 - If the periodic timer = 0, no keepalives are sent.
 - Recommended keepalive interval is one-third of the hold time interval.

Octets	16	2	1
Marker	Length	Type	

Type 3: BGP Update Message



- Update messages contain all the information BGP uses to construct a loop-free picture of the internetwork.
- A BGP update message has information on one path only; multiple paths require multiple update messages.
 - All the attributes in the update message refer to that path, and the networks are those that can be reached through it.

Octets	16	2	1	2	Variable	2	Variable	Variable
	Marker	Length	Type	Unfeasible Routes Length	Withdrawn Routes	Attribute Length	Path Attributes	NLRI

Type 3: BGP Update Message

- An update message includes the following information:
 - Unreachable routes information
 - Path attribute information
 - Network-layer reachability information (NLRI)
 - This field contains a list of IP address prefixes that are reachable by this path. 192.168.160.0/19
 - Prefix = 192.168.160.0
 - Prefix Length = 19

Octets	16	2	1	Unreachable Routes Information		Path Attributes Information		NLRI Information
	Marker	Length	Type	2	Variable	2	Variable	Variable
				Unfeasible Routes Length	Withdrawn Routes	Attribute Length	Path Attributes	NLRI

NLRI format

- The NLRI is a list of <**length**, **prefix**> tuples.
 - One tuple for each reachable destination.
 - The **prefix** represents the reachable destination
 - The prefix **length** represents the # of bits set in the subnet mask.

IP Address Subnet Mask	NLRI
10.1.1.0 255.255.255.0	24, 10.1.1.0
192.24.160.0 255.255.224.0	19, 192.24.160.0

Type 4: BGP Notification Message

- A BGP notification message is sent when an error condition is detected.
 - The BGP connection is closed immediately after this is sent.
- Notification messages include an error code, an error subcode, and data related to the error.

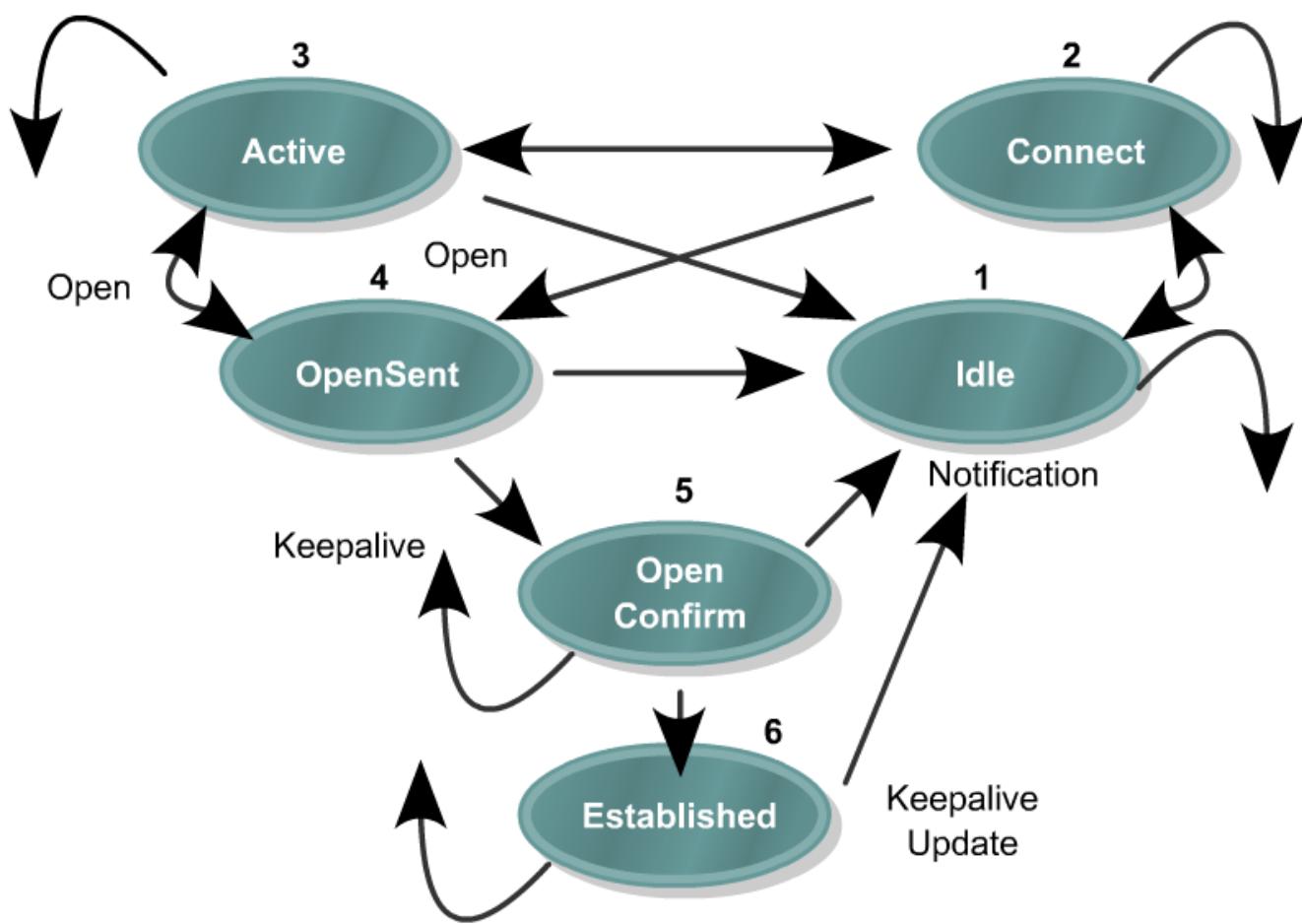
Octets	16	2	1	1	1	Variable
	Marker	Length	Type	Error Code	Error Sub-code	Diagnostic Data

Type 4: BGP Notification Message

- Sample error codes and their associated subcodes.

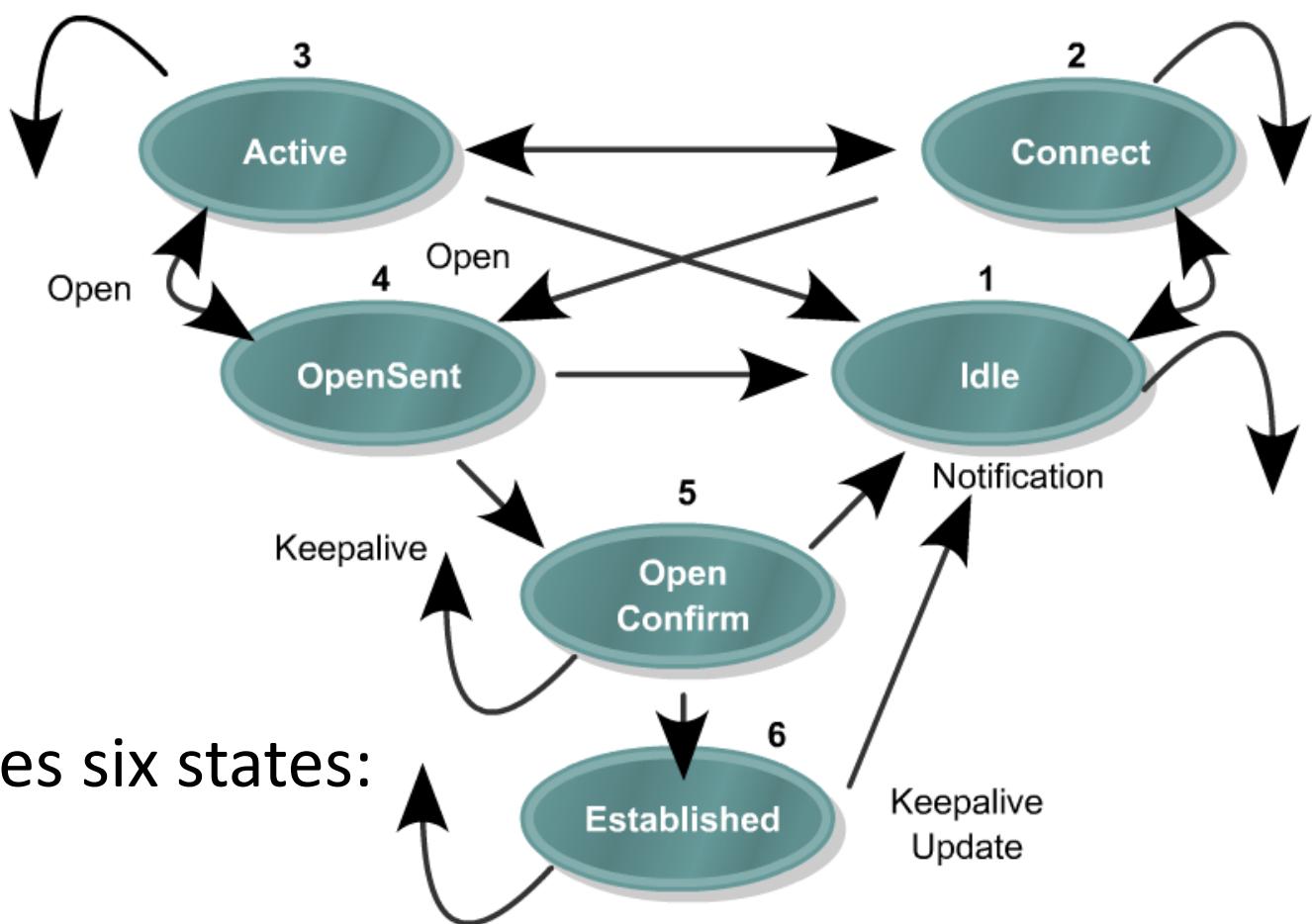
Error Code	Error Subcode
1--Message Header Error	1--Connection Not Synchronized 2--Bad Message Length 3--Bad Message Type
2--OPEN Message Error	1--Unsupported Version Number 2--Bad Peer AS 3--Bad BGP Identifier 4--Unsupported Optional Parameter 5--Authentication Failure 6--Unacceptable Hold Time
3--UPDATE Message Error	1--Malformed Attribute List 2--Unrecognized Well-Known Attribute 3--Missing Well-Known Attribute 4--Attribute Flags Error 5--Attribute Length Error 6--Invalid Origin Attribute 7--AS Routing Loop 8--Invalid NEXT_HOP Attribute 9--Optional Attribute Error 10--Invalid Network Field 11--Malformed AS_path
4--Hold Timer Expired	NOT applicable
5--Finite State Machine Error (for errors detected by the FSM)	NOT applicable
6--Cease (for fatal errors besides the ones already listed)	NOT applicable

BGP FSM



- The BGP neighbor negotiation process proceeds through various states, or stages, which can be described in terms of a finite-state machine (FSM).

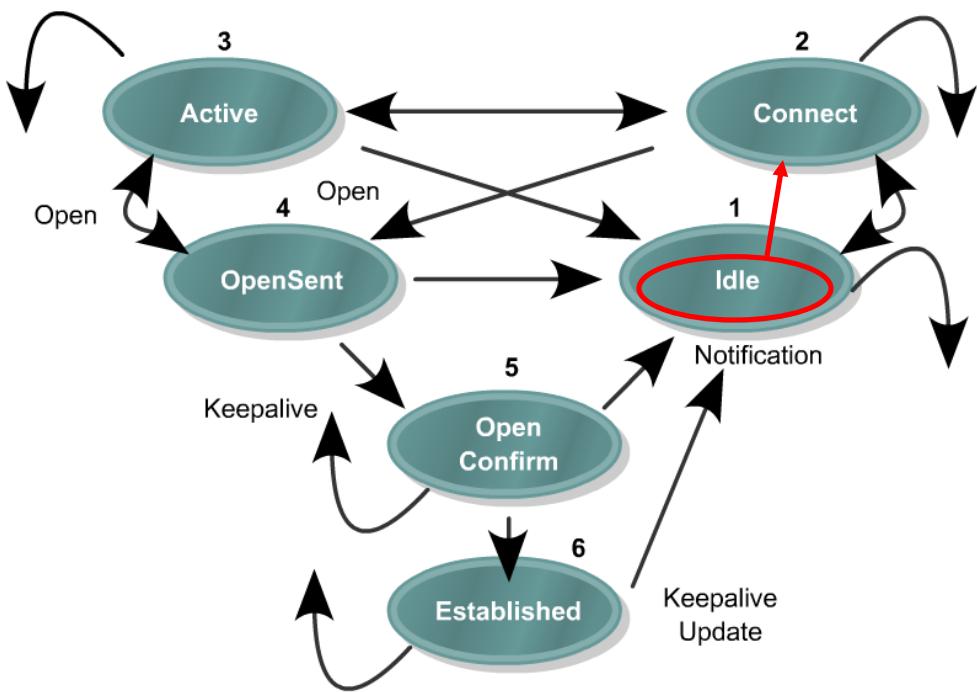
BGP FSM



BGP FSM includes six states:

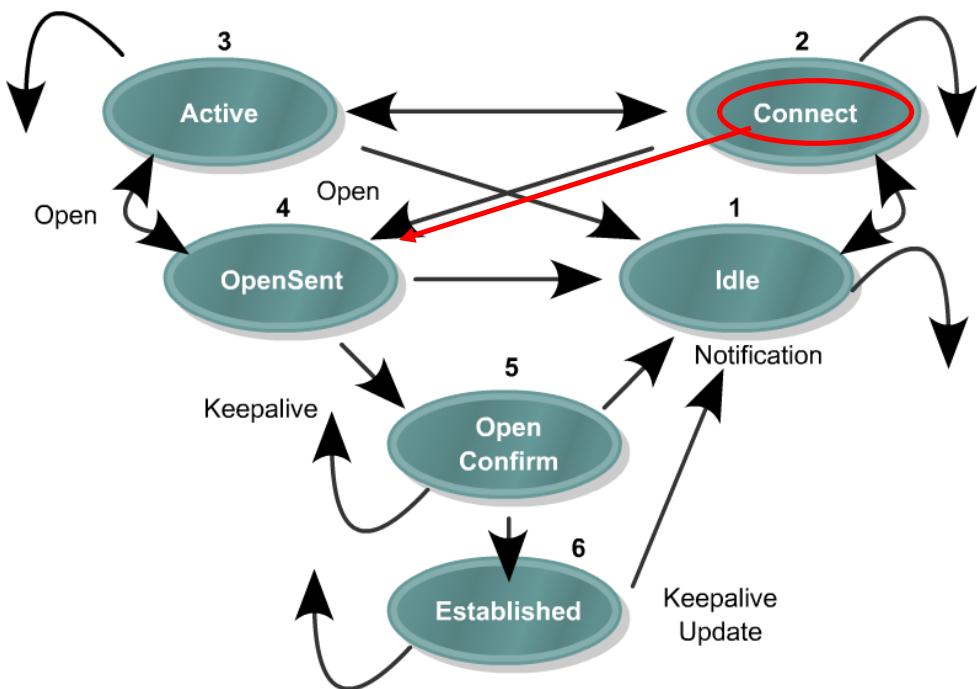
1. **Idle**
2. **Connect**
3. **Active**
4. **OpenSent**
5. **Open Confirm**
6. **Established**

Idle State



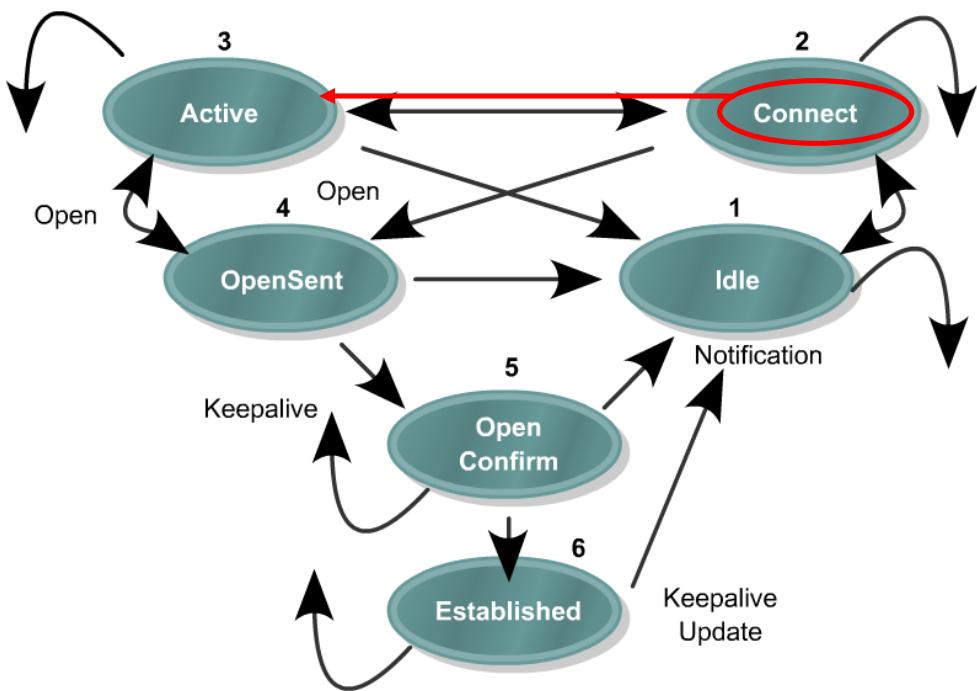
- BGP always begins in the **Idle** state, in which it refuses all incoming connections.
- It is normally initiated by an administrator or a network event.
- When Start event occurs, the BGP process:
 - Initializes all BGP resources
 - Starts the ConnectRetry timer
 - Initializes a TCP connection to the neighbor
 - Listens for a TCP initialization from the neighbor
 - Changes its state to **Connect**

Connect State



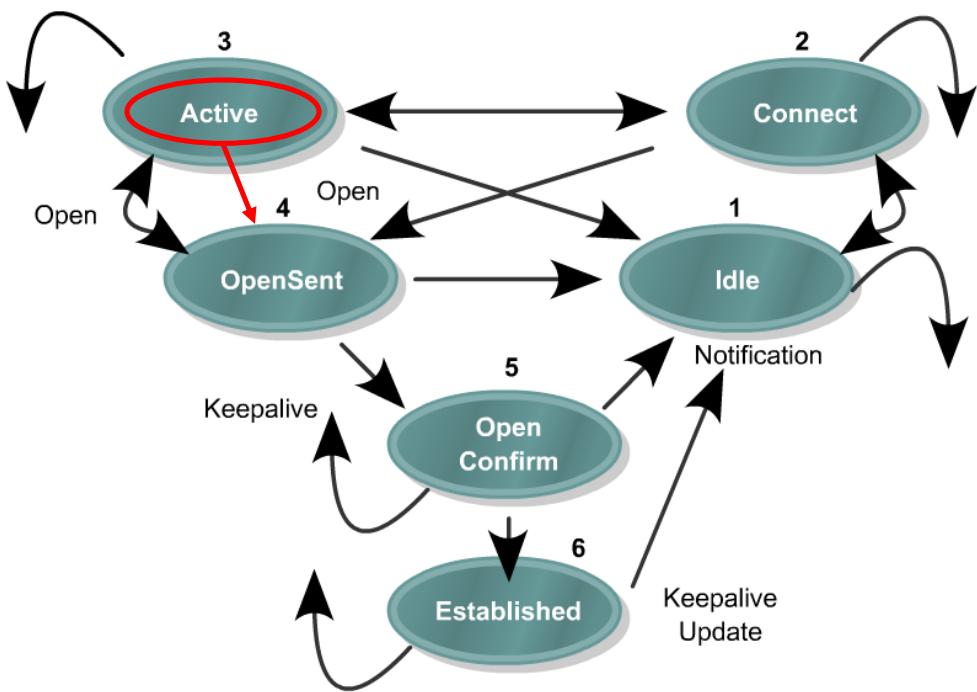
- In this state, the BGP process is waiting for the TCP connection to be completed.
- If the connection is **successful**, the BGP process:
 - Clears the ConnectRetry timer
 - Completes initialization
 - Sends an **Open message** to the neighbor
 - Transitions to the **OpenSent** state

Connect State



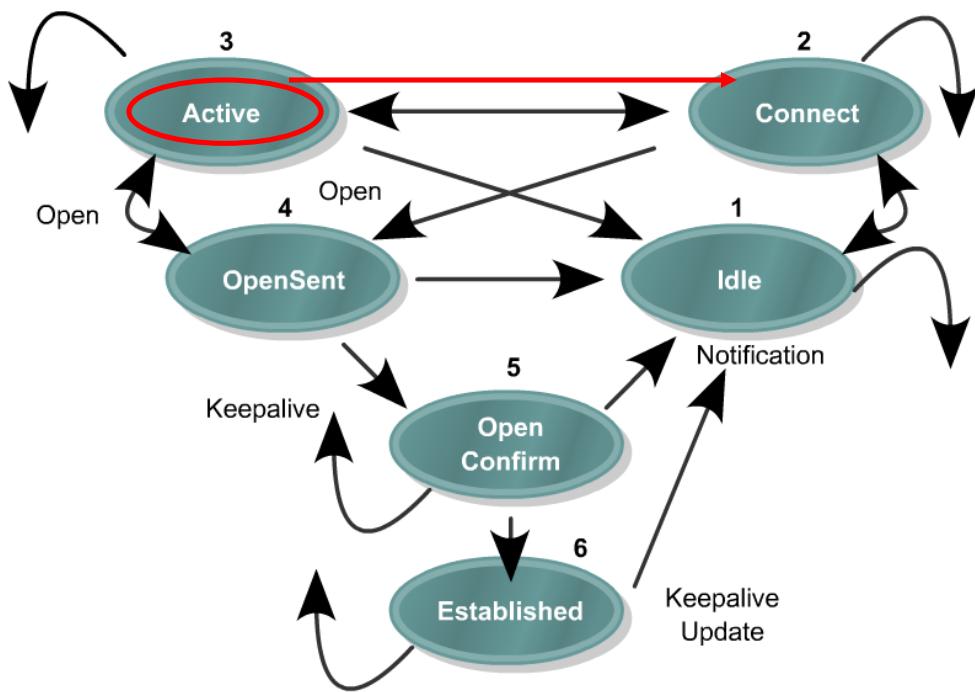
- If the connection is **unsuccessful**, the BGP process:
 - Continues to listen for a connection to be initiated by the neighbor
 - Resets the ConnectRetry timer
 - Transitions to the **Active** state

Active State



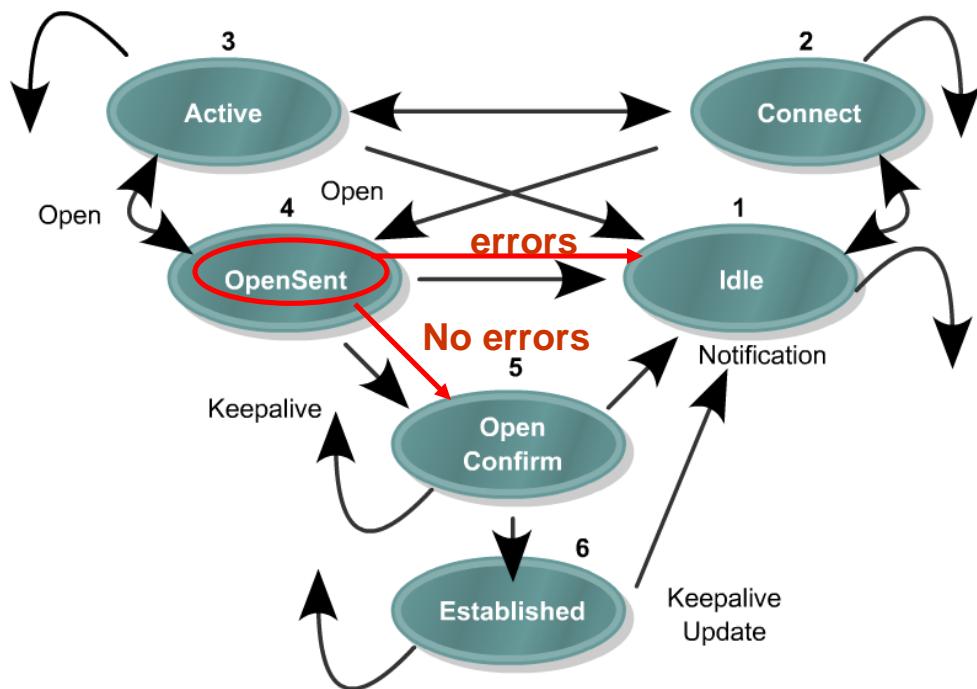
- In this state, the BGP process is trying to initiate a TCP connection with the neighbor.
- If the TCP connection is **successful**:
 - Clears the ConnectRetry timer
 - Completes initialization
 - Sends an **Open message** to the neighbor
 - Transitions to the **OpenSent** state

Active State



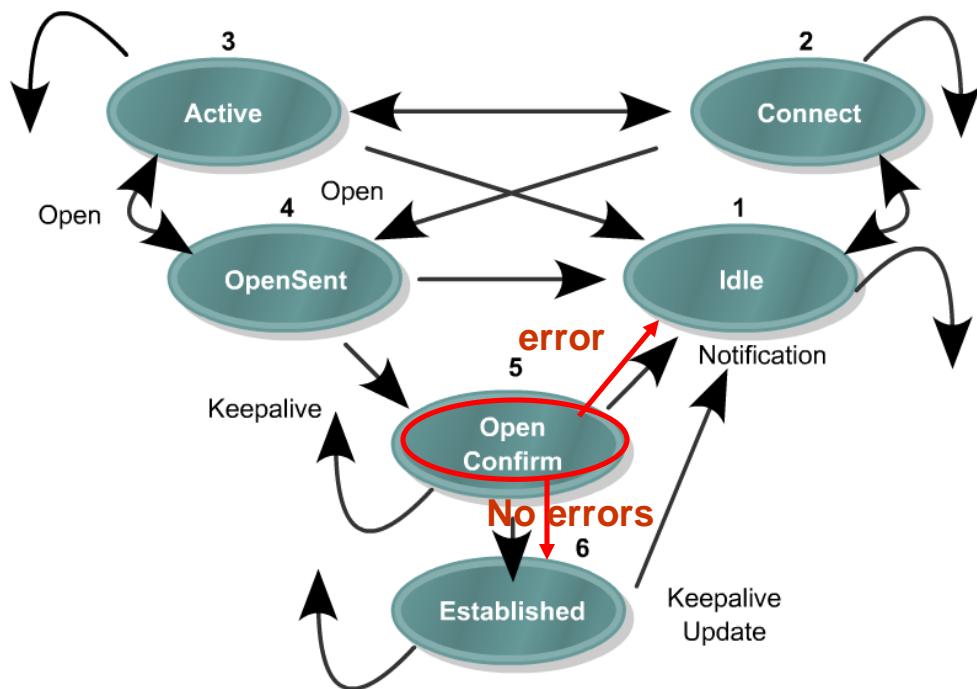
- If the ConnectRetry timer expires while BGP is in the Active State, the BGP process:
 - Transitions back to the **Connect** state
 - Resets the ConnectRetry timer
- In general, a neighbor state that is switching between "**Connect**" and "**Active**" is an indication that something is wrong and that there are problems with the TCP connection.
- It could be because of many TCP retransmissions, or the incapability of a neighbor to reach the IP address of its peer.

OpenSent State



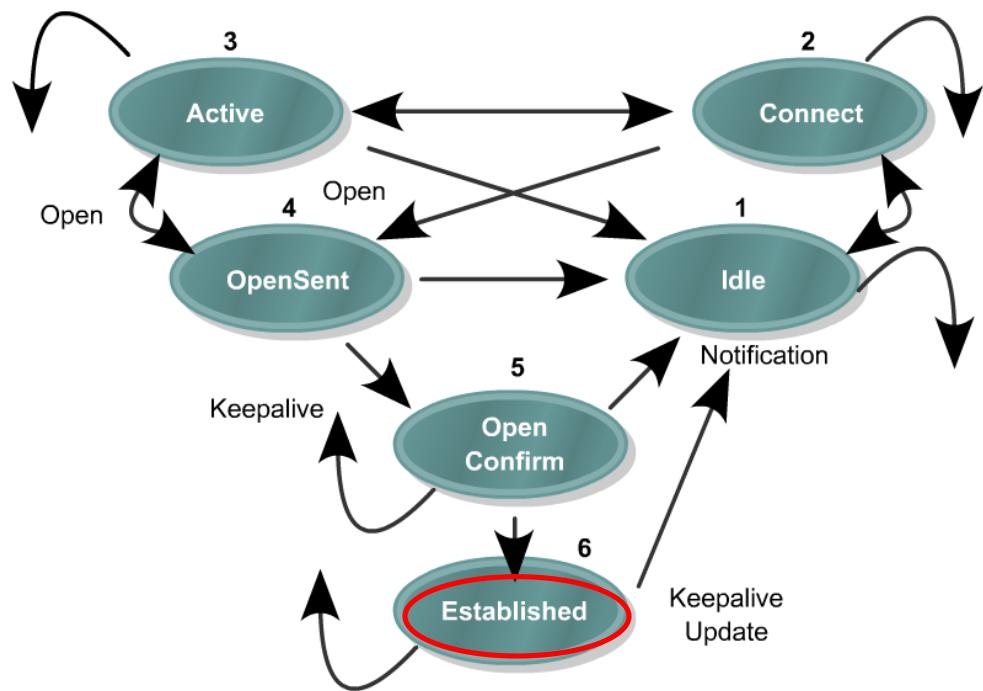
- In this state an **Open message** has been sent and BGP is waiting to hear an Open message from its neighbor.
- When an **Open message** is received, all its fields are checked.
 - If **errors** exist, a **Notification message** is sent and the state transitions to **Idle**.
 - If **no errors** exist, a **Keepalive message** is sent and the Keepalive timer is set, the peer is determined to be internal or external, and state is changed to **OpenConfirm**.

OpenConfirm State



- In this state, the BGP process waits for a **Keepalive** or **Notification message**.
- If a **Keepalive message** is received, the state transitions to **Established**.
- If a **Notification message** is received, or a TCP disconnect is received, the state transitions to **Idle**.

Established State



- In this state, the BGP connection is fully established and the peers can exchange **Update**, **Keepalive** and **Notification messages**.
- If an **Update** or **Keepalive message** is received, the Hold timer is restarted.
- If a **Notification message** is received, the state transitions to **Idle**.

Verifying BGP (Established State)

```
show ip bgp neighbors
```

Verify the BGP neighbor relationship.

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

Path Attributes

- Altering the routes changes traffic behavior.
 - How do I prevent my private networks from being advertised?
 - How do I filter routing updates coming from a particular neighbor?
 - How do I make sure that I use this link or this provider rather than another one?
- Path attributes are a set of BGP metrics describing the path to a network (route).
 - BGP uses the path attributes to determine the best path to the networks.
 - Some attributes are mandatory and automatically included in update messages while others are manually configurable.
- BGP attributes can be used to enforce a routing policy.
- When a BGP speaker receives updates from multiple autonomous systems that describe different paths to the same destination, it must choose the single best path for reaching that destination:
 - Once chosen, BGP propagates the best path to its neighbors.
 - The decision is based on the value of attributes (such as NEXT_HOP or LOCAL_PREF) that the update contains and other configurable BGP factors.

Path Attributes

- A BGP update message includes a variable-length sequence of path attributes describing the route.
- A path attribute consists of three fields:
 - Attribute type
 - Attribute length
 - Attribute value

BGP Attribute Type

- Type code 1 ORIGIN
- Type code 2 AS_PATH
- Type code 3 NEXT_HOP
- Type code 4 MULTI_EXIT_DISC
- Type code 5 LOCAL_PREF
- Type code 6 ATOMIC_AGGREGATE
- Type code 7 AGGREGATOR
- Type code 8 Community (Cisco-defined)
- Type code 9 Originator-ID (Cisco-defined)
- Type code 10 Cluster list (Cisco-defined)

Update Message					Path Attributes Information			
Octets	16	2	1	2	Variable	2	Variable	Variable
	Marker	Length	Type	Unfeasible Routes Length	Withdrawn Routes	Attribute Length	Path Attributes	NLRI

Path Attributes Within Update Message

A screenshot of the Wireshark network traffic analyzer. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Help, and various toolbar icons. A search bar labeled 'Filter:' is followed by 'Expression...', 'Clear', and 'Apply' buttons. The main window shows a single packet in the list view:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.19.51.37	172.19.51.71	BGP	UPDATE Message, UPDATE Message, UPDATE

The details view below shows the packet structure:

- Frame 1 (192 bytes on wire, 192 bytes captured)
- Ethernet II, Src: JuniperN_e6:08:c0 (00:14:f6:e6:08:c0), Dst: N (00:0c:29:1d:01:01)
- Internet Protocol, src: 172.19.51.37 (172.19.51.37), Dst: 172.19.51.71 (172.19.51.71)
- Transmission Control Protocol, src Port: bgp (179), Dst Port: 2
- Border Gateway Protocol
 - UPDATE Message
 - Marker: 16 bytes
 - Length: 51 bytes
 - Type: UPDATE Message (2)
 - Unfeasible routes length: 0 bytes
 - Total path attribute length: 25 bytes
 - Path attributes
 - ORIGIN: INCOMPLETE (4 bytes)
 - AS_PATH: 64601 (7 bytes)
 - NEXT_HOP: 172.19.50.1 (7 bytes)
 - COMMUNITIES: 42278:1123 (7 bytes)
 - Network layer reachability information: 3 bytes
 - 172.19.0.0/16
 - Border Gateway Protocol
 - Border Gateway Protocol

A yellow callout box with a black arrow points from the text "172.19.0.0/16." to the "Network layer reachability information" section in the Wireshark details pane. The callout box contains the following text:

Wireshark capture of an update message indicating the path attributes to reach network 172.19.0.0/16.

Attributes

- Some attributes are mandatory and automatically included in update messages while others are manually configurable.

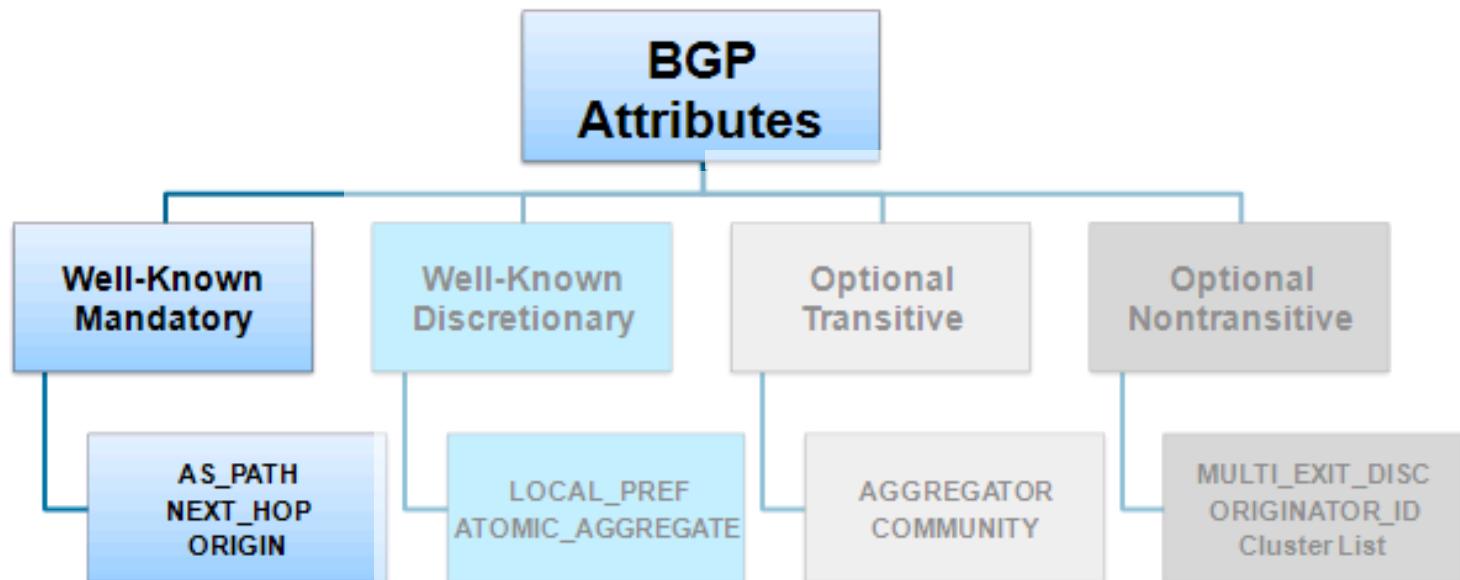
Attribute	EBGP	IBGP
AS_PATH	Well-known Mandatory	Well-known Mandatory
NEXT_HOP	Well-known Mandatory	Well-known Mandatory
ORIGIN	Well-known Mandatory	Well-known Mandatory
LOCAL_PREF	Not allowed	Well-known Discretionary
ATOMIC_AGGREGATE	Well-known Discretionary	Well-known Discretionary
AGGREGATOR	Optional Transitive	Optional Transitive
COMMUNITY	Optional Transitive	Optional Transitive
MULTI_EXIT_DISC	Optional Nontransitive	Optional Nontransitive

Automatically included in update message

Can be configured to help provide path control.

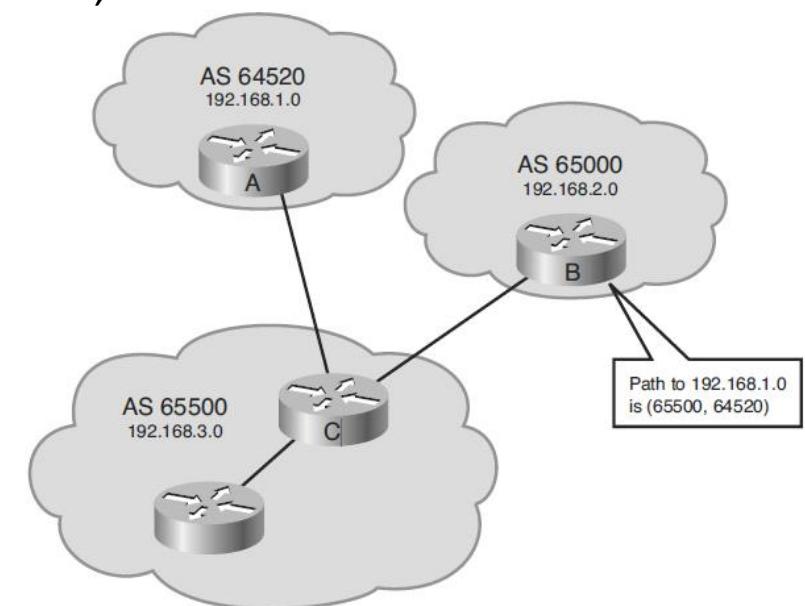
Well-Known Mandatory

- Attribute is recognized by all implementations of BGP and must appear in a BGP update message.
 - If missing, a notification error will be generated.
- Well-known mandatory attributes ensure that all BGP implementations agree on a standard set of attributes.

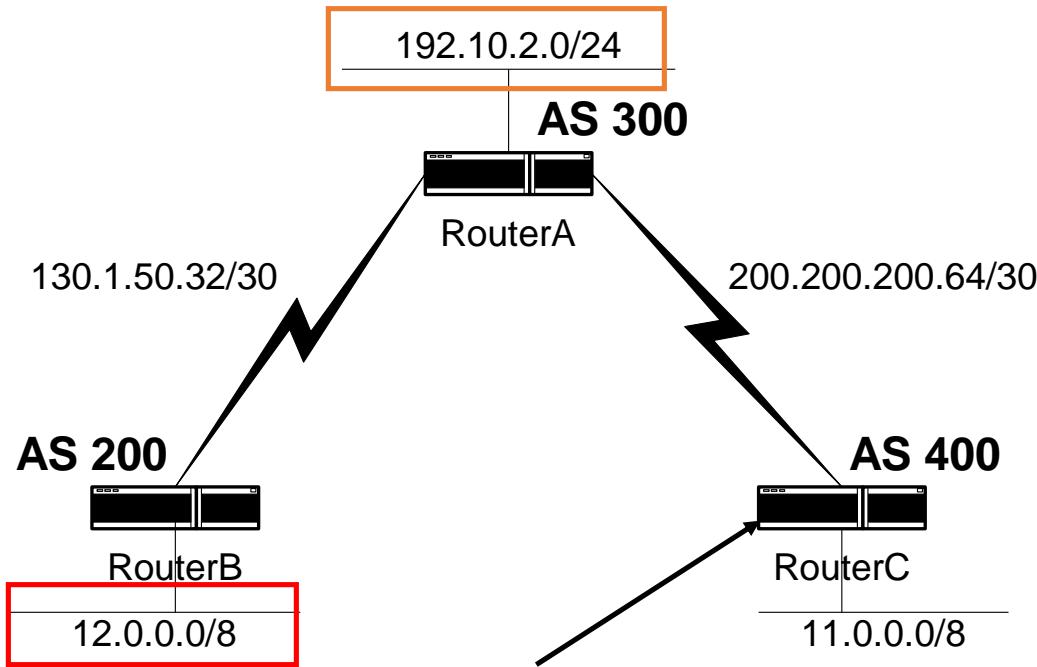


Well-Known Mandatory: AS_PATH

- An **AS_PATH** attribute is a **well-known mandatory attribute** (type code 2).
- It is the **sequence of AS numbers a route has traversed to reach a destination**.
- The AS that originates the route adds its own AS number when sending the route to its external BGP peers.
- Thereafter, each AS that receives the route and passes it on to other BGP peers will prepend its own AS number to the list.
- **Prepending** is the act of adding the AS number to the beginning of the list.
- The **final list** represents all the AS numbers that a route has traversed with the AS number of the AS that originated the route all the way at the end of the list.
- This type of **AS_PATH** list is called an **AS_SEQUENCE**, because all the AS numbers are ordered sequentially.



AS_PATH



```
RouterC# show ip bgp
```

```
BGP table version is 8, local router ID is 200.200.200.66
```

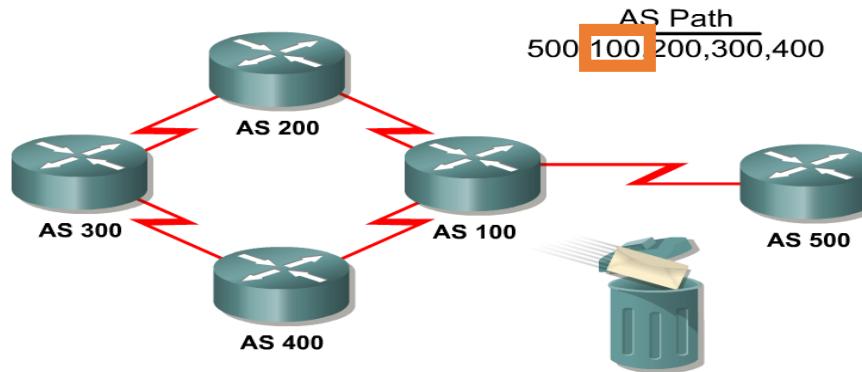
```
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 11.0.0.0	0.0.0.0	0		32768	i
*> 12.0.0.0	200.200.200.65		0	300 200	i
*> 192.10.2.0	200.200.200.65	0		0	300 i

AS_PATH

- BGP uses the **AS_PATH** attribute as part of the routing updates (**UPDATE packet**) to ensure a loop-free topology on the Internet.
- Each route that gets passed between BGP peers will carry a list of all AS numbers that the route has already been through.
- If the route is advertised to the AS that originated it, that AS will see itself as part of the **AS_PATH** attribute list and will not accept the route.

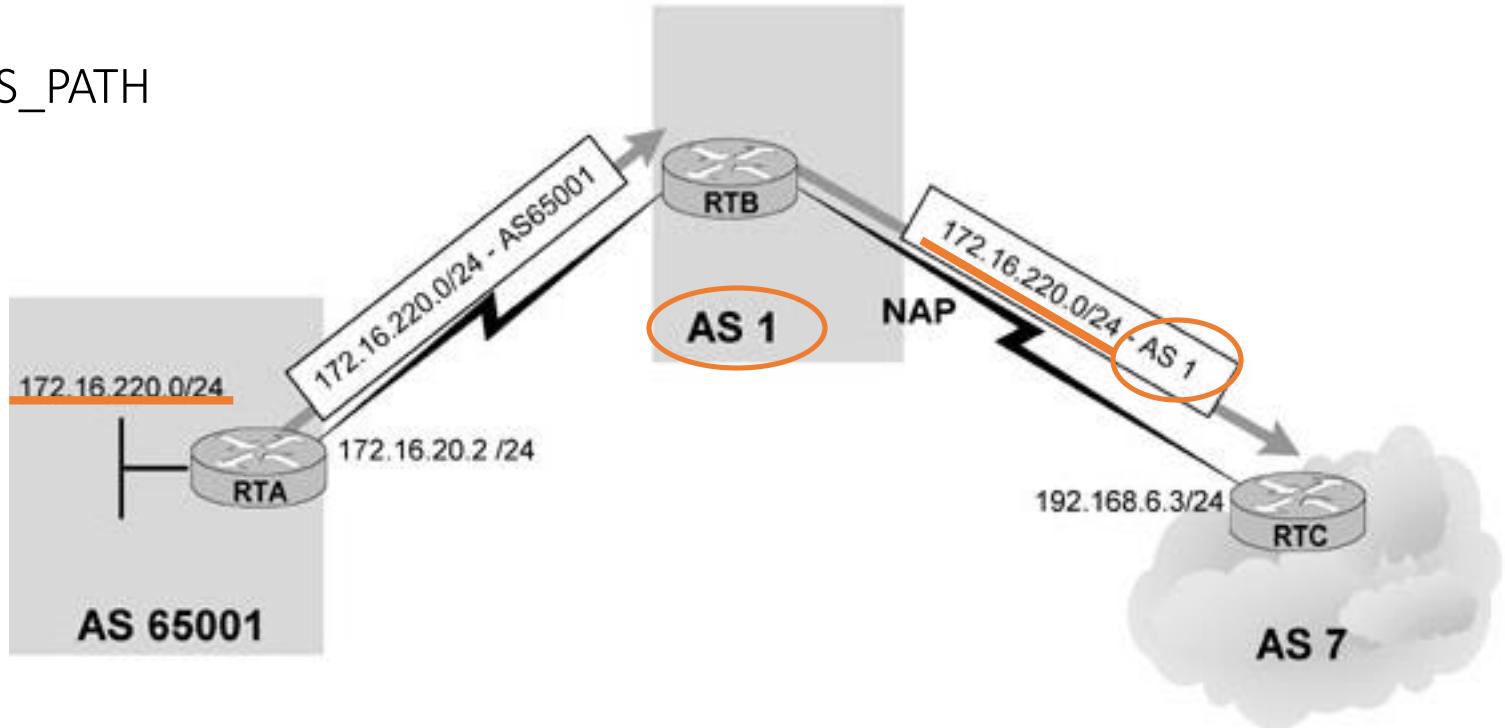


- **EBGP:** BGP speakers prepend their AS numbers when advertising routing updates to other autonomous systems (external peers).
- **IBGP:** When the route is passed to a BGP speaker within the same AS, the **AS_PATH** information is left intact.

AS_PATH – private AS numbers

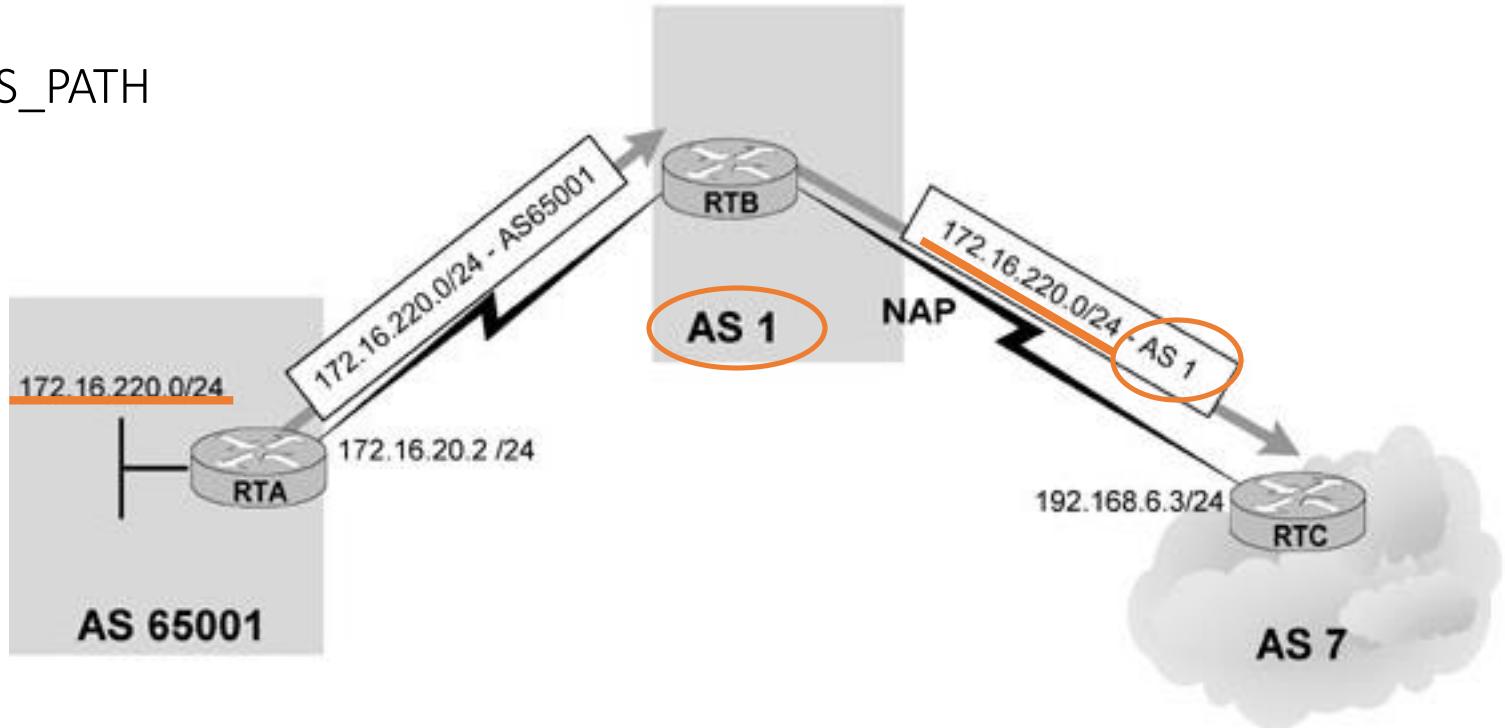
- **AS_PATH** information is one of the attributes BGP looks at to determine the **best route** to take to get to a destination.
 - In comparing two or more different routes, given that all other attributes are identical, a shorter path is always preferred.
 - In case of a tie in AS_PATH length, other attributes are used to make the decision. (later)
- **Private AS numbers** cannot be leaked to the Internet because they are not unique.
 - Cisco has implemented a feature, **remove-private-as**, to strip private AS numbers out of the **AS_PATH** list before the routes get propagated to the Internet.

AS_PATH



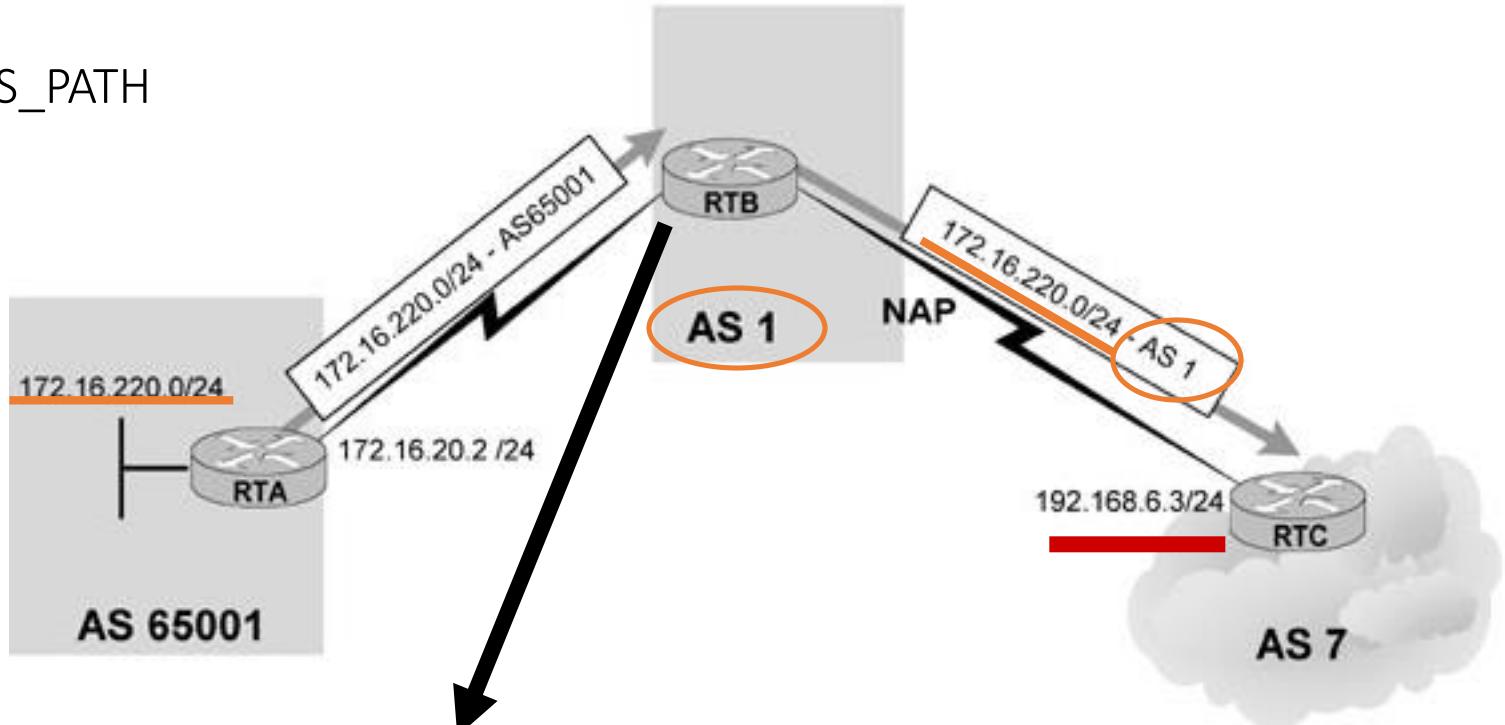
- AS1 is providing Internet connectivity to its customer AS 65001.
- Because the customer connects to only this provider and no plans to connect to an additional provider in the near future, the customer has been allocated a private AS number.
- **BGP will strip private AS numbers** only when propagating updates to the external peers.
- This means that the AS stripping would be configured on RTB as part of its neighbor connection to RTC.

AS_PATH



- Privately numbered autonomous systems should be connected only to a single provider.
- If the **AS_PATH** contains a mixture of private and legal AS numbers, BGP will view this as an illegal design and will not strip the private AS numbers from the list, and the update will be treated as usual.
- “**If the AS_PATH includes both private and public AS numbers, BGP doesn't remove the private AS numbers. This situation is considered a configuration error.**” Cisco
- Only **AS_PATH** lists that contain private AS numbers in the range 64512 to 65535 are stripped.

AS_PATH



```
RTB(config)#router bgp 1  
RTB(config-router)#neighbor 172.16.20.2 remote-as 65001  
RTB(config-router)#neighbor 192.168.6.3 remote-as 7  
RTB(config-router)#neighbor 192.168.6.3 remove-private-as
```

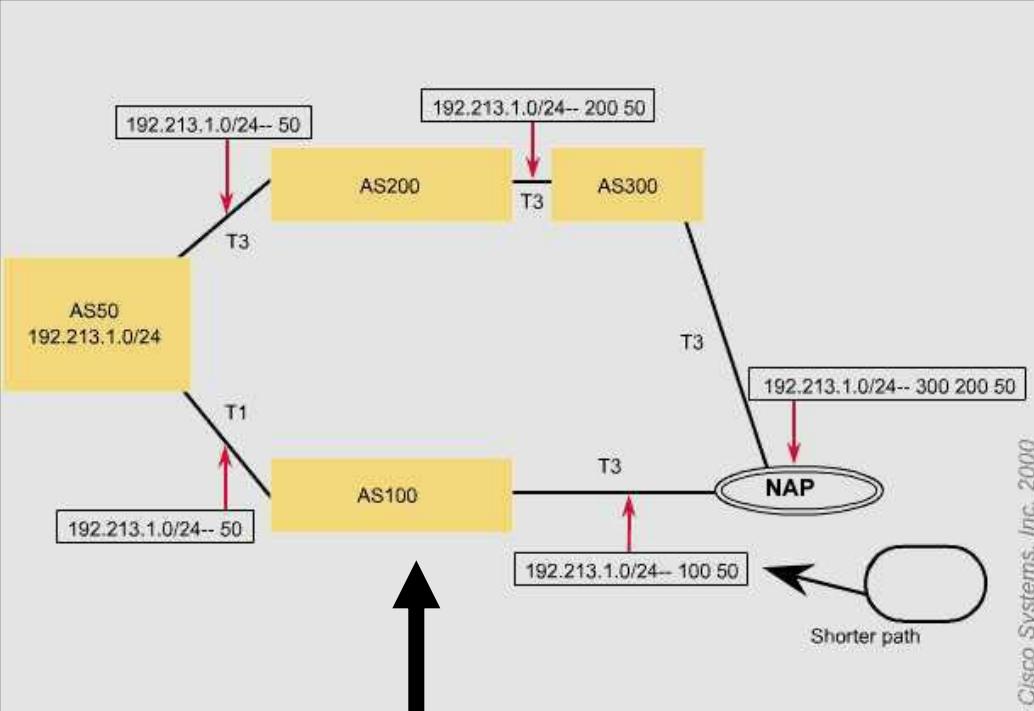
- Note how RTB is using the **remove-private-as** keyword in its neighbor connection to AS7.

<http://www.cisco.com/warp/public/459/32.html>

AS_PATH - prepend

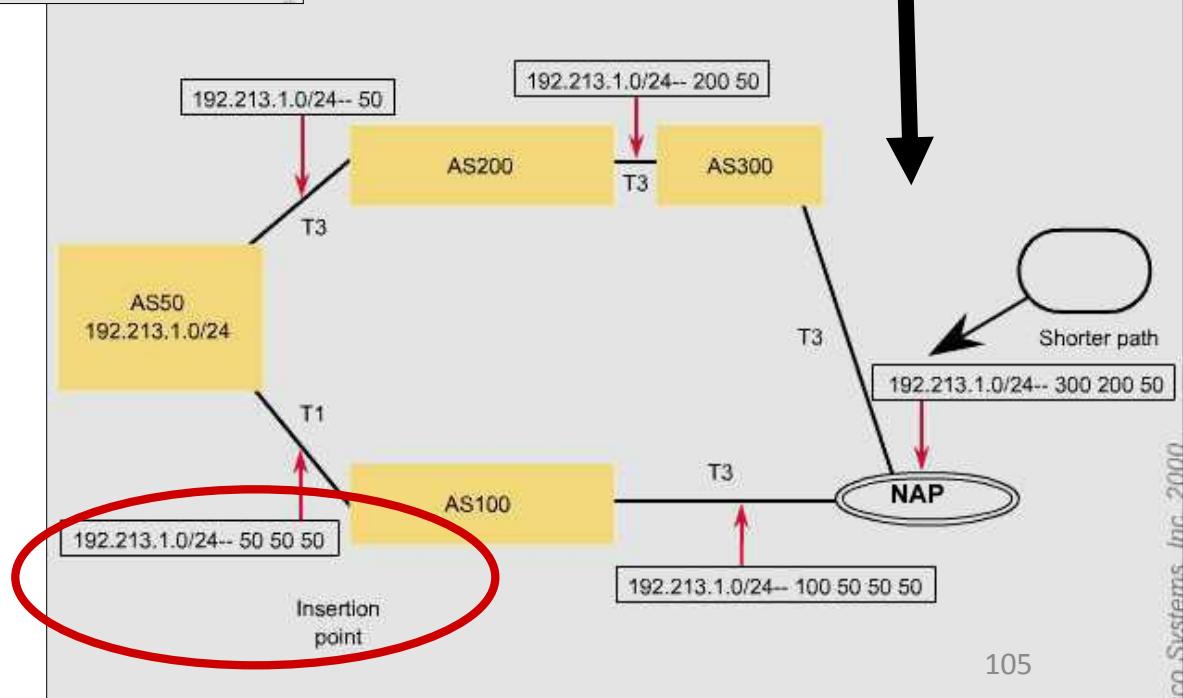
- **AS_PATH** information is manipulated to affect interdomain routing behavior.
- Because BGP prefers a shorter path over a longer one, system operators are tempted to change the path information by including dummy AS path numbers that would increase the path length and influence the traffic trajectory one way or the other.
- **Cisco's implementation** enables a user to **insert AS numbers** at the beginning of an AS_PATH to make the path length longer.

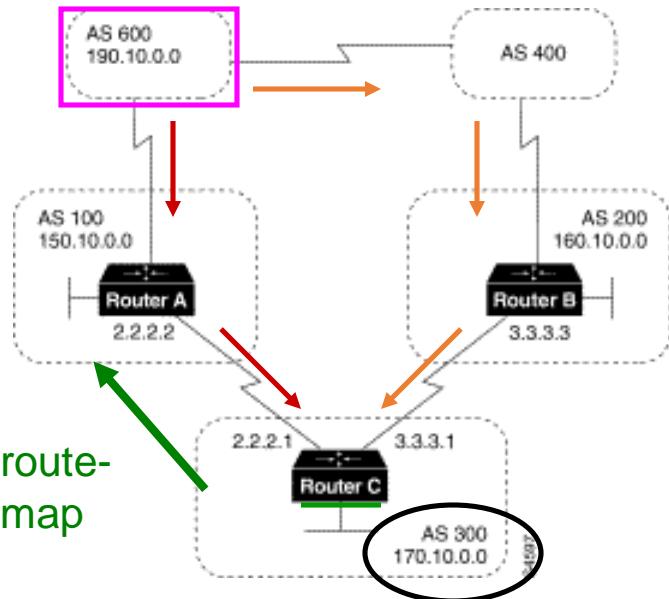
AS_PATH – prepend
Concept



Current “shorter path”

New “shorter path”





Router C

AS_PATH – prepend - Example

router bgp 300

network 170.10.0.0

neighbor 3.3.3.3 remote-as 200

neighbor 2.2.2.2 remote-as 100

neighbor 2.2.2.2 route-map SETPATH out

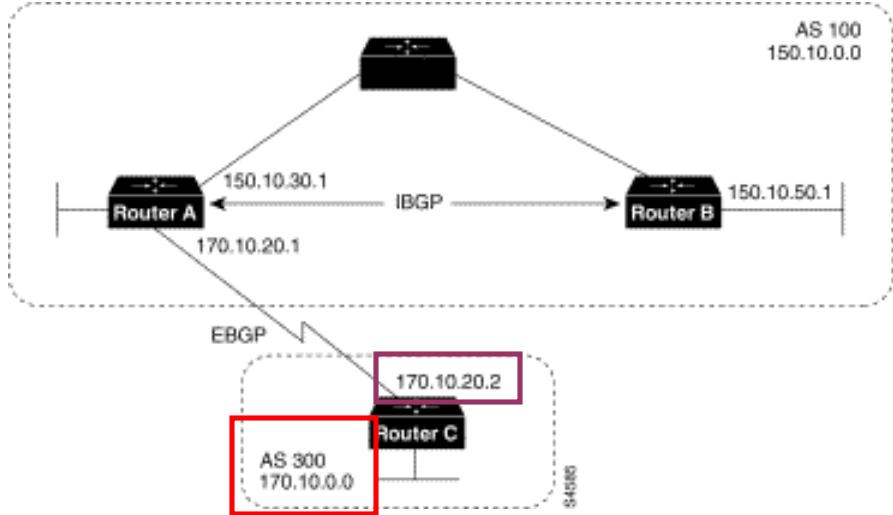
route-map SETPATH permit 10

set as-path prepend 300 300

- If you want to **use the configuration of Router C to influence the choice of paths** in **AS 600**, you can do so by prepending extra AS numbers to the **AS_path** attribute for routes that **Router C** advertises to AS 100.
- A common practice is to repeat the AS number, as in the above configuration.
- The set as-path route map configuration command with the **prepend** keyword causes **Router C to prepend 300 twice to the value of the AS_path attribute before it sends updates to the neighbor at IP address 2.2.2.2 (Router A)**.
- As a result, the **AS_path** attribute of updates for network **170.10.0.0** that **AS 600** receives via **AS 100** will be **100, 300, 300, 300**, which is longer than the value of the **AS_path** attribute of updates for network **170.10.0.0** that **AS 600** receives via **AS 400 (400, 200, 300)**.
- **AS 600 will choose (400, 200, 300) as the better path.**

Well-Known Mandatory: NEXT_HOP

- The NEXT_HOP attribute indicates the IP address that is to be used to reach a destination.
- The **NEXT_HOP** attribute is a well-known mandatory attribute (type code 3).
- In terms of an **IGP**, such as RIP, the “next hop” to reach a route is the IP address of the router that has announced the route.
 - Note: The abbreviation **IGP** (Interior Gateway Protocol) will always be in **green**, so not to get it confused with **IBGP** (Interior BGP)
- The **NEXT_HOP** concept with BGP is slightly more elaborate.
- For **EBGP** sessions, the next hop is **the IP address of the neighbor that announced the route**
- For **IBGP** sessions, **for routes originated inside the AS, the next-hop is the IP address of the neighbor that announced the route.**
- **For routes injected into the AS via EBGP, the next hop learned from EBGP is carried unaltered into IBGP.**
 - The next hop is the IP address of the EBGP neighbor from which the route was learned.



Router A

```
router bgp 100
neighbor 170.10.20.2 remote-as 300
neighbor 150.10.50.1 remote-as 100
network 150.10.0.0
```

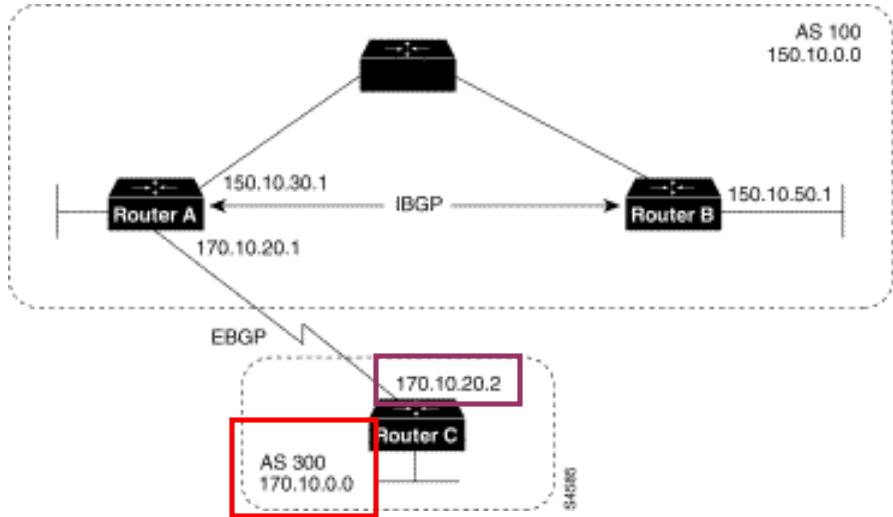
Router B

```
router bgp 100
neighbor 150.10.30.1 remote-as 100
```

Router C

```
router bgp 300
neighbor 170.10.20.1 remote-as 100
network 170.10.0.0
```

- Router C advertises network **170.10.0.0** to Router A with a next hop attribute of **170.10.20.2**, and Router A advertises network **150.10.0.0** to Router C with a **next hop attribute of 170.10.20.1**.
- BGP specifies that the **next hop of EBGP-learned routes should be carried without modification into IBGP**.
- Because of that rule, **Router A** advertises **170.10.0.0** to its IBGP peer (Router B) with a next hop attribute of **170.10.20.2**.
- As a result, according to **Router B**, the next hop to reach **170.10.0.0** is **170.10.20.2**, instead of 150.10.30.1.
- For that reason, the configuration must ensure that **Router B** can reach **170.10.20.2** via an **IGP**.
- Otherwise, Router B will drop packets destined for **170.10.0.0** because the next hop address is inaccessible.
- For example, if Router B runs IGRP, Router A should run IGRP on network **170.10.0.0**.
- You might want to make IGRP passive on the link to Router C so that only BGP updates are exchanged.



Summarize

- Router C advertises **170.10.0.0** to Router A with a **next hop** attribute of **170.10.20.2**,
- Router A advertises **170.10.0.0** to Router B with a next hop attribute of **170.10.20.2**.
- ***The next hop of EBGP-learned routes is passed to the IBGP neighbor without modification into IBGP.***

Router A

```
router bgp 100
neighbor 170.10.20.2 remote-as 300
neighbor 150.10.50.1 remote-as 100
network 150.10.0.0
```

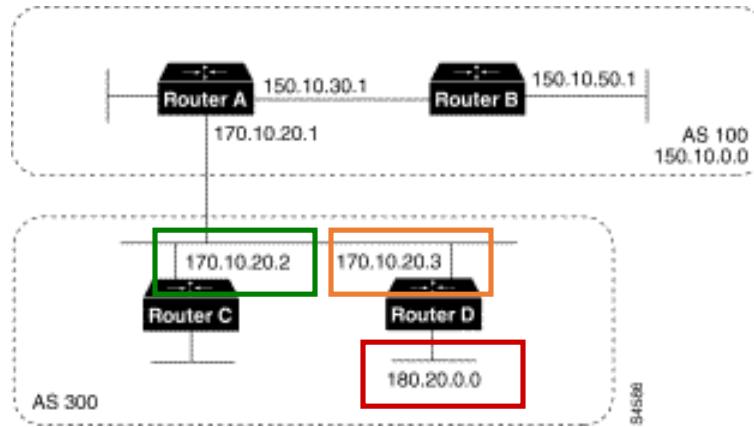
Router B

```
router bgp 100
neighbor 150.10.30.1 remote-as 100
```

Router C

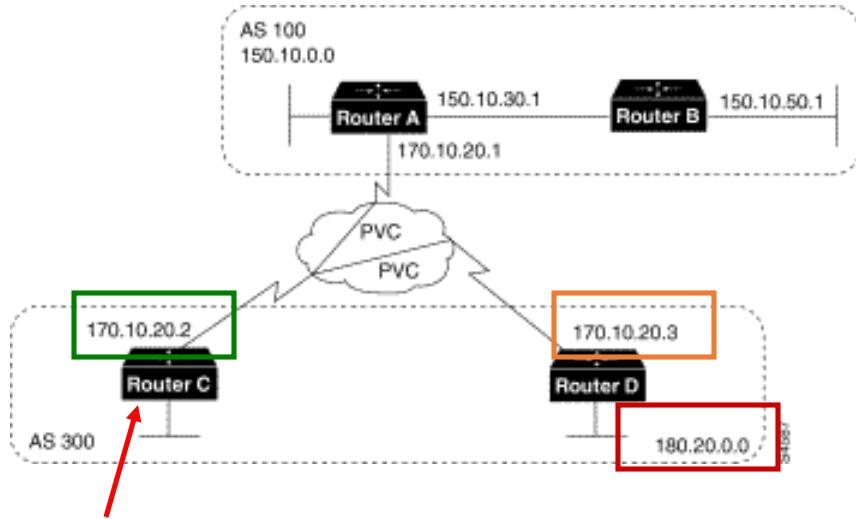
```
router bgp 300
neighbor 170.10.20.1 remote-as 100
network 170.10.0.0
```

NEXT_HOP and Multiaccess Media



- Routers C and D are in AS 300 are running OSPF.
- Router C is running BGP with Router A.
- Router C can reach network **180.20.0.0** via **170.10.20.3**.
- When Router C sends a BGP update to Router A regarding **180.20.0.0**, it sets the next hop attribute to **170.10.20.3**, instead of its own IP address (**170.10.20.2**).
- This is because Routers A, B, and C are in the same subnet, and it makes more sense for Router A to use Router D as the next hop rather than taking an extra hop via Router C.

NEXT_HOP and Multiaccess Media



Router C

```
router bgp 300
```

```
neighbor 170.10.20.1 remote-as 100
```

```
neighbor 170.10.20.1 next-hop-self
```

- Routers A, C, and D, use a common media such as Frame Relay (or any NBMA cloud).
- Router C advertises **180.20.0.0** to Router A with a next hop of **170.10.20.3**, just as it would do if the common media were Ethernet.
- The problem is that Router A does not have a direct permanent virtual connection (PVC) to Router D and cannot reach the next hop, so routing will fail.
- To remedy this situation, use the **neighbor next-hop-self** router configuration command.
- The neighbor **next-hop-self** command causes Router C to advertise **180.20.0.0** with the next hop attribute set to **170.10.20.2**.

Well-Known Mandatory: ORIGIN

- Well-known mandatory attribute (type code 1)
- Indicates the origin of the routing update
 - **IGP:**
 - The route is interior to the originating AS and normally occurs when a **network** command is used to advertise the route via BGP.
 - An origin of IGP is indicated with an “**i**” in the BGP table.
 - **EGP:**
 - (Obsolete) The route is learned via EGP which is considered a historic routing protocol and is not supported on the Internet.
 - An origin of EGP is indicated with an “**e**” in the BGP table.
 - **Incomplete:**
 - The route’s origin is unknown or is learned via some other means and usually occurs when a route is redistributed into BGP.
 - An incomplete origin is indicated with a “**?**” in the BGP table.
- BGP considers the ORIGIN attribute in its decision-making process to establish a preference ranking among multiple routes.
- Specifically, BGP prefers the path with the lowest origin type, where
 - IGP is lower than EGP
 - and EGP is lower than INCOMPLETE.

Well-Known Mandatory: ORIGIN

```
R1# show ip bgp
BGP table version is 24, local router ID is 172.16.1.2
Status codes: s suppressed, d damped, h history, * valid, > k
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

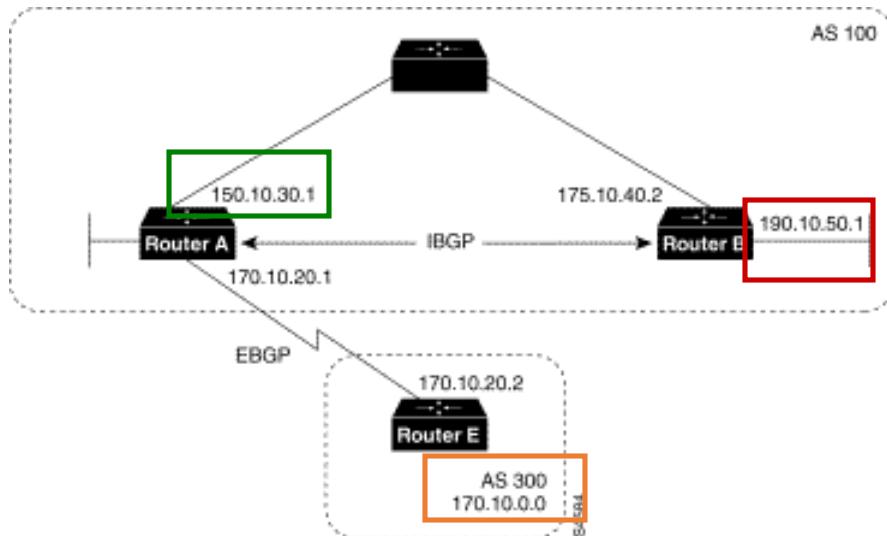
Network	Next Hop	Metric	LocPrf	Weight	Path
*> 192.208.10.0	192.208.10.5	0	0	0	300 i
*> 172.16.1.0	0.0.0.0	0		32768	i

```
R1# show ip bgp
<output omitted>
Network          Next Hop          Metric  LocPrf  Weight  Path
*> 10.1.1.0/24    0.0.0.0          0        32768   ?
*> 192.168.1.0/24 10.1.1.2        84      32768   ?
*> 192.168.2.0/24 10.1.1.2        74      32768   ?
<output omitted>
```

i = Route generated by the **network** command.

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.1.1.0/24	0.0.0.0	0	32768	?	
*> 192.168.1.0/24	10.1.1.2	84	32768	?	
*> 192.168.2.0/24	10.1.1.2	74	32768	?	

? = Route generated by unknown method (usually redistributed).



Router A

```
router bgp 100
neighbor 190.10.50.1 remote-as 100
neighbor 170.10.20.2 remote-as 300
network 150.10.0.0
 redistribute static
ip route 190.10.0.0 255.255.0.0 null 0
```

Router B

```
router bgp 100
neighbor 150.10.30.1 remote-as 100
network 190.10.50.0
```

Router E

```
router bgp 300
neighbor 170.10.20.1 remote-as 100
network 170.10.0.0
```

Given these configurations, the following is true:

- **From Router A**, the route for reaching **170.10.0.0** has an AS_path of 300 and an origin attribute of **IGP**.
- **From Router A**, the route for reaching **190.10.50.0** has an empty AS_path (the route is in the same AS as Router A) and an origin attribute of **IGP**.
- **From Router E**, the route for reaching **150.10.0.0** has an AS_path of 100 and an origin attribute of **IGP**.
- **From Router E**, the route for reaching **190.10.0.0** has an AS_path of 100 and an origin attribute of **Incomplete** (because **190.10.0.0** is a **redistributed** route)

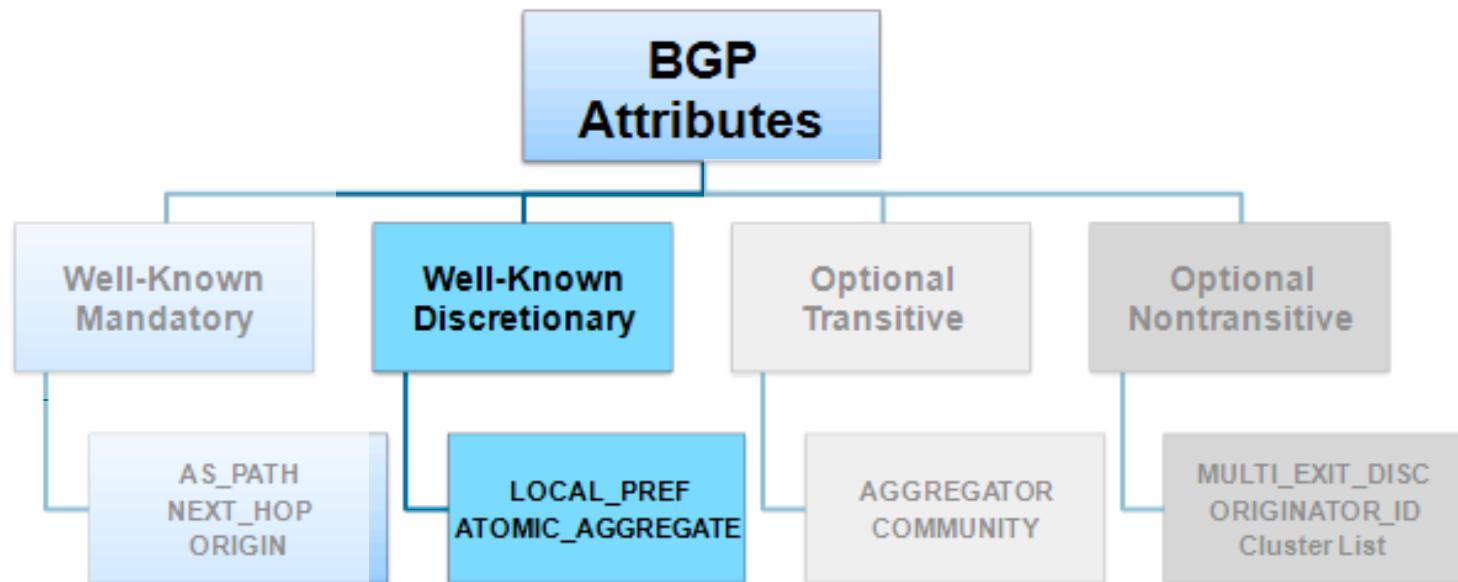
The ORIGIN attribute

- Use a route map and **set origin** command to manipulate the ORIGIN attribute.

```
route-map SETORIGIN permit 10  
    set origin igr
```

Well-Known Discretionary

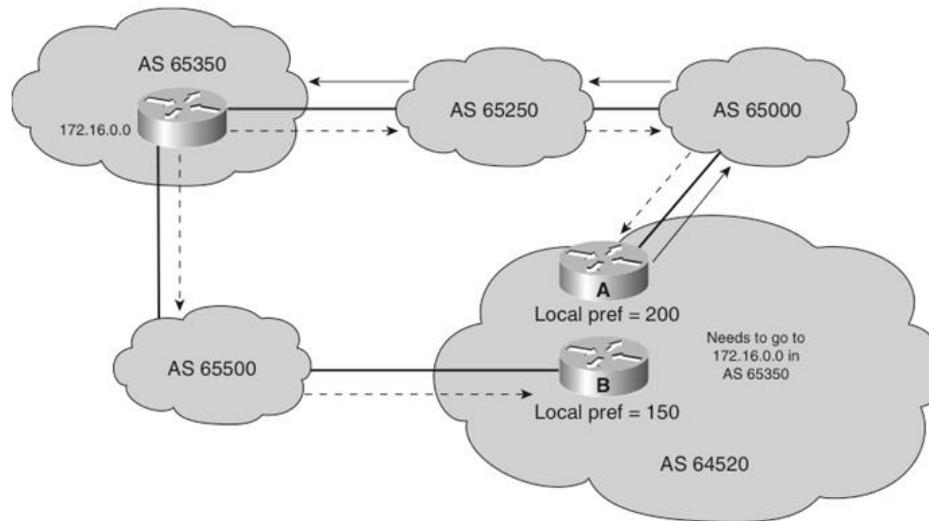
- Attribute is recognized by all implementations of BGP but may not be sent in the BGP update message.



Well-Known Discretionary: LOCAL_PREF

- The Local Preference attribute provides an indication to the “local” routers in the AS about which path is preferred to exit the AS.
 - A path with a higher local preference is preferred.
 - The default value for local preference on a Cisco router is 100.
- It is configured on a router and exchanged between IGP routers.
 - It is not passed to EBGP peers.

Well-Known Discretionary: LOCAL_PREF



- Routers A and B are IBGP neighbors in AS 64520 and both receive updates about network 172.16.0.0 from different directions.
 - The local preference on router A is set to 200.
 - The local preference on router B is set to 150.
- Because the local preference for router A is higher, it is selected as the preferred exit point from AS 64520.

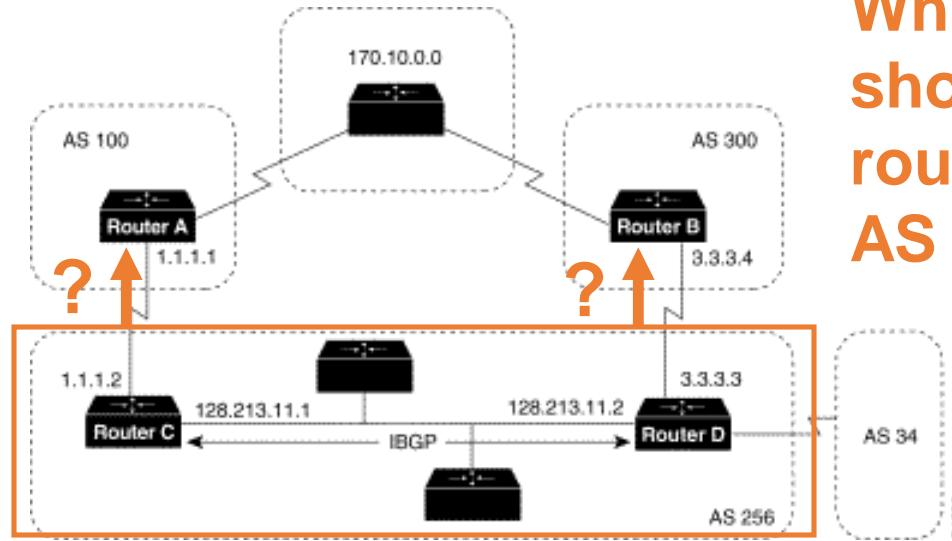
Configuring the Default Local Preference

- The **bgp default local-preference** command changes the default local preference value.
 - With this command, all IBGP routes that are advertised have the local preference set to the value specified.
 - If an EBGP neighbor receives a local preference value, the EBGP neighbor ignores it.

The LOCAL_PREF Attribute

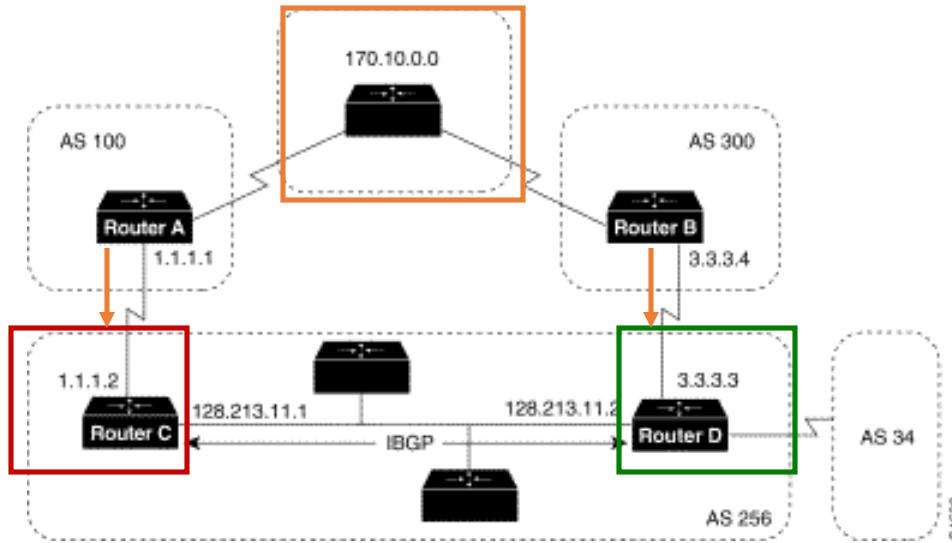
- Well-known discretionary attribute (type code 5).
- Degree of preference given to a route to compare it with other routes for the same destination
 - Higher LOCAL_PREF values are preferred
- **Local to the AS**
 - Exchanged between IBGP peers only
 - It is not advertised to EBGP peers
- **Routers within a multi-homed AS may learn that they can reach the same destination network via neighbors in two (or more) different autonomous systems.**
 - There could be two or more exit points from the local AS to any given destination.
- You can use the **LOCAL_PREF** attribute **to force your BGP routers to prefer one exit point over another** when routing to a particular destination network.

The LOCAL_PREF Attribute

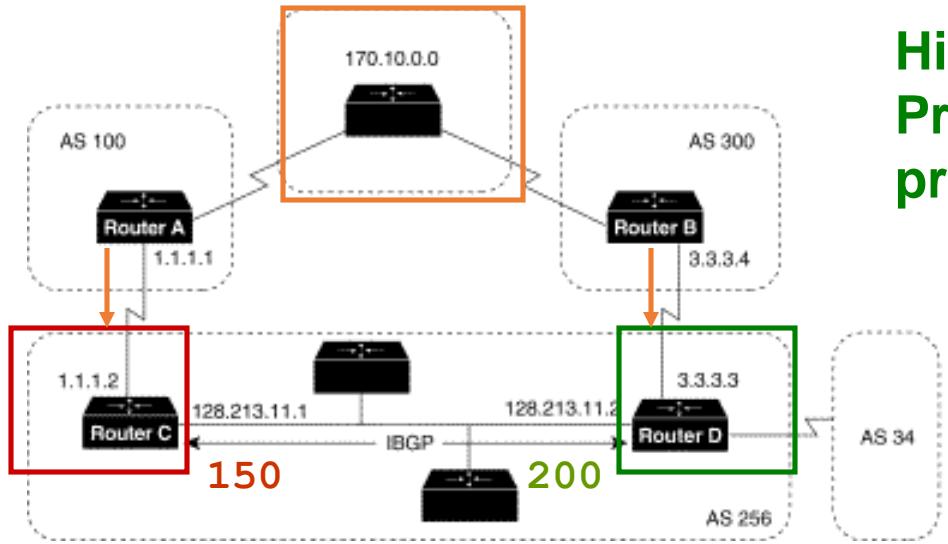


Which exit should all the routers within AS 256 use?

- Because this attribute is communicated within all BGP routers inside the AS, **all BGP routers will have a common view on how to exit the AS**.
- Although routers always prefer the **lowest-route metric and administrative distance** for a given destination, **BGP routers prefer higher LOCAL_PREF values over lower ones**.
- When there are **multiple paths to the same destination, the local preference attribute indicates the preferred path**.
- The path with the **higher preference is preferred** (the **default value of the local preference attribute is 100**).
- Unlike the **weight attribute**, which is only **relevant to the local router**, the **local preference attribute is part of the routing update and is exchanged among routers in the same AS**.



- AS 256 receives route updates for network **170.10.0.0** from **AS 100** and **AS 300**.
- There are two ways to set local preference:
 - Using the **bgp default local-preference** command
 - Using a **Route Map** to Set Local Preference - **FYI**



Higher Local Preference is preferred!

Using the `bgp default local-preference` Command

- The following configurations use the `bgp default local-preference` router configuration command to set the local preference attribute on Routers C and D:

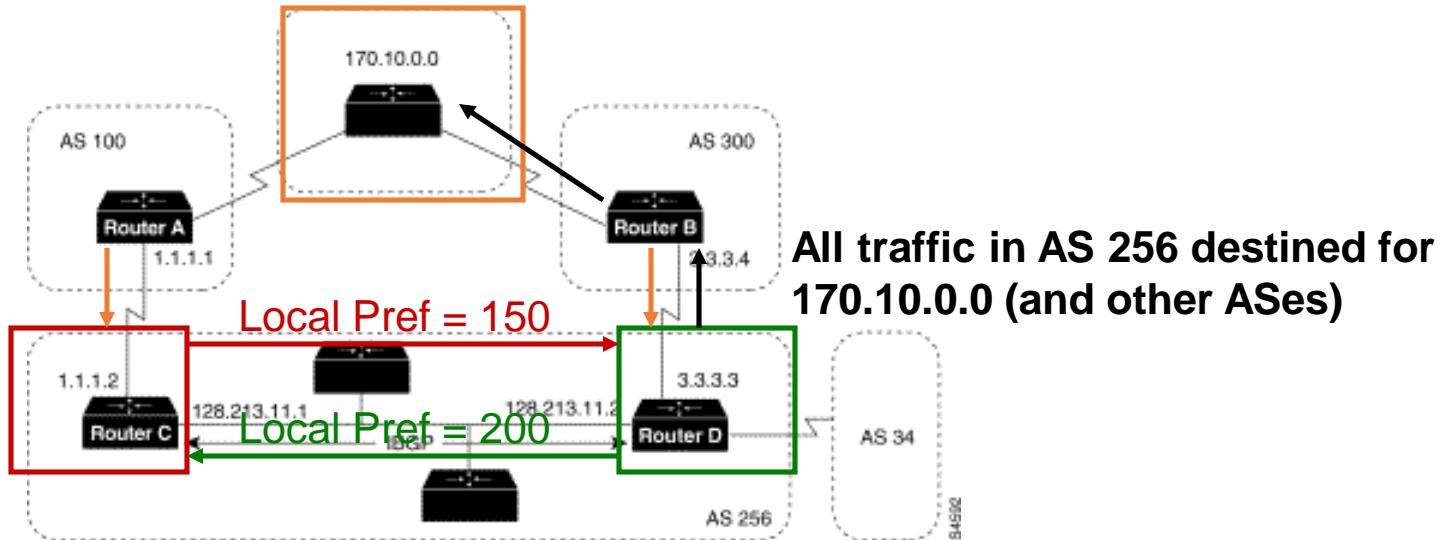
Router C

```
router bgp 256
    neighbor 1.1.1.1 remote-as 100
    neighbor 128.213.11.2 remote-as 256
        bgp default local-preference 150
```

Router D

```
router bgp 256
    neighbor 3.3.3.4 remote-as 300
    neighbor 128.213.11.1 remote-as 256
        bgp default local-preference 200
```

Higher Local Preference is preferred!



Router C

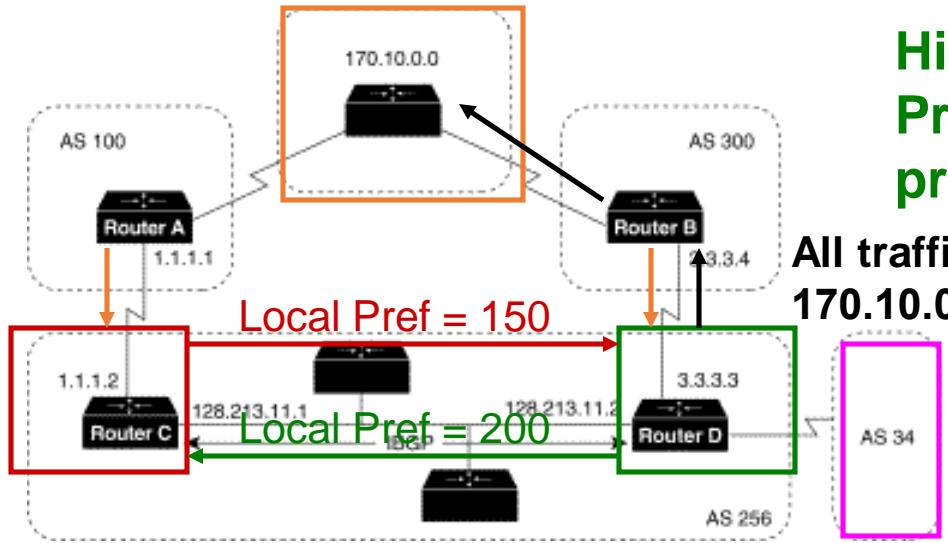
```
router bgp 256
  bgp default local-preference 150
```

Router D

```
router bgp 256
  bgp default local-preference 200
```

Higher Local Preference is preferred!

- The configuration for **Router C** causes it to set the **local preference** of all updates from **AS 300** to **150 (routes learned from RouterD)**, and the configuration for **Router D** causes it to set the **local preference** for all updates from **AS 100** to **200 (routes learned from RouterC)**.
- Because **local preference** is exchanged within the AS, both Routers C and D determine that updates regarding network **170.10.0.0** have a higher **local preference** when they come from **AS 300** than when they come from AS 100.
- As a **result, all traffic in AS 256 destined for network 170.10.0.0 is sent to Router D as the exit point.**

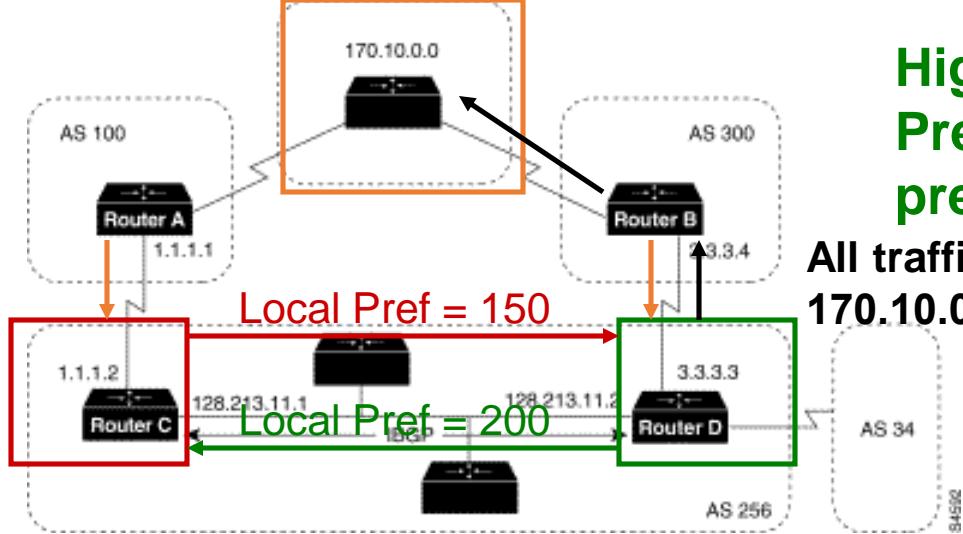


Higher Local Preference is preferred!

All traffic in AS 256 destined for 170.10.0.0 (and other AS's)

Using a Route Map to Set Local Preference - FYI

- Route maps provide more flexibility than the `bgp default local-preference` router configuration command.
- When the `bgp default local-preference` command is used on Router D, the local preference attribute of all updates received by Router D will be set to 200, including updates from AS 34.



Higher Local Preference is preferred!

All traffic in AS 256 destined for 170.10.0.0 (and other AS's)

- The following configuration uses a route map to set the **local preference** attribute on Router D specifically for updates regarding AS 300:

Router D

```

router bgp 256

neighbor 3.3.3.4 remote-as 300
route-map SETLOCALIN in
neighbor 128.213.11.1 remote-as 256
ip as-path 7 permit ^300$
route-map SETLOCALIN permit 10
match as-path 7
set local-preference 200
route-map SETLOCALIN permit 20
    
```

- With this configuration, the **local preference** attribute of any update coming **from AS 300 is set to 200.**
- Instance 20 of the SETLOCALIN route map accepts all other routes.

Well-Known Discretionary: ATOMIC_AGGREGATOR

- The Atomic Aggregate attribute is used to indicate that routes have been summarized.
 - Attribute warns that the received information may not necessarily be the most complete route information available.
- Attribute is set to either True or False with “true” alerting other BGP routers that multiple destinations have been grouped into a single update.
 - Router update includes its router ID and AS number along with the supernet route enabling administrators to determine which BGP router is responsible for a particular instance of aggregation.
 - Tracing a supernet to its original "aggregator" may be necessary for troubleshooting purposes.

ATOMIC_AGGREGATE

- This attribute uses the **aggregate-address** command.
- A BGP speaking router can transmit overlapping routes to another BGP speaker.
- **Overlapping routes are non-identical routes that point to the same destination.**
- For example, 206.25.192.0/19 and 206.25.128.0/17 are overlapping, as the first route is included in the second route.
- The second route, 206.25.128.0/17, points to other more specific routes besides 206.25.192.0/19.
- When making a best path decision, a router always chooses the **more-specific path**.
- When advertising routes, however, the BGP speaker has several options with overlapping routes.

ATOMIC_ AGGREGATE

Choices:

- Advertise both the more-specific and the less-specific route
- Advertise only the more-specific route
- Advertise only the non-overlapping part of the route
- Aggregate (summarize) the two routes and advertise the aggregate
- Advertise the less-specific route only
- Advertise neither route.

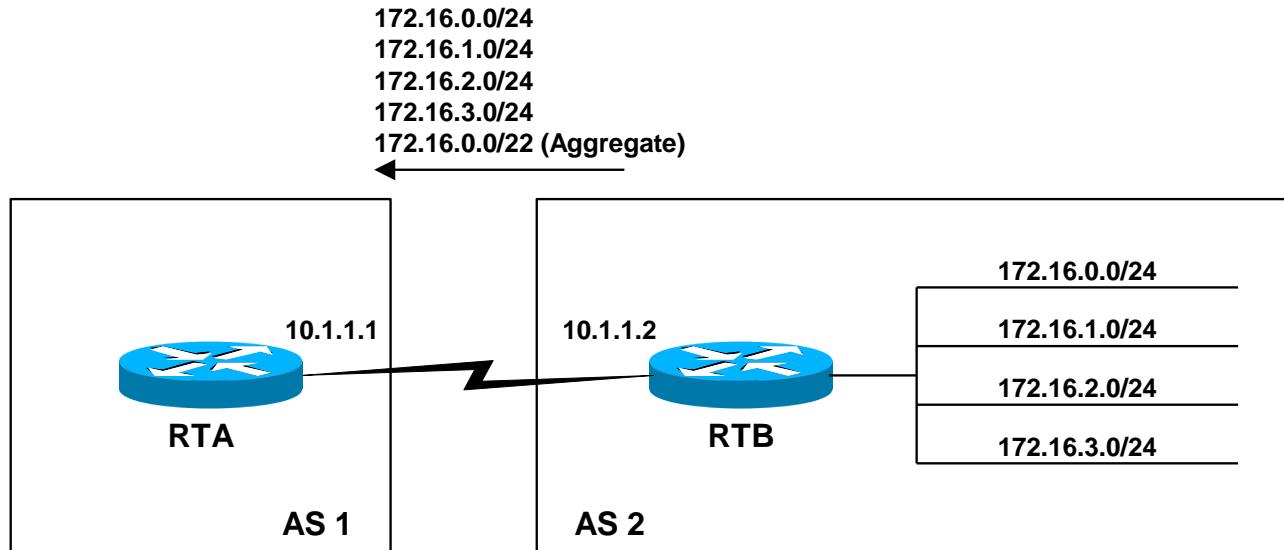
ATOMIC_AGGREGATE

- The **ATOMIC_AGGREGATE** is a well-known discretionary attribute (type code 6).
- The **ATOMIC_AGGREGATE** attribute is set to either “**True**” or “**False**.”
- If **true**, this attribute alerts BGP routers that multiple destinations have been grouped into a single update.
 - In other words, the **BGP router that sent the update had a more specific route to the destination, but did not send it.**
 - **ATOMIC_AGGREGATE** warns receiving routers that the information they are receiving is **not necessarily the most complete route information available.**

You can manually configure BGP to summarize routes by using the **aggregate-address** command, which has the following syntax:

```
Router(config-router)#aggregate-address address mask [as-set] [summary-only] [suppress-map map-name] [advertise-map map-name] [attribute-map map-name]
```

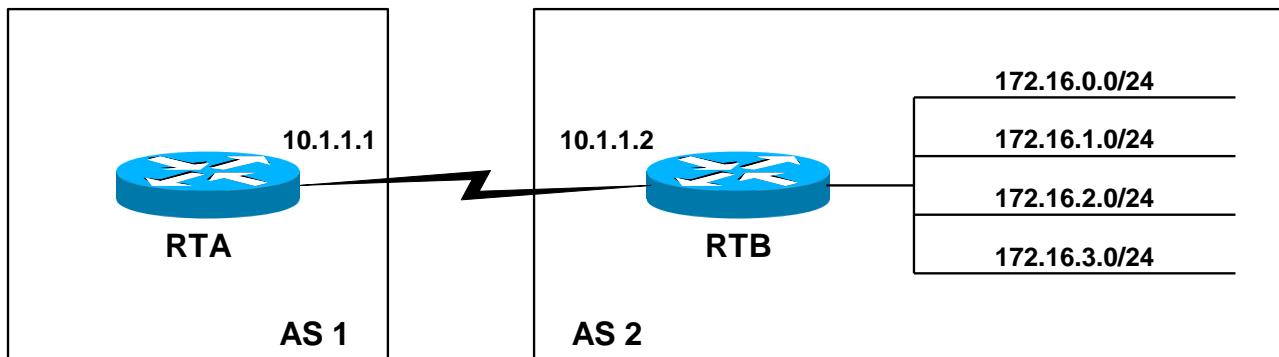
ATOMIC_AGGREGATE



- The purpose of this command is to create an **aggregate** (summarized) entry in the BGP table.
- There are two ways to create an aggregate address under BGP:
 1. Create a static entry in the routing table for the aggregate address and then advertise it with the network command.
 2. Use the aggregate-address command.
- An aggregate is created only if a more-specific route to the aggregate exists in the BGP table.

172.16.0.0/24
172.16.1.0/24
172.16.2.0/24
172.16.3.0/24
172.16.0.0/22 (Aggregate)

Example 1: Aggregating Local Routes



RTA

```
router bgp 1
neighbor 10.1.1.2 remote-as 2
```

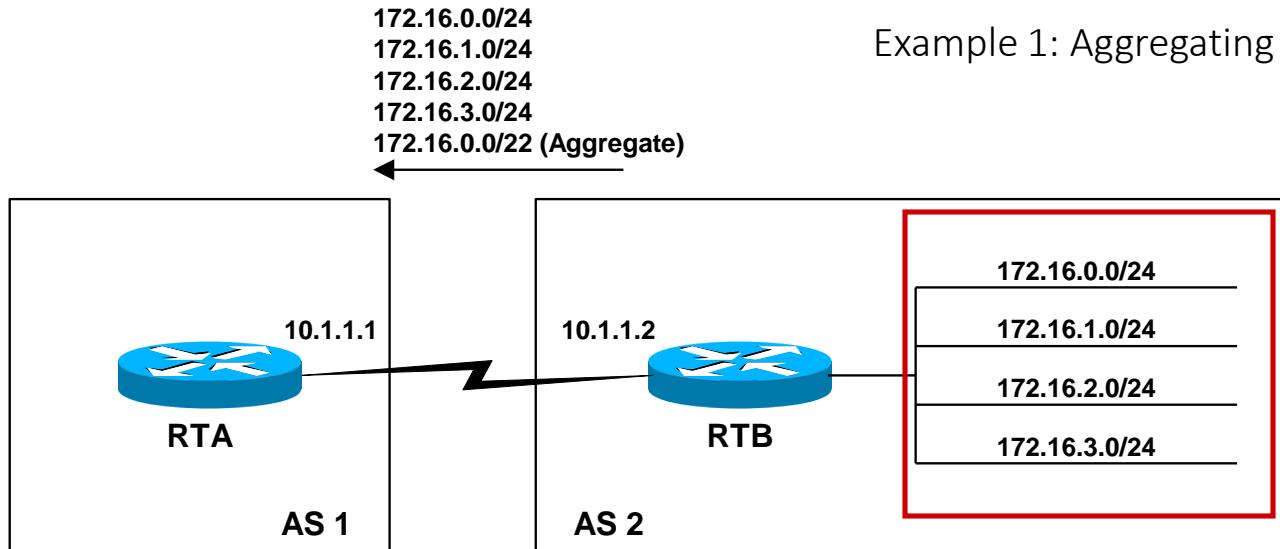
RTB

```
router bgp 2
neighbor 10.1.1.1 remote-as 1
network 172.16.0.0 mask {/24}
network 172.16.1.0 mask {/24}
network 172.16.2.0 mask {/24}
network 172.16.3.0 mask {/24}
```

Before aggregating locally sourced routes,
lets configure the more-specific networks.

- RTB has four loopbacks used to simulate the networks along with BGP network commands.
- RTA and RTB will have all 172.16.n.0/24 routes in its BGP table (show ip bgp)

Example 1: Aggregating Local Routes



RTB

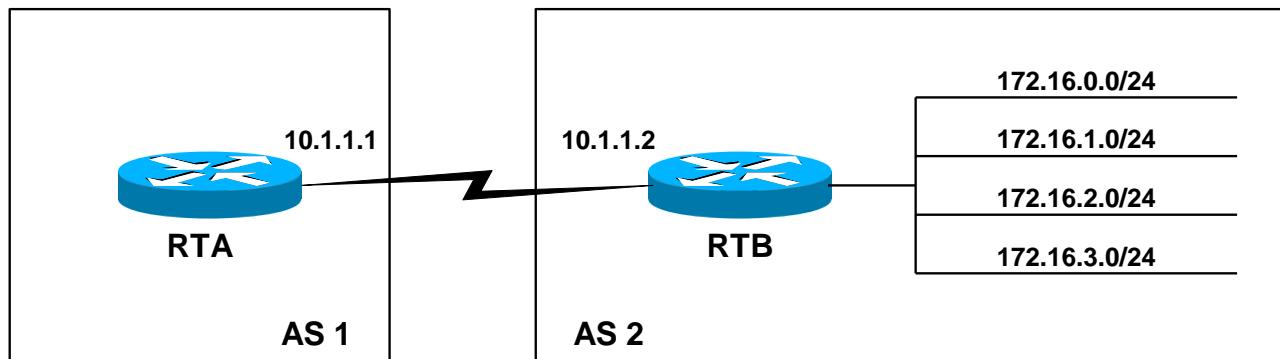
```
router bgp 2  
  
neighbor 10.1.1.1 remote-as 1  
  
network 172.16.0.0 mask {/24}  
network 172.16.1.0 mask {/24}  
network 172.16.2.0 mask {/24}  
network 172.16.3.0 mask {/24}  
aggregate-address 172.16.0.0  
255.255.252.0 {/22}
```

Now modify the BGP on RGB to enable the advertisement of the aggregate:

- We need only one of the more-specific network commands in RTB in order to send the aggregate, but by configuring all of them **the aggregate will be sent in case one of the networks goes down.**
- RTA and RTB will have all 172.16.n.0/22 routes in its BGP table (show ip bgp), and the the aggregate address of 172.16.0.0/22

172.16.0.0/24
172.16.1.0/24
172.16.2.0/24
172.16.3.0/24
172.16.0.0/22 (Aggregate)

Example 1: Aggregating Local Routes



show ip bgp 172.16.0.0 will display that this route has the “atomic-aggregate” attribute set.

```
RTA#show ip bgp 172.16.0.0 255.255.252.0
```

BGP routing table entry for 172.16.0.0/22, version 18

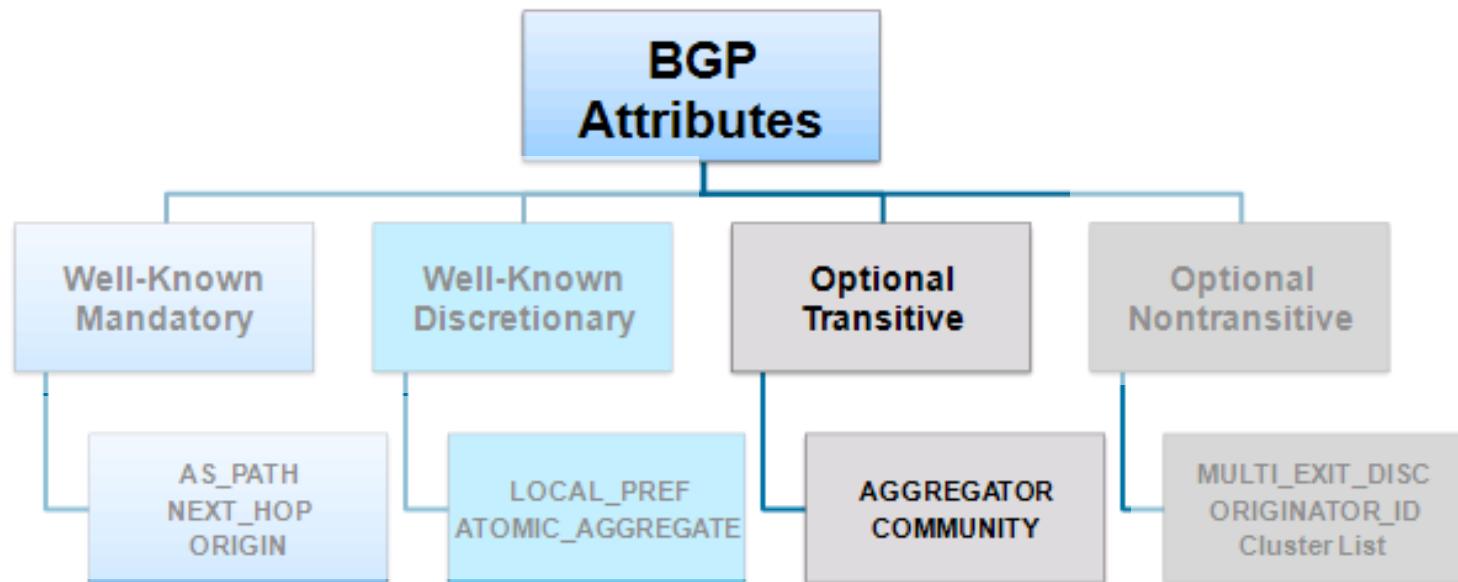
Paths: (1 available, best #1)

<text omitted>

Origin IGP, localpref 100, valid, external, **atomic- aggregate**, best

Optional Transitive

- Attribute may or may not be recognized by all BGP implementations.
- Because the attribute is transitive, BGP accepts and advertises the attribute even if it is not recognized.

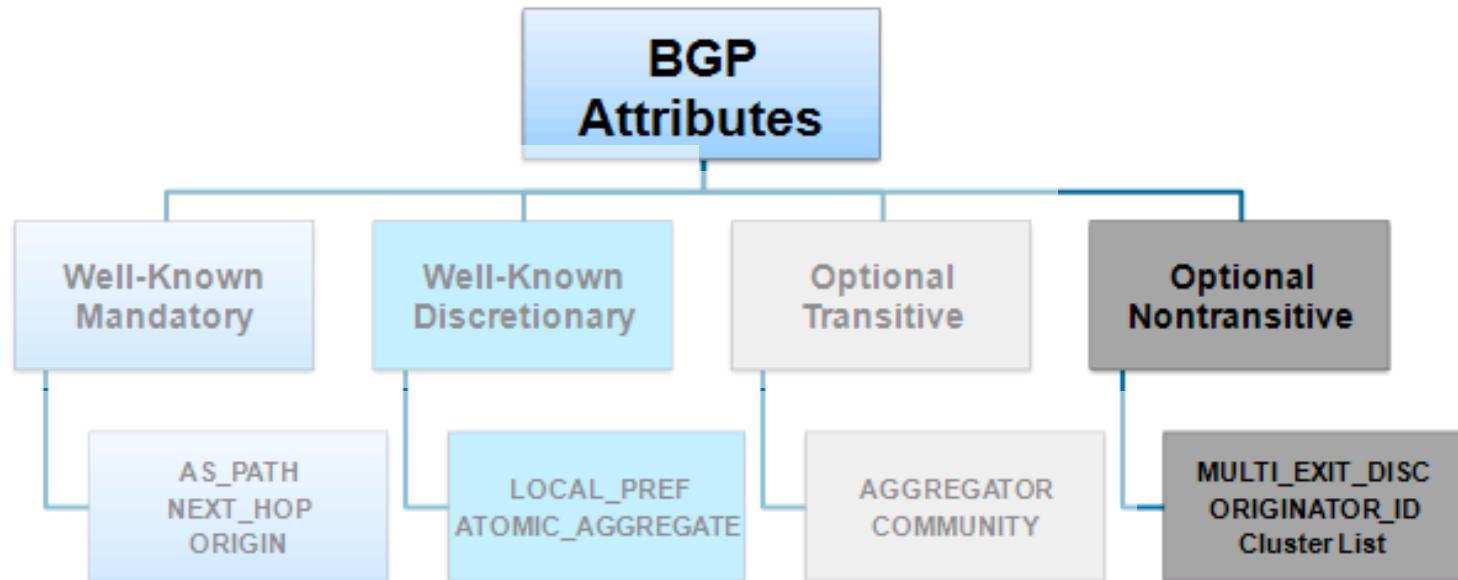


Optional Transitive: Community

- The BGP community attribute can be used to filter incoming or outgoing routes.
 - BGP routers can tag routes with an indicator (the community) and allow other routers to make decisions based on that tag.
- If a router does not understand the concept of communities, it defers to the next router.
 - However, if the router does understand the concept, it must be configured to propagate the community; otherwise, communities are dropped by default.
- Communities are not restricted to one network or one AS, and they have no physical boundaries.

Optional Nontransitive

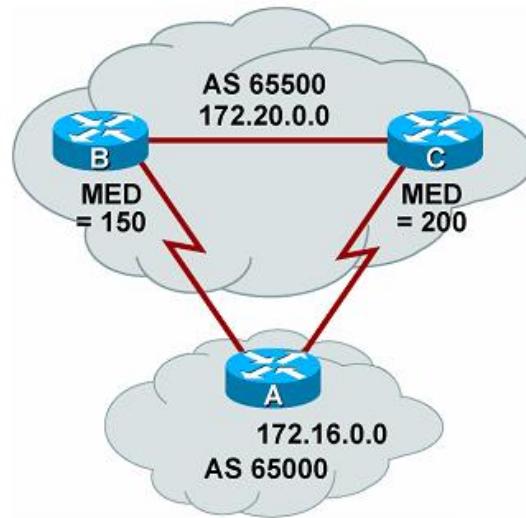
- Attribute that may or may not be recognized by all BGP implementations.
- Whether or not the receiving BGP router recognizes the attribute, it is nontransitive and is not passed along to other BGP peers.



Optional Nontransitive: MED

- The Multiple Exit Discriminator (MED) attribute, also called the *metric*, provides a hint to external neighbors about the preferred path into an AS that has multiple entry points.
 - Lower MED is preferred over a higher MED!
- The MED is sent to EBGP peers and those routers propagate the MED within their AS.
 - The routers within the AS use the MED, but do not pass it on to the next AS.
 - When the same update is passed on to another AS, the metric will be set back to the default of 0.
- By using the MED attribute, BGP is the only protocol that can affect how routes are sent into an AS.

Optional Nontransitive: MED



- Routers B and C include a MED attribute in the updates to router A.
 - Router B MED attribute is set to 150.
 - Router C MED attribute is set to 200.
- When A receives updates from B and C, it picks router B as the best next hop because of the lower MED.

The MED attribute

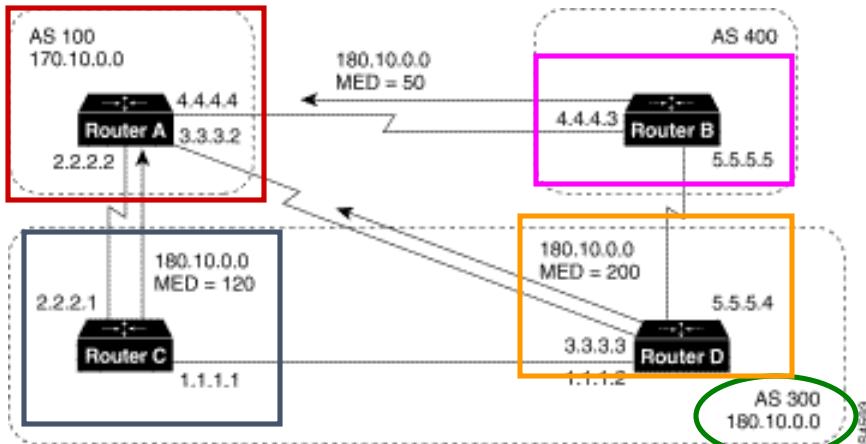
- The MULTI_EXIT_DISC (Multi-Exit Discriminator) attribute is an optional non-transitive attribute (type code 4).
- **Informs external neighbors about the preferred path** into an AS that has multiple entry points.
- A lower MULTI_EXIT_DISC (or MED) is preferred over a higher MED.

The MED attribute

Multi-Exit Discriminator Attribute

- The multi-exit discriminator (MED) attribute is a ***hint*** to external neighbors about the preferred path **into an AS** when there are **multiple entry points** into the AS.
- A **lower MED value is preferred** over a higher MED value.
- The **default** value of the **MED** attribute is **0**.
- Unlike local preference, the **MED attribute is exchanged between AS's**, but a **MED attribute that comes into an AS does not leave the AS**.
- When an update enters the AS with a certain MED value, that value is used for decision making within the AS.
- When BGP sends that update to another AS, the MED is reset to 0.
- Unless otherwise specified, the router compares MED attributes for paths from external neighbors that are in the same AS.
- **If you want MED attributes from neighbors in other ASes to be compared**, you must configure the **bgp always-compare-med** command.

- **AS 100** receives updates regarding network **180.10.0.0** from Routers **B**, **C**, and **D**.
- Routers C and D are in AS 300, and Router B is in AS 400.



Router A

```
router bgp 100
  neighbor 2.2.2.1 remote-as 300
  neighbor 3.3.3.3 remote-as 300
  neighbor 4.4.4.3 remote-as 400
```

Router B

```
router bgp 400
  neighbor 4.4.4.4 remote-as 100
  neighbor 4.4.4.4 route-map
    SETMEDOUT out
  neighbor 5.5.5.4 remote-as 300
route-map SETMEDOUT permit 10
  set metric 50
```

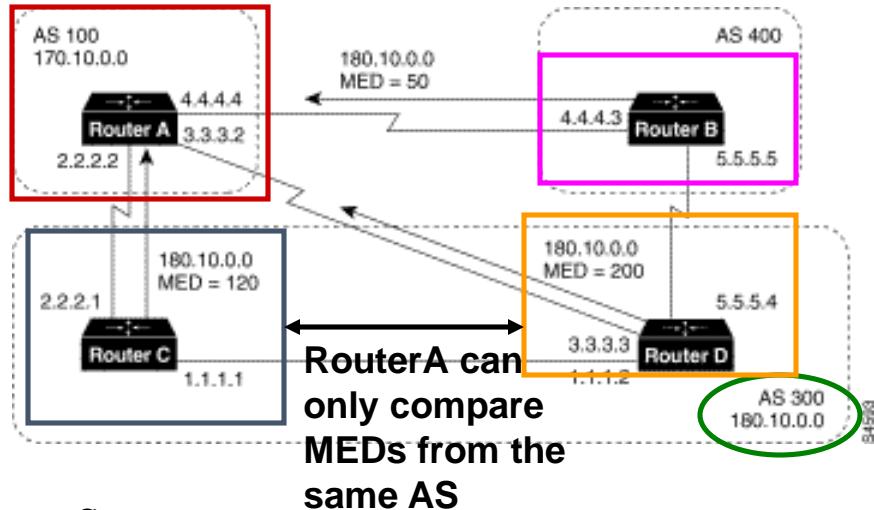
Router C

```
router bgp 300
  neighbor 2.2.2.2 remote-as 100
  neighbor 2.2.2.2 route-map SETMEDOUT out
  neighbor 5.5.5.5 remote-as 400
  neighbor 1.1.1.2 remote-as 300
route-map SETMEDOUT permit 10
  set metric 120
```

Router D

```
router bgp 300
  neighbor 3.3.3.2 remote-as 100
  neighbor 3.3.3.2 route-map SETMEDOUT out
  neighbor 1.1.1.1 remote-as 300
route-map SETMEDOUT permit 10
  set metric 200
```

- By default, BGP compares the MED attributes of routes coming from neighbors in the same external AS *as the route* (such as AS 300).
- Router A can only compare the MED attribute coming from Router C (120) to the MED attribute coming from Router D (200) even though the update coming from Router B has the lowest MED value.



Router A

```
router bgp 100
  neighbor 2.2.2.1 remote-as 300
  neighbor 3.3.3.3 remote-as 300
  neighbor 4.4.4.3 remote-as 400
```

Router B

```
router bgp 400
  neighbor 4.4.4.4 remote-as 100
  neighbor 4.4.4.4 route-map
    SETMEDOUT out
  neighbor 5.5.5.4 remote-as 300
route-map SETMEDOUT permit 10
  set metric 50
```

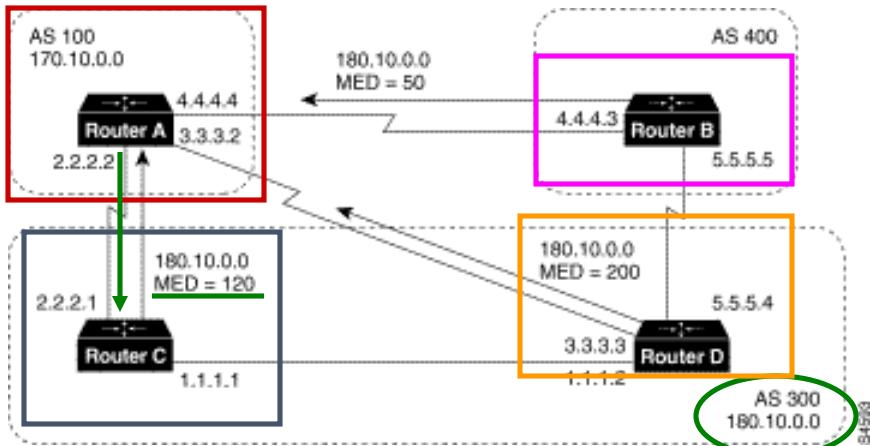
Router C

```
router bgp 300
  neighbor 2.2.2.2 remote-as 100
  neighbor 2.2.2.2 route-map SETMEDOUT out
  neighbor 5.5.5.5 remote-as 400
  neighbor 1.1.1.2 remote-as 300
route-map SETMEDOUT permit 10
  set metric 120
```

Router D

```
router bgp 300
  neighbor 3.3.3.2 remote-as 100
  neighbor 3.3.3.2 route-map SETMEDOUT out
  neighbor 1.1.1.1 remote-as 300
route-map SETMEDOUT permit 10
  set metric 200
```

- Router A will choose Router C as the best path for reaching network 180.10.0.0.



Router A

```
router bgp 100
neighbor 2.2.2.1 remote-as 300
neighbor 3.3.3.3 remote-as 300
neighbor 4.4.4.3 remote-as 400
```

Router B

```
router bgp 400
neighbor 4.4.4.4 remote-as 100
neighbor 4.4.4.4 route-map
      SETMEDOUT out
neighbor 5.5.5.4 remote-as 300
route-map SETMEDOUT permit 10
      set metric 50
```

Router C

```
router bgp 300
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 route-map SETMEDOUT out
neighbor 5.5.5.5 remote-as 400
neighbor 1.1.1.2 remote-as 300
route-map SETMEDOUT permit 10
      set metric 120
```

Router D

```
router bgp 300
neighbor 3.3.3.2 remote-as 100
neighbor 3.3.3.2 route map SETMEDOUT out
neighbor 1.1.1.1 remote-as 300
route-map SETMEDOUT permit 10
      set metric 200
```

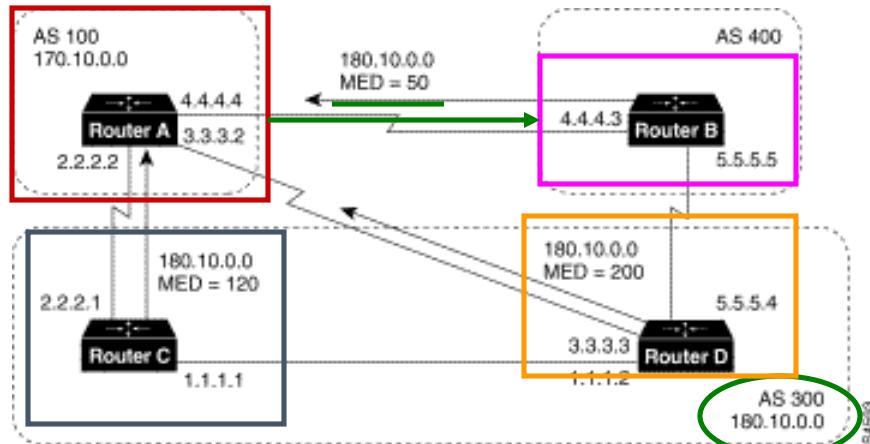
- To force **Router A** to include updates for network **180.10.0.0** from **Router B** in the comparison, use the **bgp always-compare-med** router configuration command on **Router A**:
- Router A will choose Router B** as the best next hop for reaching network **180.10.0.0** (assuming that all other attributes are the same).

Router A

```
router bgp 100
neighbor 2.2.2.1 remote-as 300
neighbor 3.3.3.3 remote-as 300
neighbor 4.4.4.3 remote-as 400
bgp always-compare-med
```

Router B

```
router bgp 400
neighbor 4.4.4.4 remote-as 100
neighbor 4.4.4.4 route-map
    SETMEDOUT out
neighbor 5.5.5.4 remote-as 300
route-map SETMEDOUT permit 10
set metric 50
```



Router C

```
router bgp 300
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 route-map SETMEDOUT out
neighbor 5.5.5.5 remote-as 400
neighbor 1.1.1.2 remote-as 300
route-map SETMEDOUT permit 10
    set metric 120
```

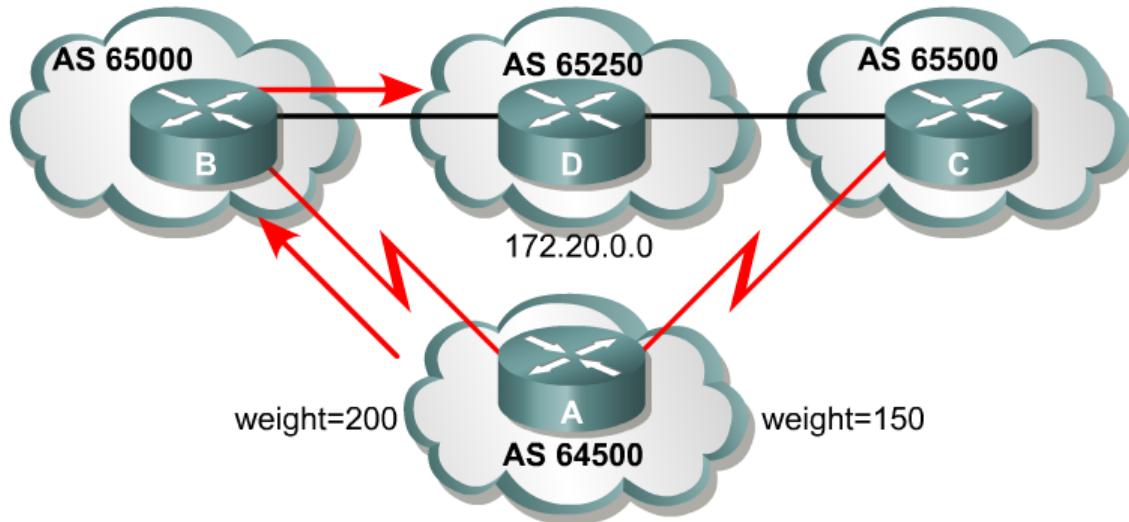
Router D

```
router bgp 300
neighbor 3.3.3.2 remote-as 100
neighbor 3.3.3.2 route-map SETMEDOUT out
neighbor 1.1.1.1 remote-as 300
route-map SETMEDOUT permit 10
    set metric 200
```

Cisco Weight Attribute

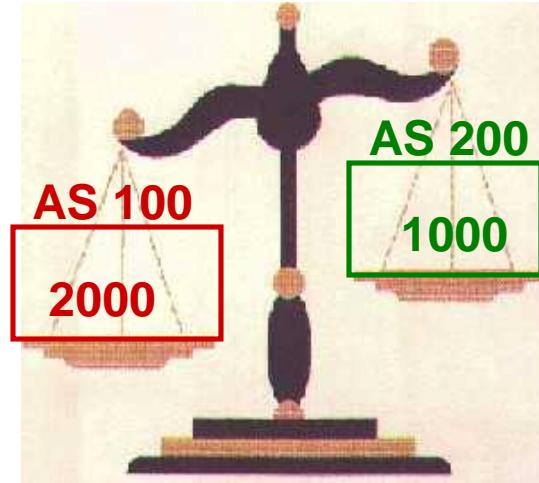
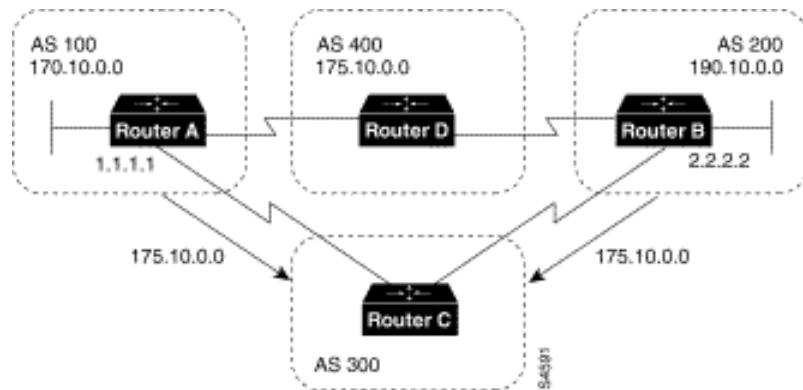
- The Weight attribute is a Cisco proprietary attribute.
- Similar in function to the local preference, the weight attribute applies when 1 router has multiple exit points.
 - Local preference is used when 2+ routers provide multiple exit points.
- It is configured locally on a router and is not propagated to any other routers.
 - Routes with a higher weight are preferred when multiple routes exist to the same destination.
- The weight can have a value from 0 to 65535.
 - Paths that the router originates have a weight of 32768 by default, and other paths have a weight of 0 by default.

Cisco Weight Attribute



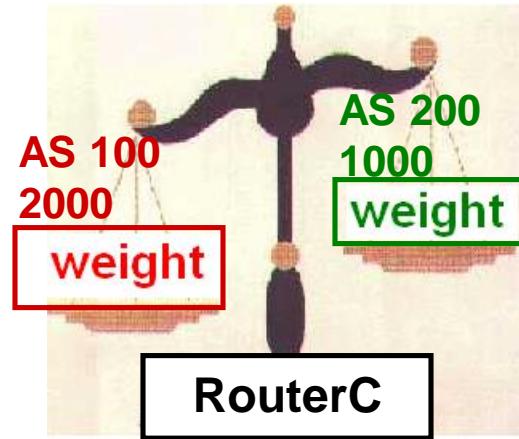
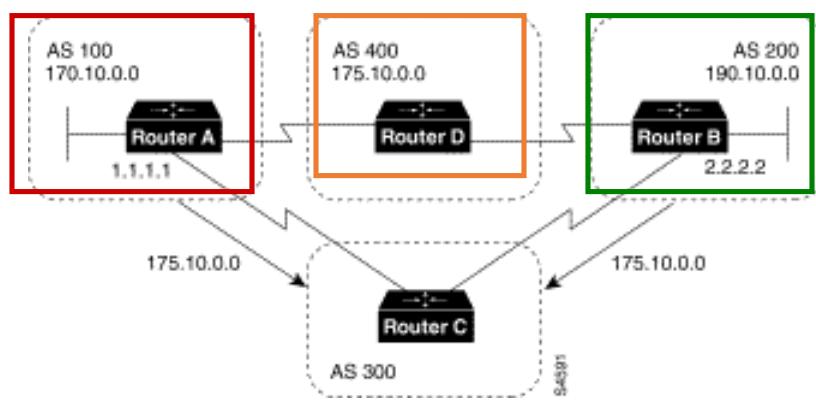
- Routers B and C learn about network 172.20.0.0 from AS 65250 and propagate the update to router A.
 - Therefore Router A has two ways to reach 172.20.0.0.
- Router A sets the weight of updates as follows:
 - Updates coming from router B are set to 200
 - Updates coming from router C are set to 150.
- Router A uses router B because of the higher weight.

The WEIGHT attribute



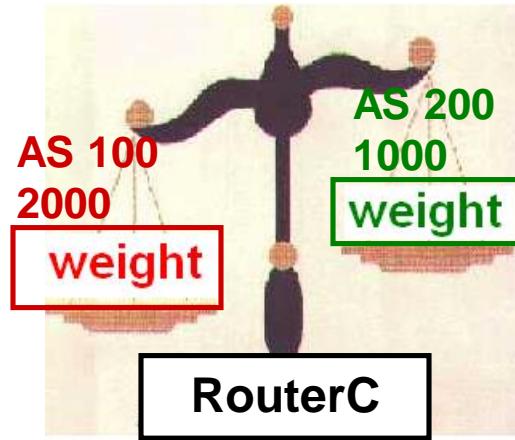
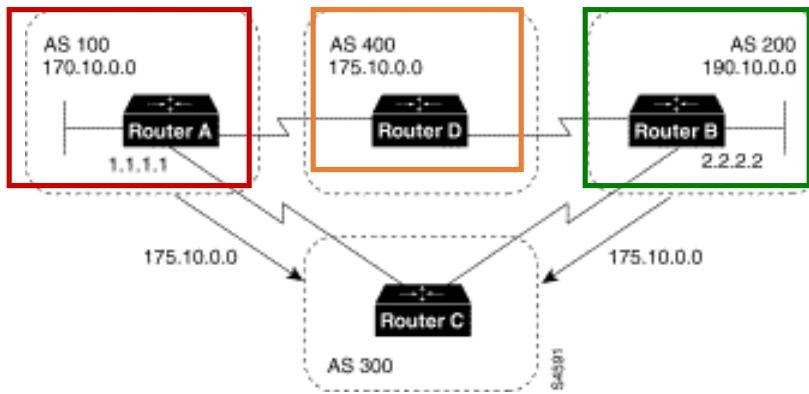
- The weight attribute is a **special Cisco attribute** that is used in the path selection process **when there is more than one route to the same destination**.
- The weight attribute is **local to the router on which it is assigned**, and it is not propagated in routing updates.
- By **default**, the weight attribute is **32768** for paths that the router originates and **zero** for other paths.
- Routes with a **higher weight** are **preferred** when there are multiple routes to the same destination.

The WEIGHT attribute



- **Router A** and **Router B** learn about network **175.10.0.0** from **AS 400**, and each propagates the update to **Router C**.
- **Router C** has two routes for reaching **175.10.0.0** and has to decide which route to use.
- If, on Router C, you set the weight of the updates coming in from **Router A** to be higher than the updates coming in from **Router B**, **Router C** will use **Router A** as the next hop to reach network **175.10.0.0**.

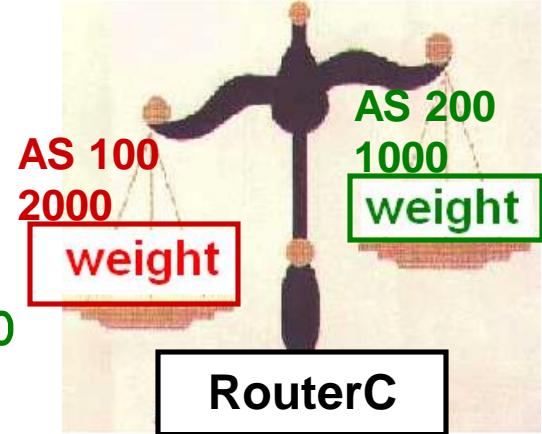
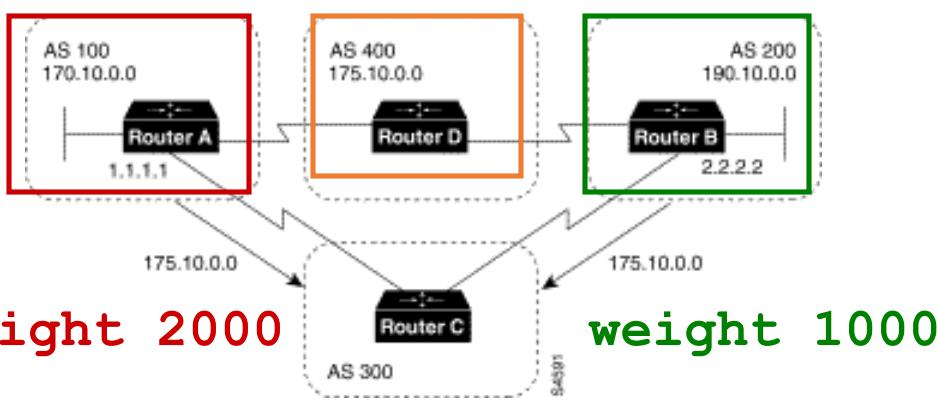
The WEIGHT attribute



- There are three ways to set the weight for updates coming in from Router A:
 - Using the **neighbor weight** Command to Set the Weight Attribute
 - What we will use.
 - Because of time reasons, we will only discuss this option.
 - Using an **Access List** to Set the Weight Attribute
 - FYI
 - Using a **Route Map** to Set the Weight Attribute
 - FYI

Higher weight preferred

→ **weight 2000**



Using the neighbor weight Command to Set the Weight Attribute

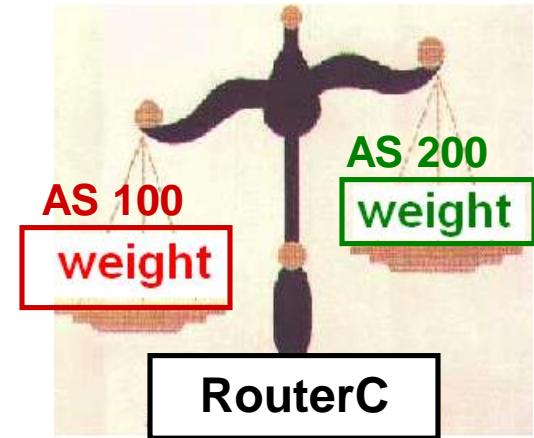
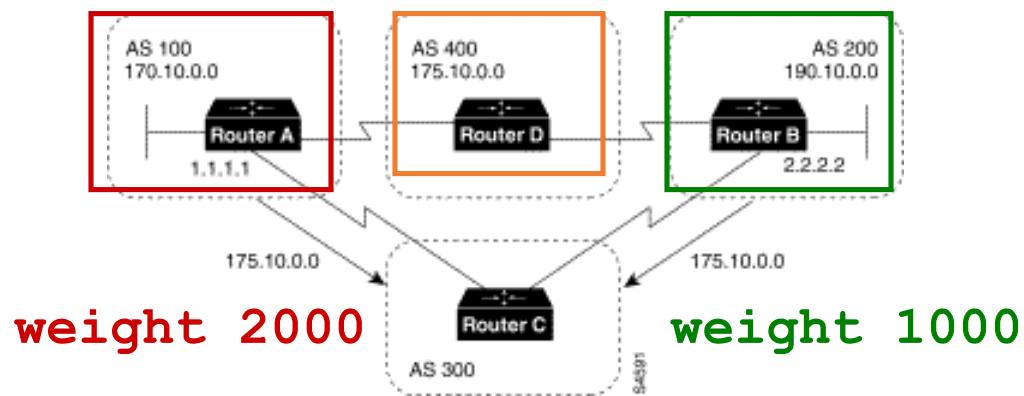
- The following configuration for Router C uses the neighbor weight router configuration command:

Router C

```
router bgp 300
    neighbor 1.1.1.1 remote-as 100
    neighbor 1.1.1.1 weight 2000
    neighbor 2.2.2.2 remote-as 200
    neighbor 2.2.2.2 weight 1000
```

- This configuration sets the **weight** of all route updates from **AS 100 to 2000**, and the **weight** of all route updates coming from **AS 200 to 1000**.
- Result:** The higher **weight** assigned to route updates from AS 100 causes Router C to send traffic through **Router A**.

The WEIGHT attribute



Using an Access List to Set the Weight Attribute - FYI

- The following commands on Router C use access lists and the value of the AS_path attribute to assign a weight to route updates:

Router C

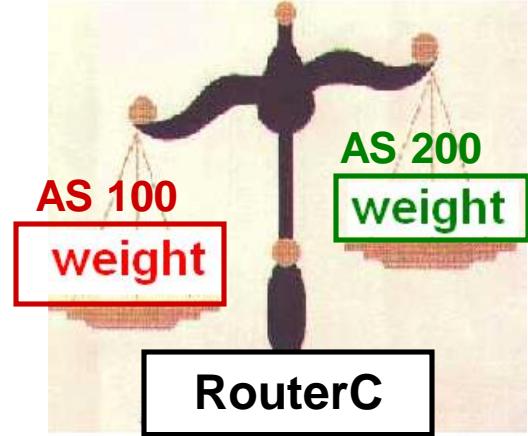
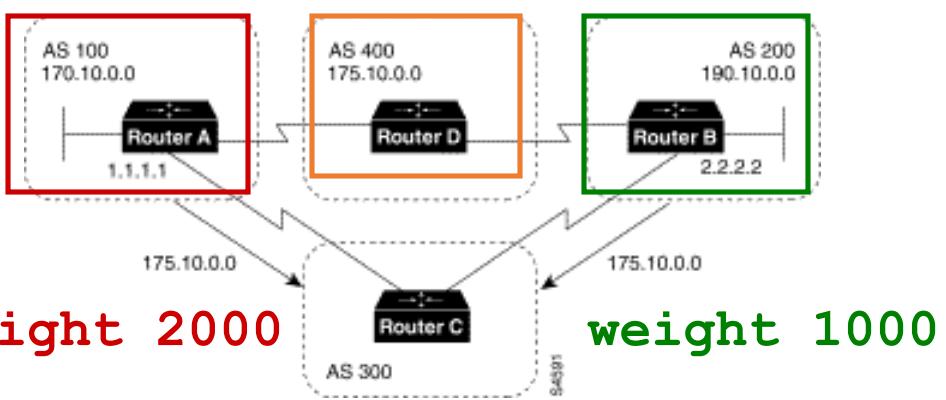
```
router bgp 300
    neighbor 1.1.1.1 remote-as 100
    neighbor 1.1.1.1 filter-list 5 weight 2000
    neighbor 2.2.2.2 remote-as 200
    neighbor 2.2.2.2 filter-list 6 weight 1000
    ip as-path access-list 5 permit ^100$
    ip as-path access-list 6 permit ^200$
```

The WEIGHT attribute – Access list

Router C

```
router bgp 300  
    neighbor 1.1.1.1 remote-as 100  
        neighbor 1.1.1.1 filter-list 5 weight 2000  
    neighbor 2.2.2.2 remote-as 200  
        neighbor 2.2.2.2 filter-list 6 weight 1000  
ip as-path access-list 5 permit ^100$  
ip as-path access-list 6 permit ^200$
```

- In this example, **2000 is assigned to the weight attribute of updates from the neighbor at IP address 1.1.1.1 that are permitted by access list 5.**
- Access list 5 permits updates whose AS_path attribute starts with 100 (as specified by ^) and ends with 100 (as specified by \$). (The ^ and \$ symbols are used to form regular expressions. For a complete explanation of regular expressions, see the appendix on regular expressions in the Cisco Internetwork Operating System (Cisco IOS) software configuration guides and command references.)
- This example also **assigns 1000 to the weight attribute of updates from the neighbor at IP address 2.2.2.2 that are permitted by access list 6.** Access list 6 permits updates whose AS_path attribute starts with 200 and ends with 200.
- In effect, this configuration assigns 2000 to the weight attribute of all route updates received from AS 100 and assigns 1000 to the weight attribute of all route updates from AS 200.



Using a Route Map to Set the Weight Attribute - FYI

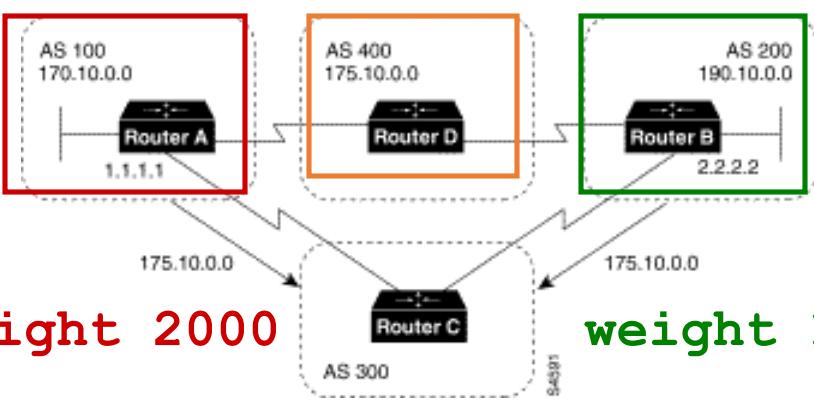
- The following commands on Router C use a route map to assign a weight to route updates:

Router C

```

router bgp 300
  neighbor 1.1.1.1 remote-as 100
  neighbor 1.1.1.1 route-map SETWEIGHTIN in
  neighbor 2.2.2.2 remote-as 200
  neighbor 2.2.2.2 route-map SETWEIGHTIN in
  ip as-path access-list 5 permit ^100$
route-map SETWEIGHTIN permit 10
  match as-path 5
  set weight 2000
route-map SETWEIGHTIN permit 20
  set weight 1000

```



Router C

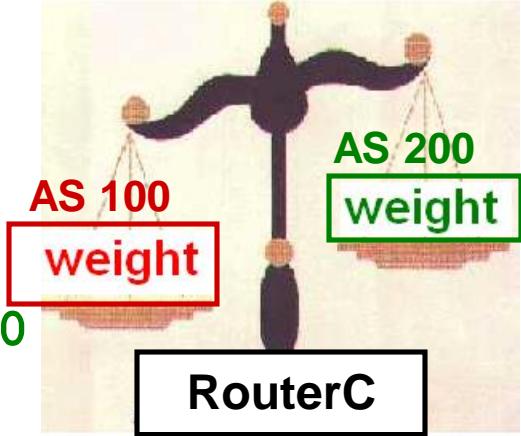
router bgp 300

```

neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 route-map SETWEIGHTIN in
neighbor 2.2.2.2 remote-as 200
neighbor 2.2.2.2 route-map SETWEIGHTIN in
ip as-path access-list 5 permit ^100$
route-map SETWEIGHTIN permit 10
match as-path 5
set weight 2000
route-map SETWEIGHTIN permit 20
set weight 1000

```

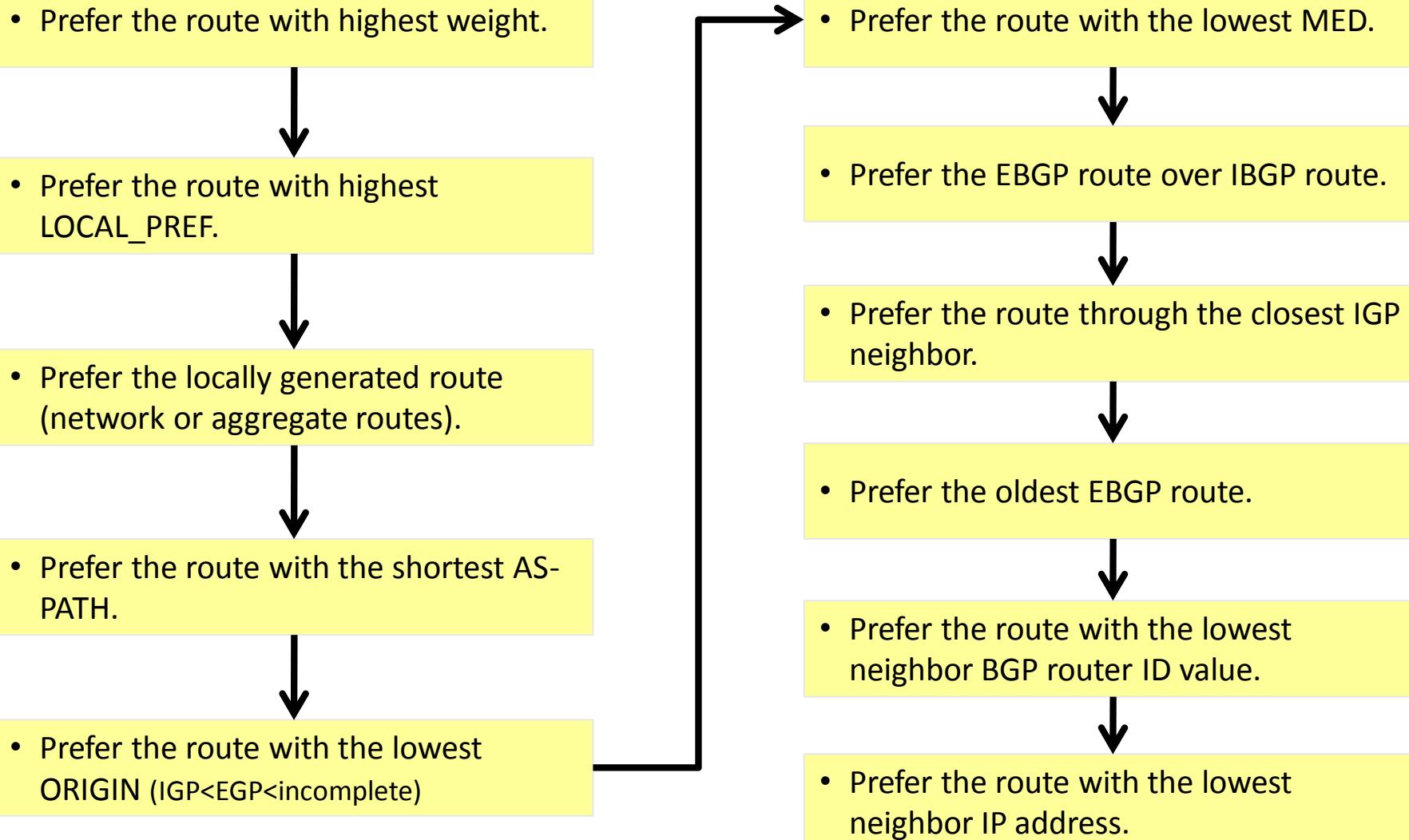
- This first instance of the setweightin route map assigns 2000 to any route update from AS 100, and the second instance of the setweightin route map assigns 1000 to route updates from any other AS



BGP Route Selection Process

- The BGP best path decision is based on the value of attributes that the update contains and other BGP-configurable factors.
- BGP considers only synchronized routes with no AS loops and a valid next-hop address.

BGP Route Selection Process



BGP Configuration

Implementing Basic BGP

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

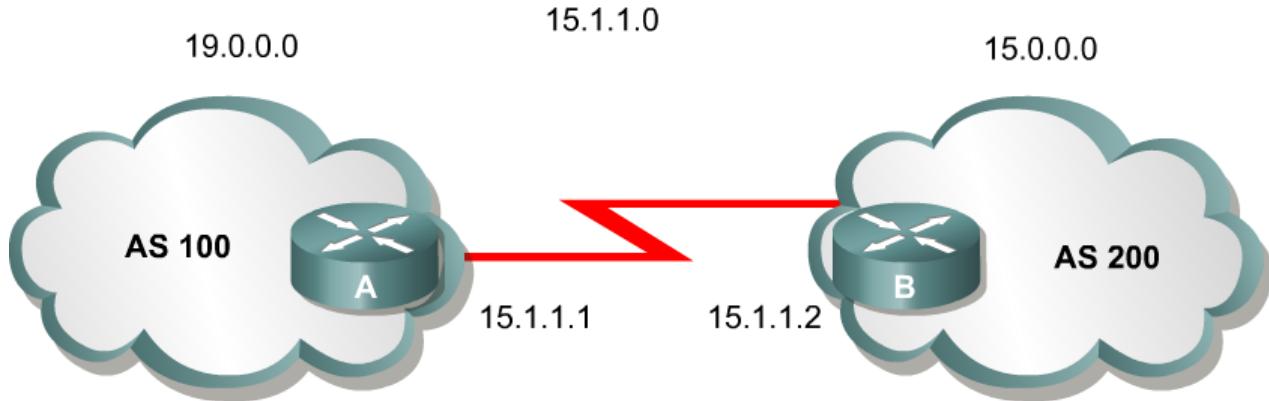
Verifying BGP

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

Documenting

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

BGP Configuration



Configuration for A

```
router bgp 100  
network 19.0.0.0  
neighbor 15.1.1.2 remote-as 200
```

Configuration for B

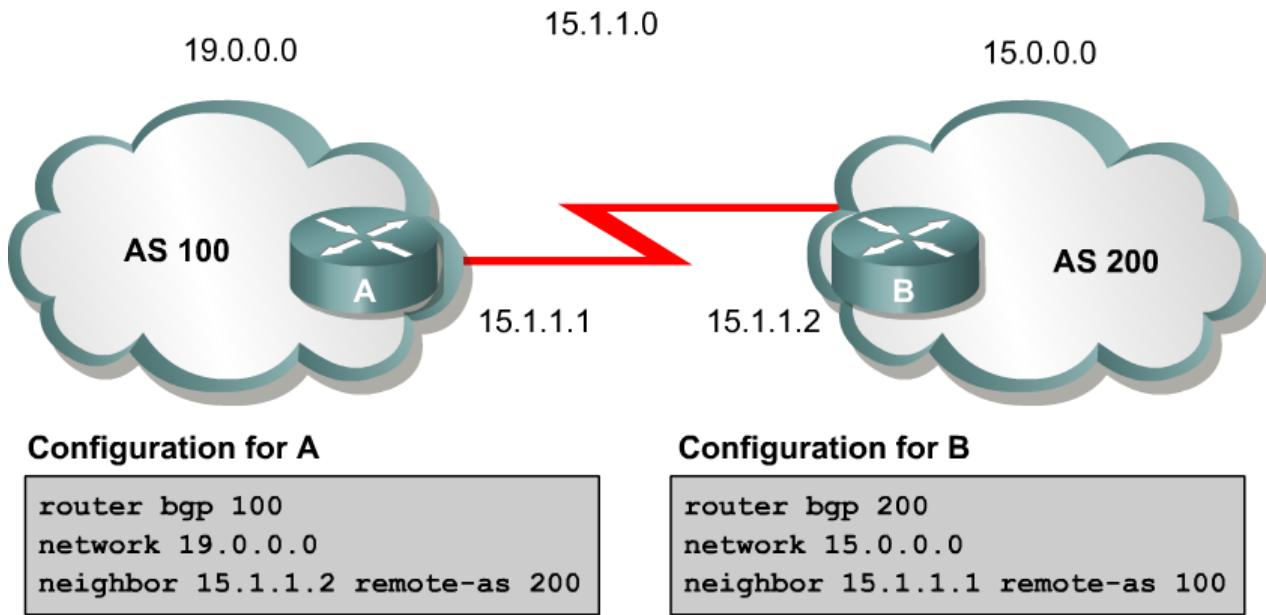
```
router bgp 200  
network 15.0.0.0  
neighbor 15.1.1.1 remote-as 100
```

- To begin configuring a BGP process, issue the following familiar command:

```
Router (config) #router bgp AS-number
```

- BGP configuration commands appear on the surface to mirror the syntax of familiar IGP (for example, RIP, OSPF) commands.
- Although the syntax is similar, the function of these commands is significantly different.
- Note:** Cisco IOS permits only one BGP process to run at a time, thus, **a router cannot belong to more than one AS**.

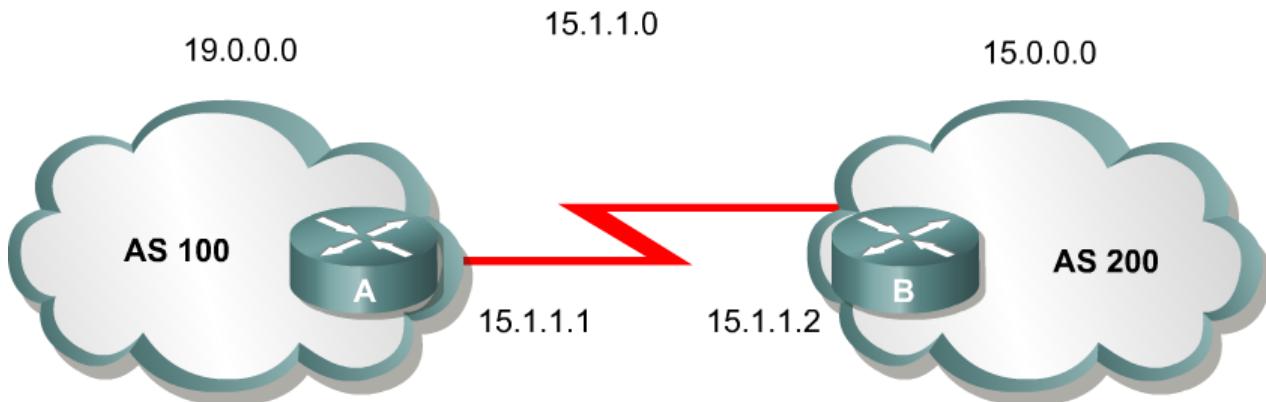
BGP Configuration



Router(config-router) #**network network-number [mask network-mask]**

- The **network** command is used with **IGPs**, such as RIP, to determine the interfaces on which to send and receive updates, as well as which directly connected networks to advertise.
- However, when configuring **BGP**, the **network** command does not affect what interfaces BGP runs on.
- In BGP, the **network** command tells the BGP process **what locally learned networks to advertise**.
- The networks can be **connected routes, static routes, or routes learned via a dynamic routing protocol**, such as RIP.
 - Thus, configuring just a **network** statement will not establish a BGP neighbor relationship. This is a major difference between BGP and IGPs.

BGP Configuration



Configuration for A

```
router bgp 100
network 19.0.0.0
neighbor 15.1.1.2 remote-as 200
```

Configuration for B

```
router bgp 200
network 15.0.0.0
neighbor 15.1.1.1 remote-as 100
```

network command continued...

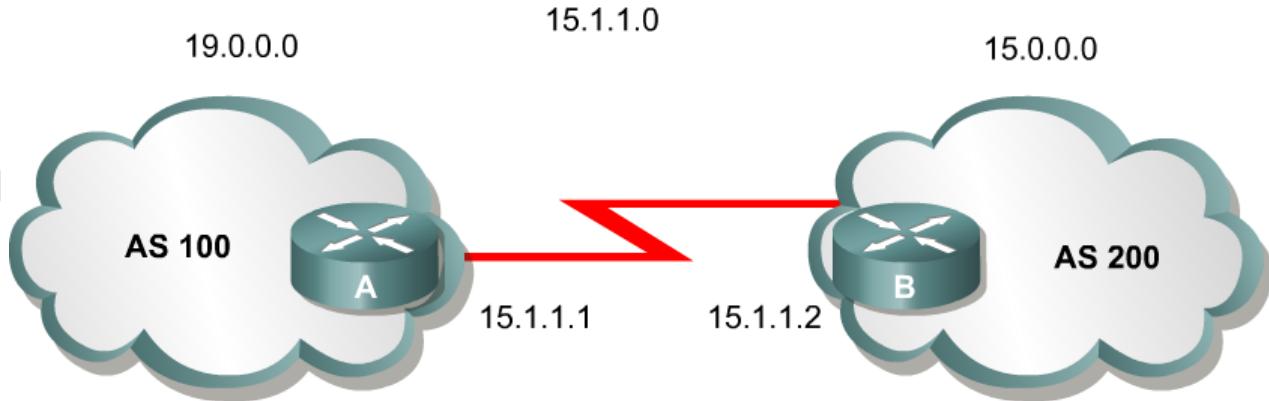
- *These networks must also exist in the local router's routing table (show ip route), or they will not be sent out in updates.*
- You can use the **mask** keyword with the **network** command to specify individual subnets.
- The **mask** parameter indicates that BGP-4 supports subnetting and supernetting.
 - If the mask is not specified, this command announces only the classful network
- Routes learned by the BGP process are propagated by default, but are often filtered by a routing policy.

BGP Route Must Be in IP Routing Table

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

```
network 192.168.0.0 mask 255.255.0.0
ip route 192.168.0.0 255.255.0.0 null0
```
- Now BGP can find an exact match in the routing table and announce the 192.168.0.0/16 network to its neighbors.
 - The advertised static route would never actually be used since BGP would contain longer prefix matching routes in its routing table.

BGP Configuration



Configuration for A

```
router bgp 100
network 19.0.0.0
neighbor 15.1.1.2 remote-as 200
```

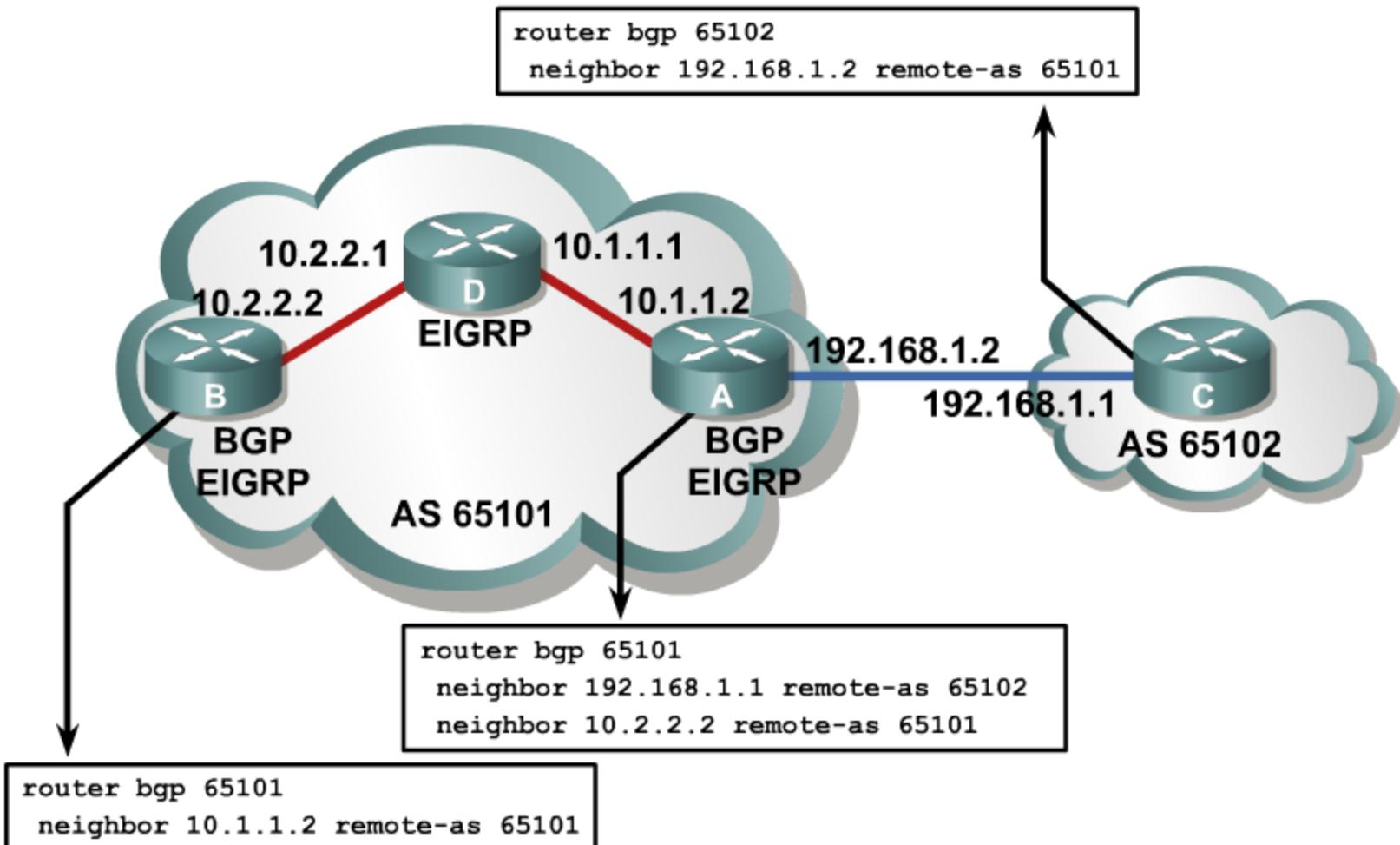
Configuration for B

```
router bgp 200
network 15.0.0.0
neighbor 15.1.1.1 remote-as 100
```

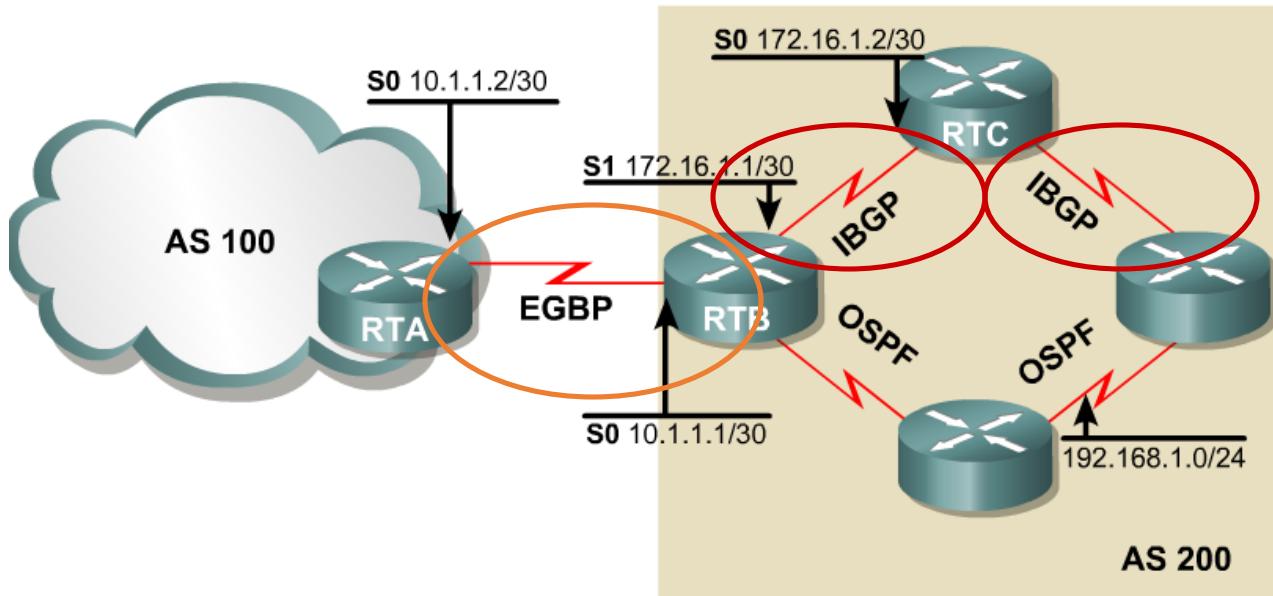
Router (config-router) #**neighbor ip-address remote-as AS-number**

- In order for a BGP router to **establish a neighbor relationship with another BGP router**, you must issue the this configuration command.
- This command serves to identify a peer router with which the local router will establish a session.
- The **AS-number** argument determines whether the neighbor router is an EBGP or an IBGP neighbor.

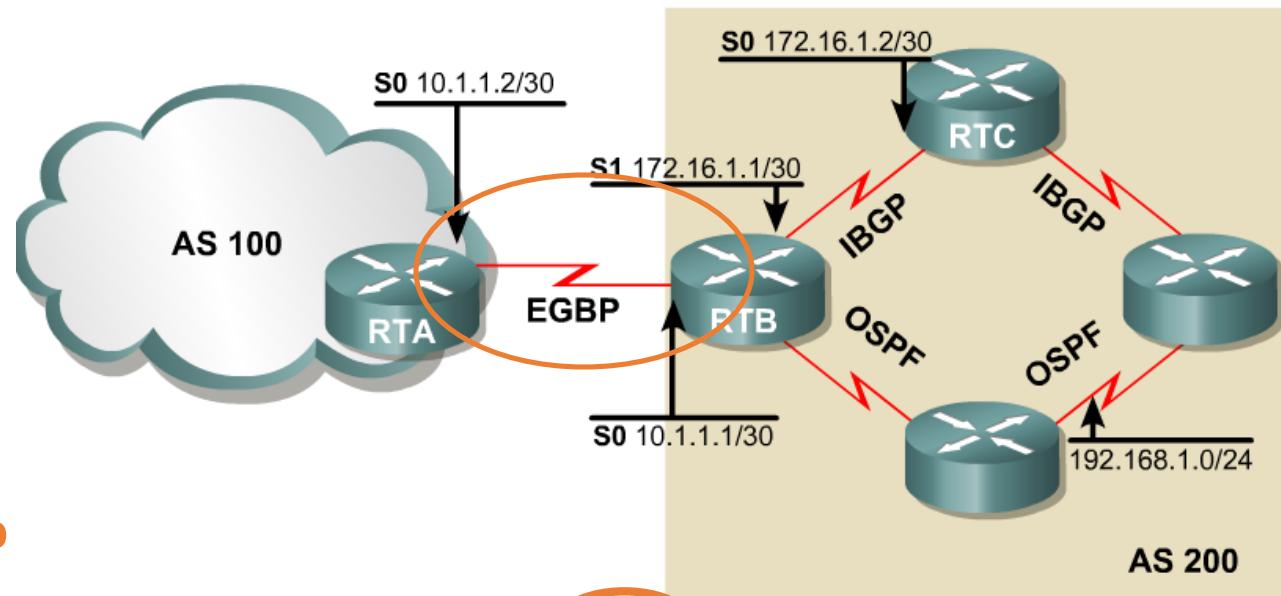
BGP Configuration



BGP Configuration



- If the **AS-number** configured in the **router bgp** command is **identical** to the **AS-number** configured in the **neighbor** statement, BGP will initiate an **internal session - IBGP**.
- If the field values are **different**, BGP will build an **external session - EBGP**.



EBGP

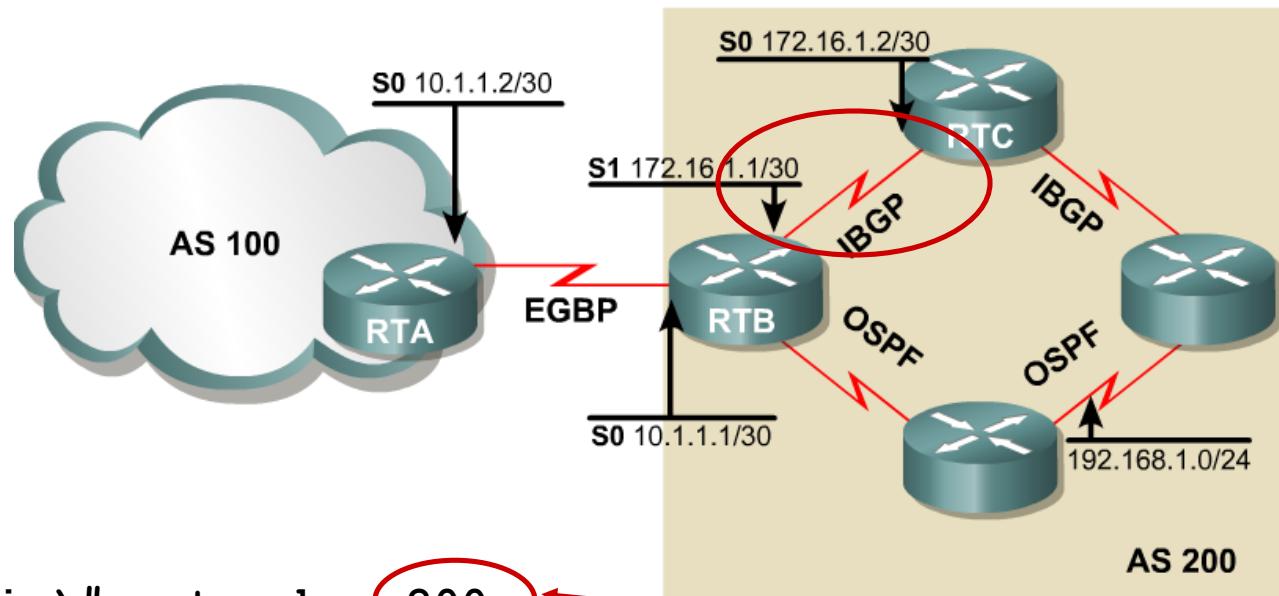
```
RTA(config) #router bgp 100
```

```
RTA(config-router) #neighbor 10.1.1.1 remote-as 200
```

```
RTB(config) #router bgp 200
```

```
RTB(config-router) #neighbor 10.1.1.2 remote-as 100
```

- RTB: Note that the **neighbor** command's **remote-as** value, 100, is different from the AS number specified by the **router bgp** command (200).
- Because the two AS numbers are different, BGP will start an **EBGP** connection with RTA.
- Communication will occur between autonomous systems.



IBGP

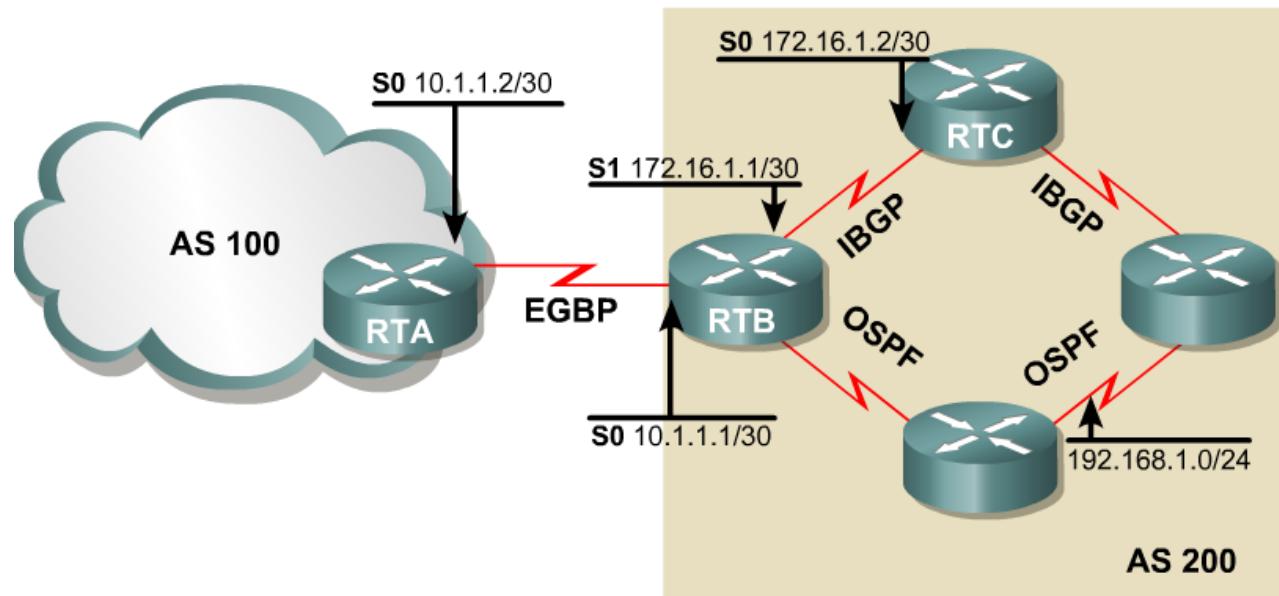
```

RTB (config) #router bgp 200
RTB (config-router) #neighbor 172.16.1.2 remote-as 200
RTB (config-router) #neighbor 172.16.1.2 update-source loopback 0

RTC (config) #router bgp 200
RTC (config-router) #neighbor 172.16.1.1 remote-as 200
RTC (config-router) #neighbor 172.16.1.1 update-source loopback 0

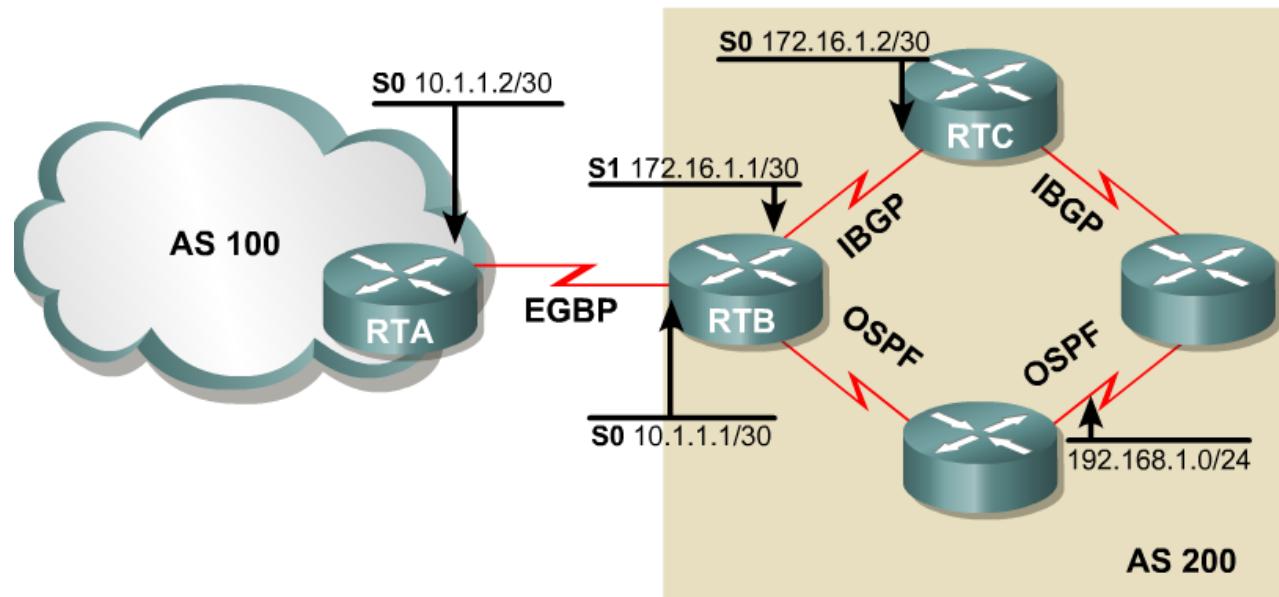
```

- Since the **remote-as** value (200) is the same as RTB's BGP AS number, BGP recognizes that this connection will occur within AS 200, so it attempts to establish an **IBGP** session.
- In reality, AS 200 is not a remote AS at all; it is the local AS, since both routers live there. But for simplicity, the keyword **remote-as** is used when configuring both EBGP and IBGP sessions.

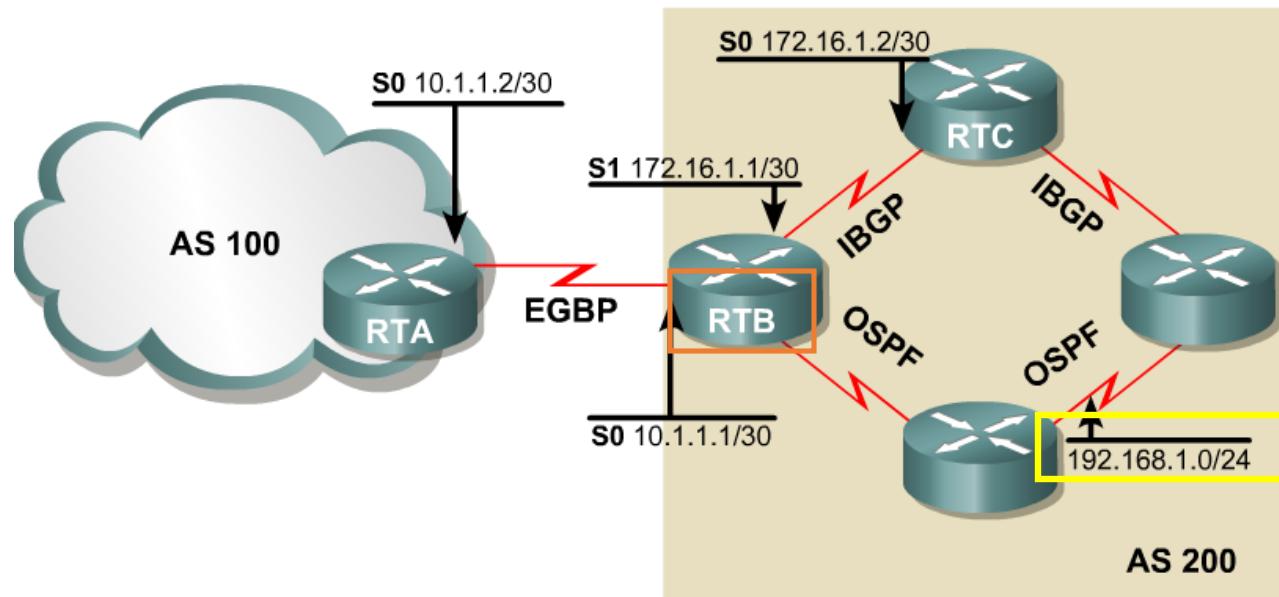


```
RTB(config-router)#neighbor 172.16.1.2 update-source loopback 0
RTC(config-router)#neighbor 172.16.1.1 update-source loopback 0
```

- The **update-source loopback 0** command is used to instruct the router to use *any* operational interface for TCP connections (as long as Lo0 is up and configured with an IP address).
- Without the **update-source loopback 0** command, BGP routers can use only the closest IP interface to the peer.
- The ability to use any operational interface provides BGP with robustness in the event the link to the closest interface fails.
 - Since EBGP sessions are typically point-to-point, there is no need to use this command with EBGP.



- Assume the following route appears in RTB's table:
- ```
0 192.168.1.0/24 [110/74] via 10.2.2.1, 00:31:34,
Serial2
```
- RTB learned this route via an IGP, in this case, OSPF.
  - This AS uses OSPF internally to exchange route information.
  - Can RTB advertise this network via BGP?
  - Certainly, redistributing OSPF into BGP will do the trick, but the BGP **network** command will do the same thing.



```

RTB(config) #router bgp 200
RTB(config-router) #network 172.16.1.0 mask 255.255.255.254
RTB(config-router) #network 10.1.1.0 mask 255.255.255.254
RTB(config-router) #network 192.168.1.0

```

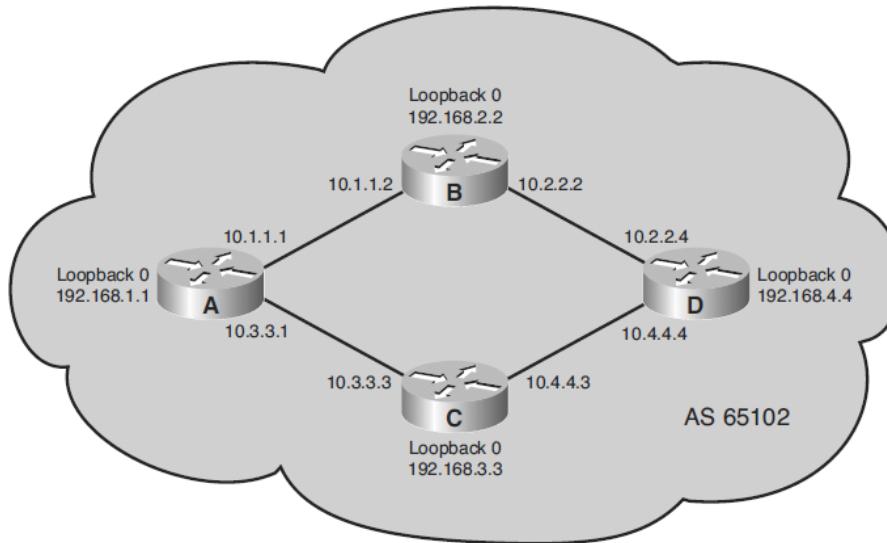
- The first two **network** commands include the **mask** keyword, so that only a particular subnet is specified.
- The third **network** command results in the OSPF route being advertised by BGP *without* redistribution.
- Remember that the BGP **network** command works differently than the IGP **network** command!

# IBGP Source IP Address Problem

## **update-source**

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

# IBGP Source IP Address Problem



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

# IBGP Source IP Address Solution

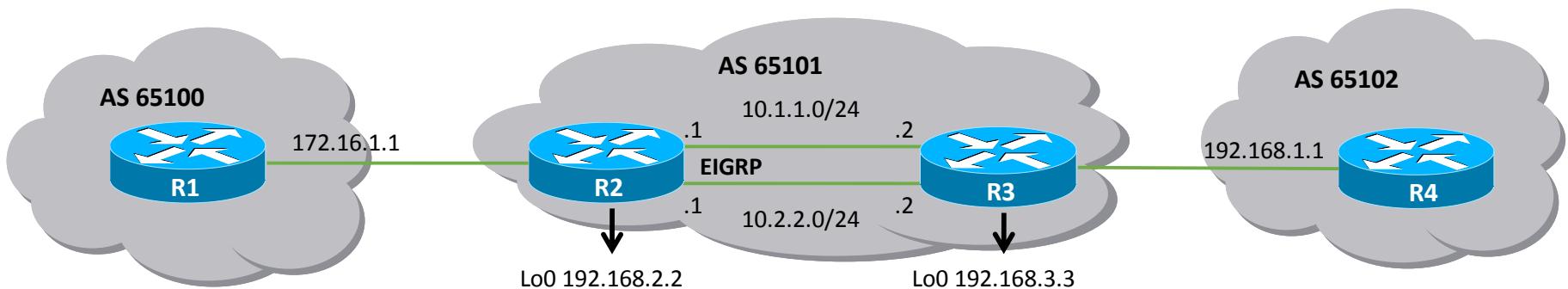
- Establish the IBGP session using a loopback interface.

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

```
neighbor {ip-address | peer-group-name} update-source
loopback interface-number
```

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

# IBGP Source IP Address Example



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

```

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

```

# EBGP Dual-Homed Problem

## **update-source ebgp-multipoint**



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

# EBGP Dual-Homed Solution



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

# Enable Multihop EBGP

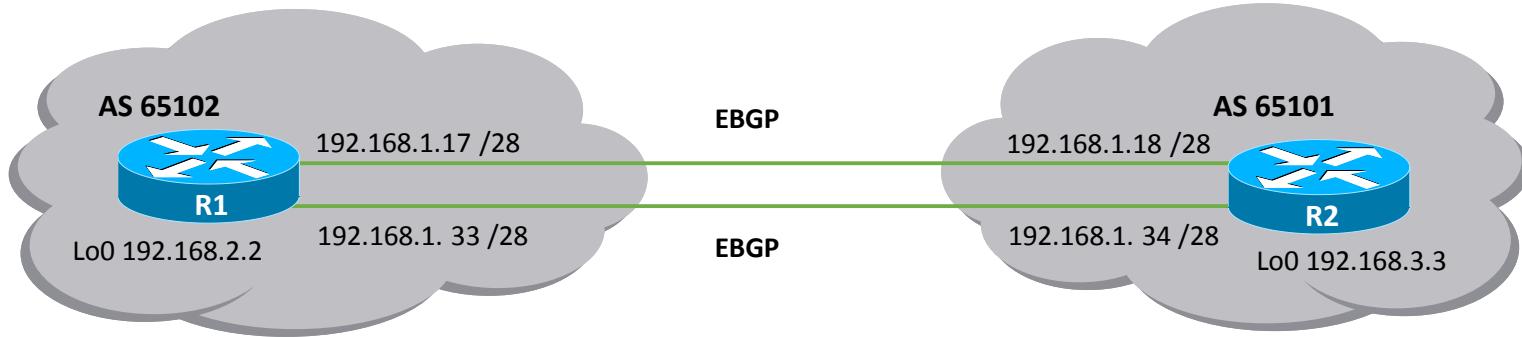
- Increase the time-to-live (TTL) for EBGP connections.

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

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

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

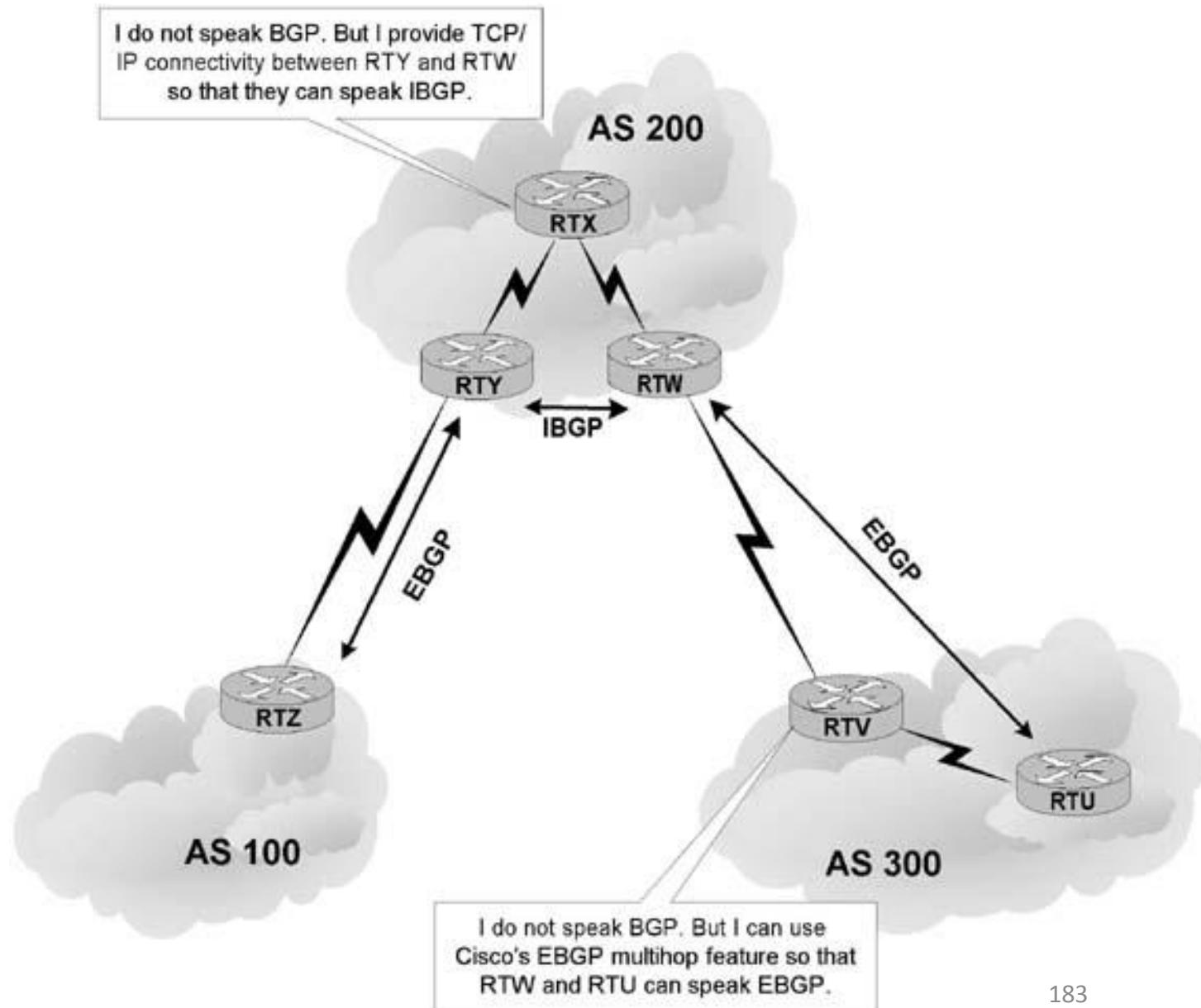
# Multihop EBGP Example



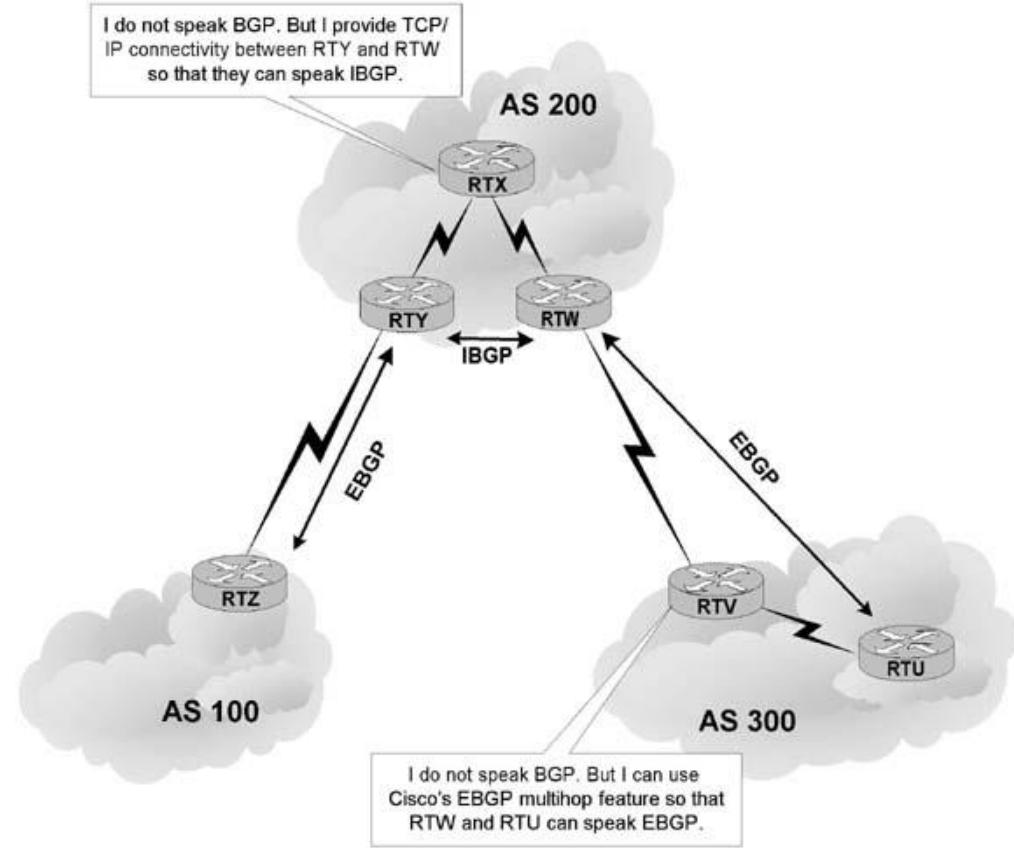
```
R1(config)# router bgp 65102
R1(config-router)# neighbor 172.16.1.1 remote-as 65101
R1(config-router)# neighbor 172.16.1.1 update-source loopback0
R1(config-router)# neighbor 172.16.1.1 ebgp-multihop 2
R1(config-router)# exit
R1(config)# ip route 172.16.1.1 255.255.255.255 192.168.1.18
R1(config)#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

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

# EBGP vs IBGP (Multihop)

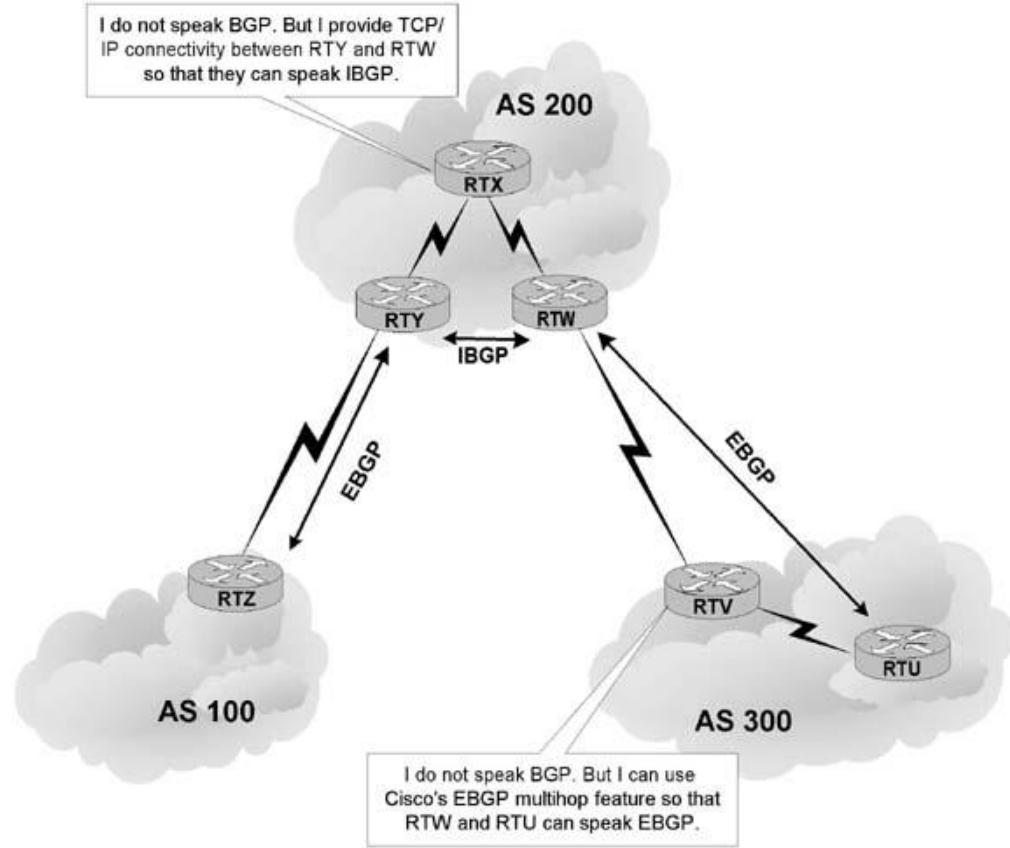


# EBGP vs IBGP (Multihop)



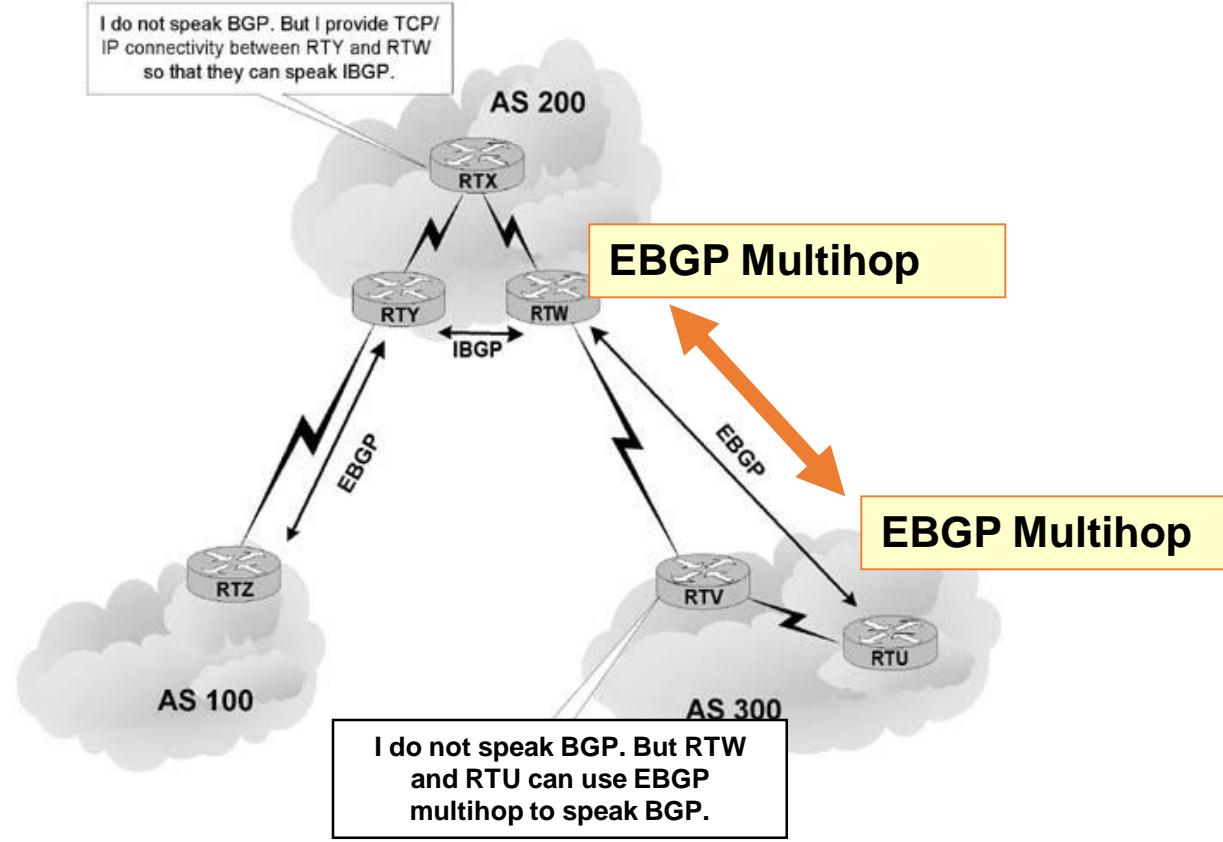
- **EBGP** peers must be **directly connected**, but there are certain exceptions to this requirement.
- In contrast, **IBGP** peers merely **require TCP/IP connectivity** within the same AS.
  - As long as RTY can communicate with RTW using TCP, both routers can establish an IBGP session.
  - If needed, an IGP such as OSPF can provide IBGP peers with routes to each other.

# IBGP (Multihop)



- In a typical configuration, an **IBGP** router maintains **IBGP** sessions with all other **IBGP** routers in the AS, forming a logical full-mesh.
  - This is necessary because IBGP routers do not advertise routes learned via IBGP to other IBGP peers (to prevent routing loops).
  - In other words, if you want your IBGP routers to exchange BGP routes with each other, you should configure a full-mesh.
  - An alternative to this approach: configuring a route reflector (later)

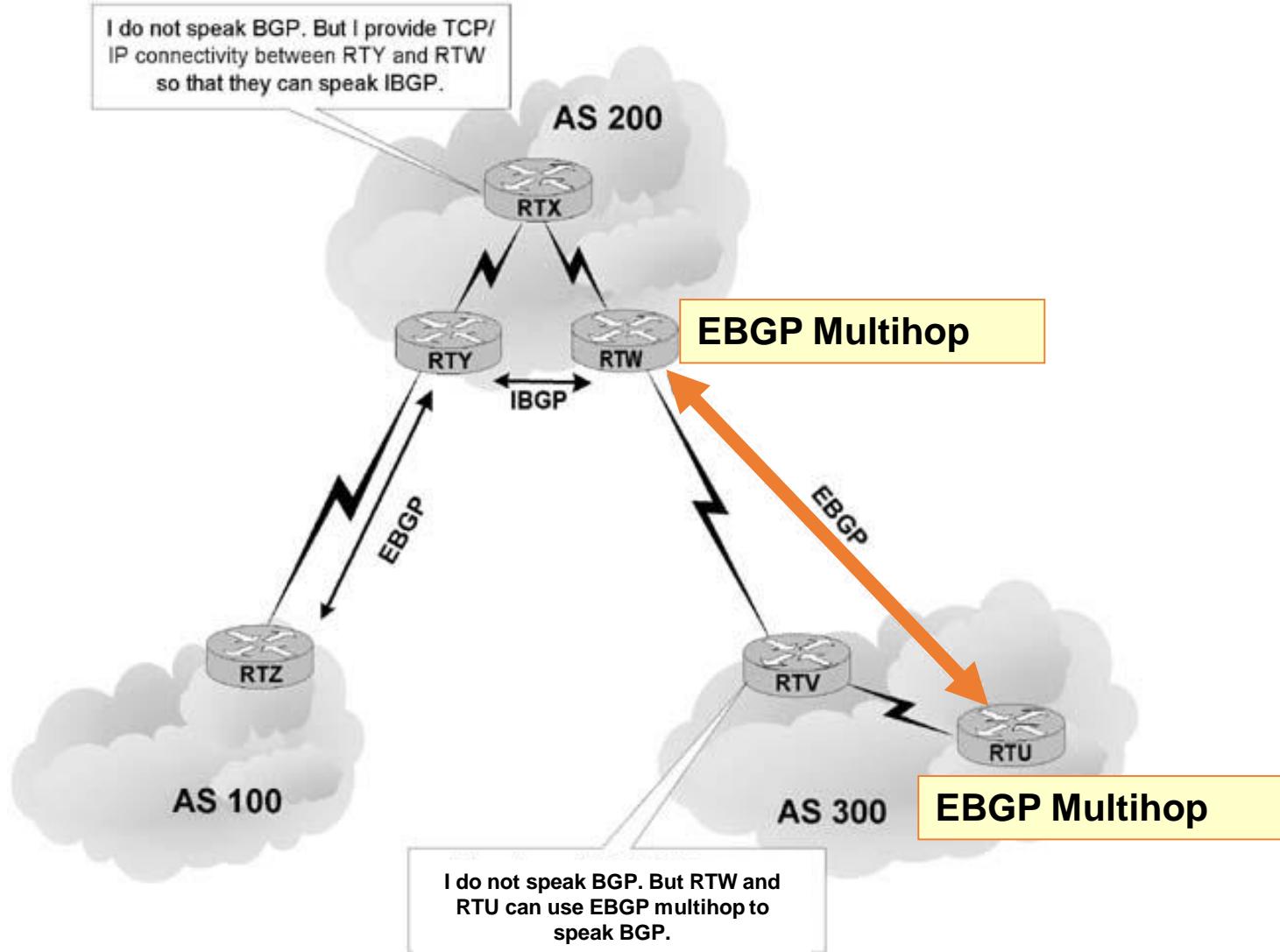
# EBGP (Multihop)



- **EBGP** neighbors must be directly connected in order to establish an **EBGP** session.
- However, **EBGP multihop** is a Cisco IOS option allows RTW and RTU to be logically connected in an **EBGP** session, despite the fact that RTV does not support BGP.
- The **EBGP** multihop option is configured on each peer with the following command:

```
Router (config-router) #neighbor IP-address ebgp-multihop [hops]
```

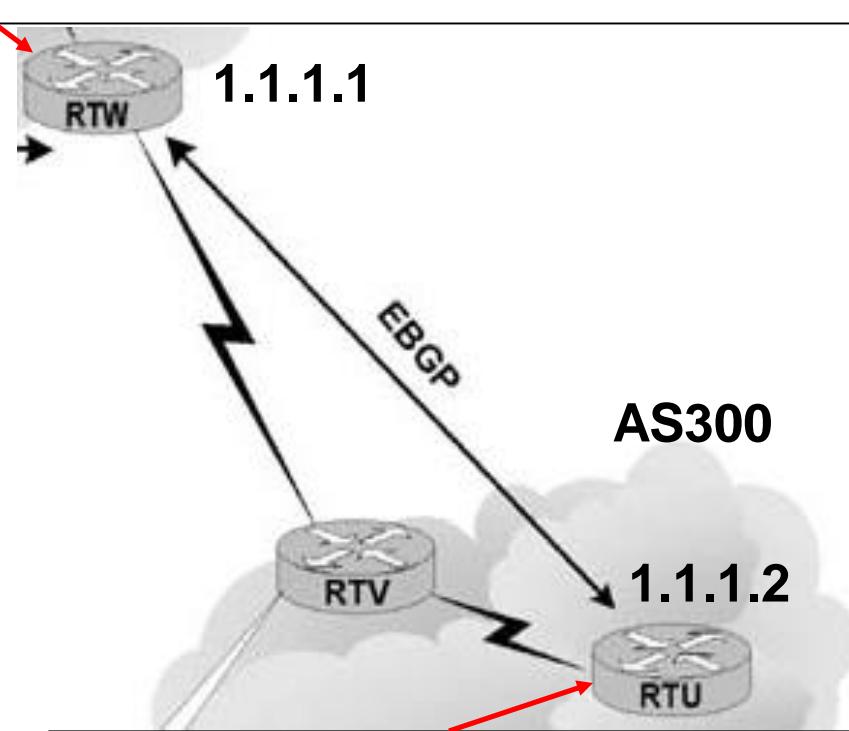
# EBGP (Multihop)



# EBGP (Multihop)

```
RTW(config)#router bgp 200
RTW(config-router)#neighbor 1.1.1.2 remote-as 300
RTW(config-router)#neighbor 1.1.1.2 ebgp-multihop 2
```

AS200



```
RTU(config)#router bgp 300
RTU(config-router)#neighbor 1.1.1.1 remote-as 200
RTU(config-router)#neighbor 1.1.1.1 ebgp-multihop 2
```

# Advertising EBGP Routes to IBGP Peers

## **next-hop-self**

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

# neighbor next-hop-self Command

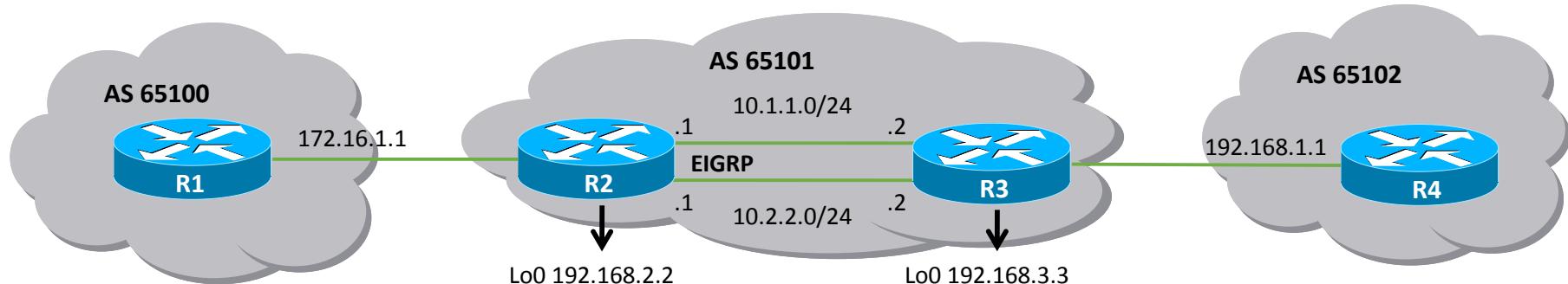
- Configure the router as the next hop for a BGP-speaking peer.

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

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

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

# Next Hop Self Example



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

# BGP Authentication

- BGP supports message digest 5 (MD5) neighbor authentication.
  - MD5 sends a “message digest” (also called a “hash”), which is created using the key and a message.
  - The message digest is then sent instead of the key.
  - The key itself is not sent, preventing it from being read by someone eavesdropping on the line while it is being transmitted.
- To enable MD5 authentication on a TCP connection between two BGP peers, use the router configuration command:

```
neighbor {ip-address | peer-group-name}
password string
```

# Enable MD5 authentication

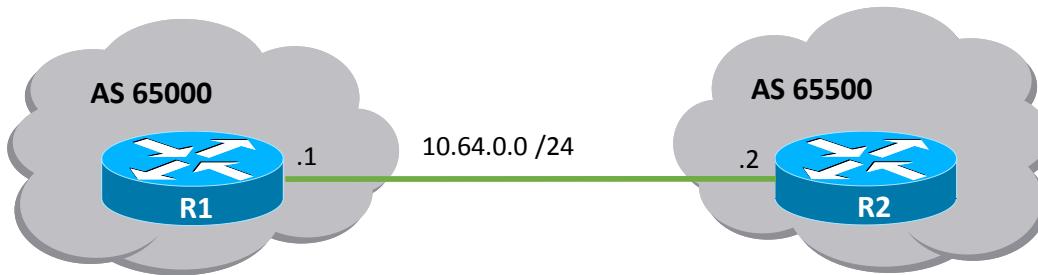
- Enable MD5 authentication between two BGP peers.

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

```
neighbor {ip-address | peer-group-name} password string
```

- This is the only command required to enable MD5 authentication.
- The *string* value is:
  - Case-sensitive password of up to 25 characters.
  - The first character cannot be a number.
  - The string can contain any alphanumeric characters, including spaces.
  - You cannot specify a password in the format number-space-anything.
  - The space after the number can cause authentication to fail.

# Configuring MD5 Authentication



```
R1(config)# router bgp 65000
R1(config-router)# neighbor 10.64.0.2 remote-as 65500
R1(config-router)# neighbor 10.64.0.2 password BGP-Pa55w0rd
R1(config-router)#

```

```
R2(config)# router bgp 65500
R2(config-router)# neighbor 10.64.0.1 remote-as 65000
R2(config-router)# neighbor 10.64.0.1 password BGP-Pa55w0rd
R2(config-router)#

```

# MD5 Configuration Problems

- If a router has a password configured for a neighbor, but the neighbor router does not have a password configured, the following message will appear on the console screen:

```
%TCP-6-BADAUTH: No MD5 digest from
10.1.0.2(179) to 10.1.0.1(20236)
```

- Similarly, if the two routers have different passwords configured, the following will appear:

```
%TCP-6-BADAUTH: Invalid MD5 digest from
10.1.0.1(12293) to 10.1.0.2(179)
```

# Shut Down a BGP Neighbor

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

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

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

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

# Clearing the BGP Session

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

# Hard Reset of BGP Sessions

- Reset all BGP connections with this router.

Router#

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

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

**Use this command with CAUTION, better yet, not at all, in a production network. From the net...**

“`clear ip bgp *` OOPS! Not me but a colleague who was an employee of a large ISP with a 3 letter title. Got back from a Cisco routing course and thought they would try out some commands on the core network. It took 45 minutes for the core to reconverge. P45 followed”

# Soft Reset Outbound

- Resets all BGP connections without loss of routes.

Router#

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

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

# Soft Reset Inbound: Method #1

- Two commands are required.

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

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

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

```
Router#
```

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

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

# Soft Reset Inbound: Method #2

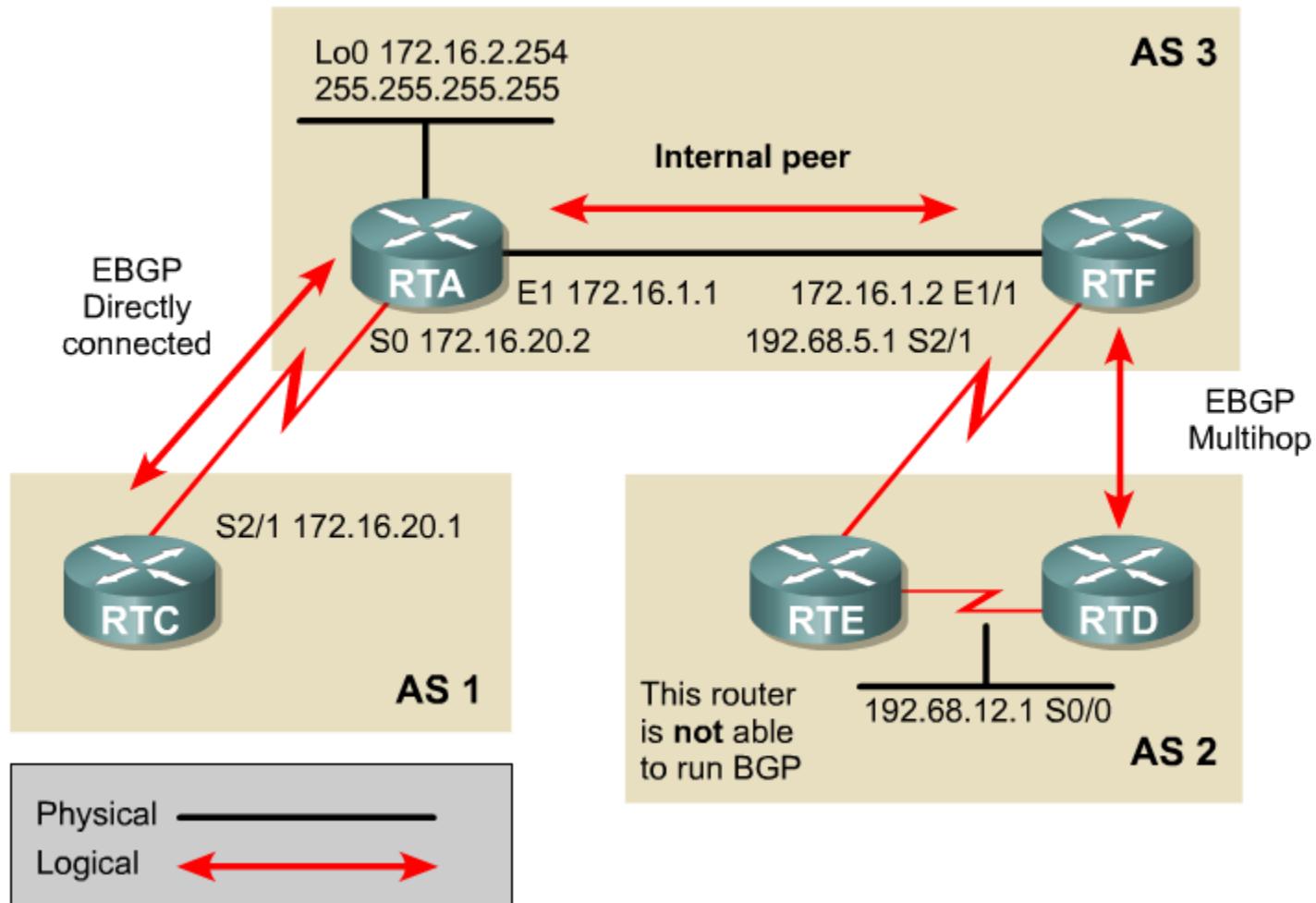
- Also called route refresh.

Router#

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

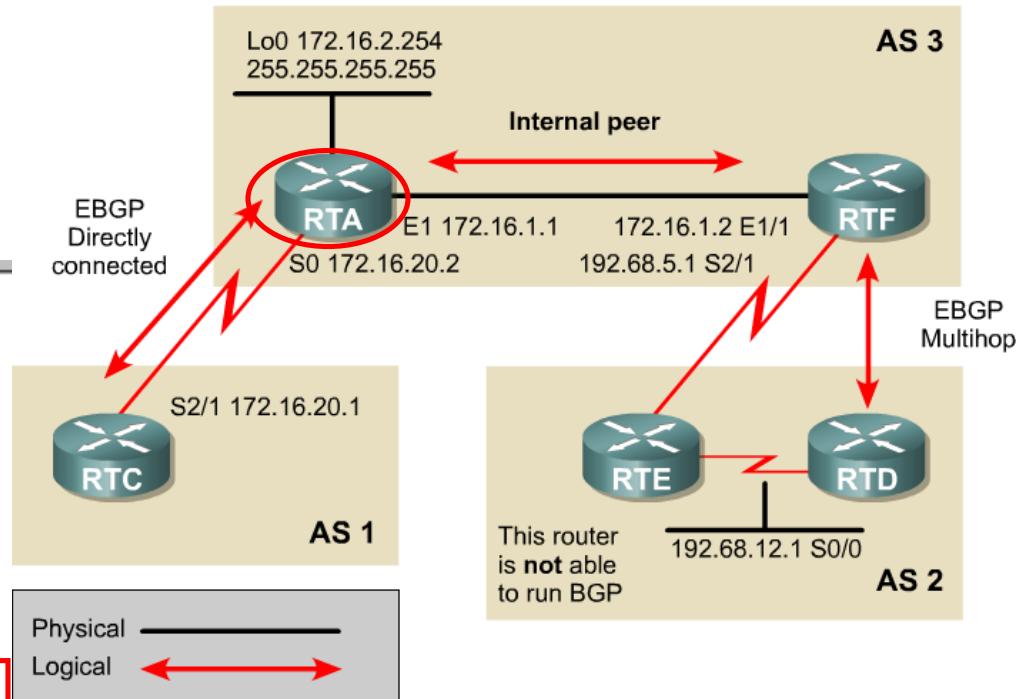
- This dynamically soft resets inbound updates.
- Unlike method #1, this method requires no preconfiguration and requires significantly less memory.

# BGP Configuration Example #0



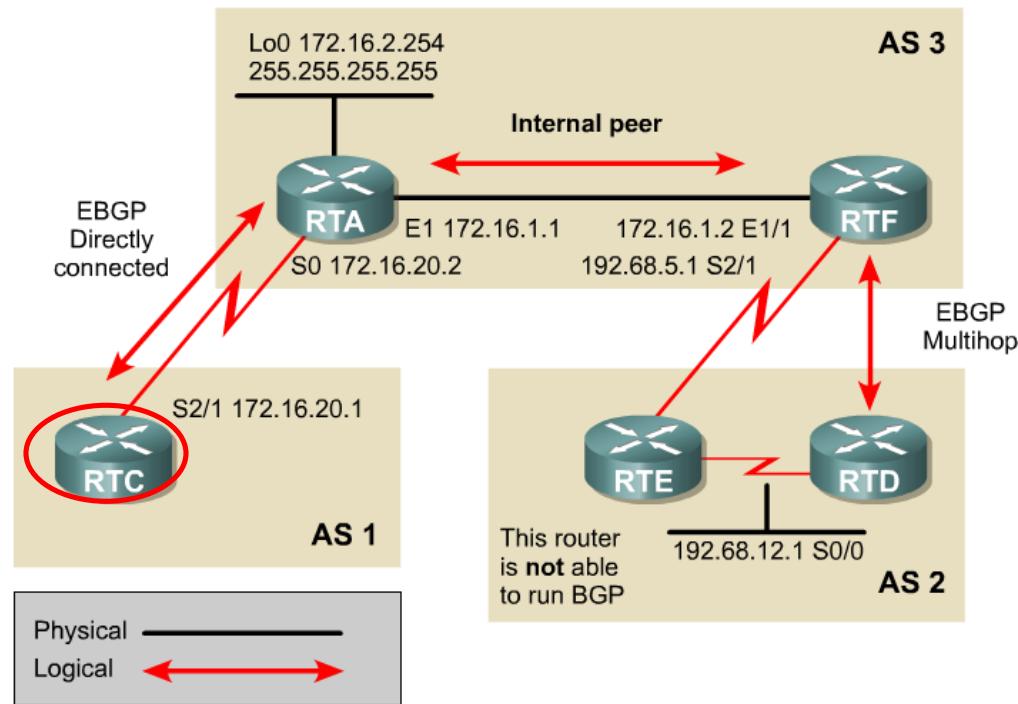
# BGP Configuration Example #0

```
RTA#show running-config
ip subnet-zero
interface Loopback0
ip address 172.16.2.254 255.255.255.255
interface Ethernet1
ip address 172.16.1.1 255.255.255.0
interface Serial0
ip address 172.16.20.2 255.255.255.0
router ospf 10
network 172.16.0.0 0.0.255.255 area 0
router bgp 3
no synchronization
neighbor 172.16.1.2 remote-as 3
neighbor 172.16.1.2 update-source Loopback0
neighbor 172.16.20.1 remote-as 1
no auto-summary
ip classless
RTA#
```



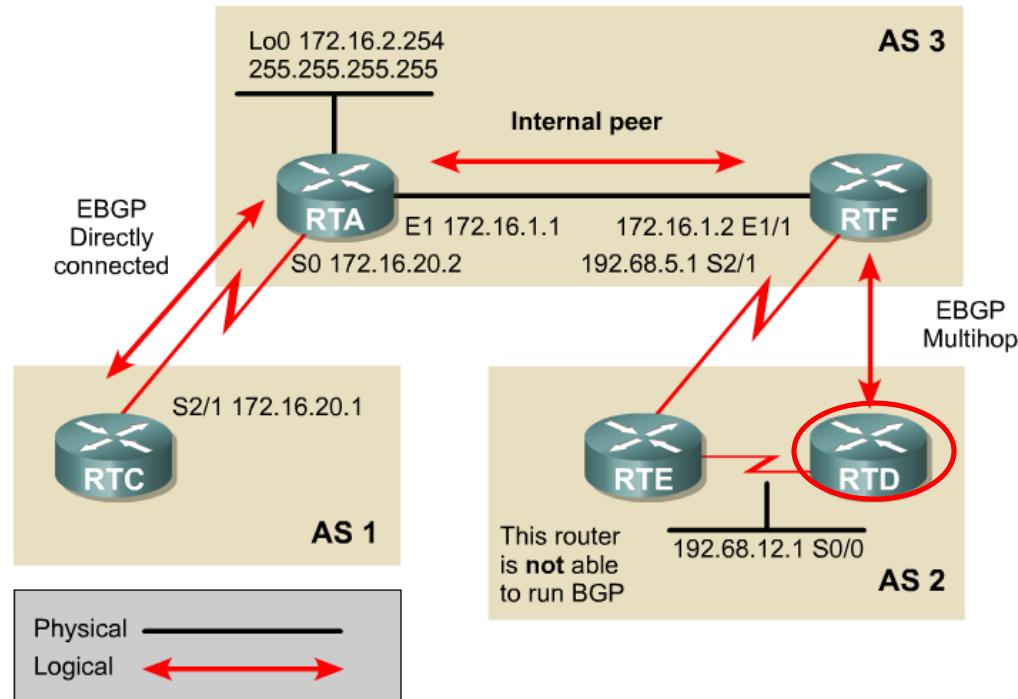
# BGP Configuration Example #0

```
RTC#show running-config
ip subnet-zero
interface Serial2/1
ip address 172.16.20.1 255.255.255.0
router bgp 1
neighbor 172.16.20.2 remote-as 3
no auto-summary
ip classless
RTC#
```



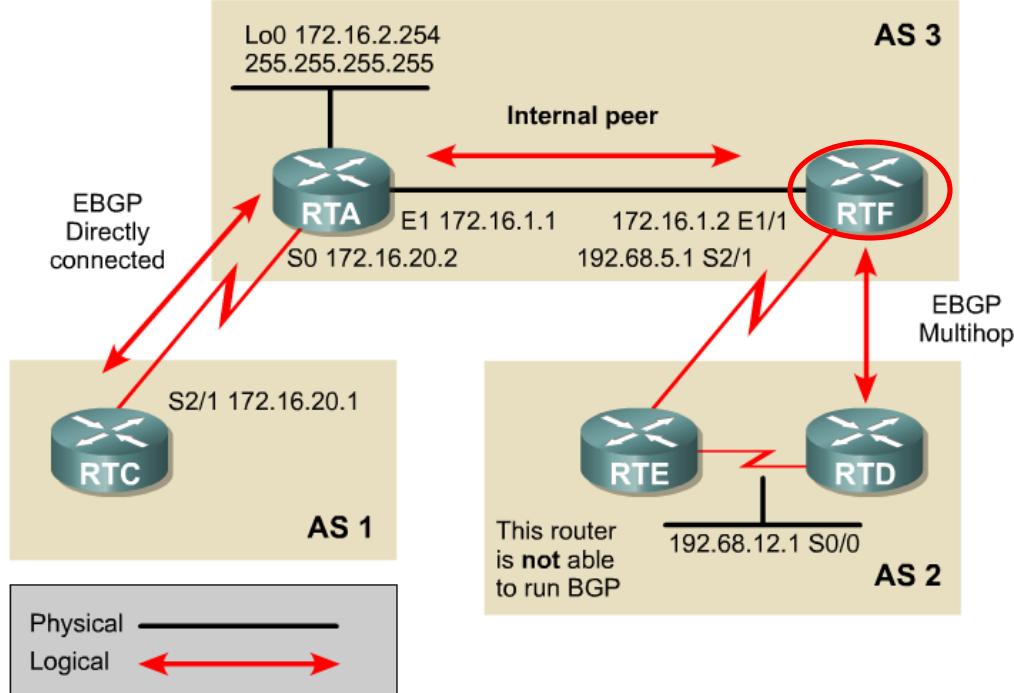
# BGP Configuration Example #0

```
RTD#show running-config
ip subnet-zero
interface Serial0/0
ip address 192.68.12.1 255.255.255.0
router ospf 10
network 192.68.0.0 0.0.255.255 area 0
router bgp 2
neighbor 192.68.5.1 remote-as 3
neighbor 192.68.5.1 ebgp-multihop 2
no auto-summary
ip classless
RTD#
```



```
RTF#show running-config
```

```
ip subnet-zero
interface Ethernet1/1
ip address 172.16.1.2 255.255.255.0
interface Serial2/1
ip address 192.68.5.1 255.255.255.0
router ospf 10
network 172.16.0.0 0.0.255.255 area 0
network 192.68.0.0 0.0.255.255 area 0
router bgp 3
no synchronization
neighbor 172.16.2.254 remote-as 3
neighbor 192.68.12.1 remote-as 2
neighbor 192.68.12.1 ebgp-multihop 2
no auto-summary
ip classless
RTF#
```



```
RTF#show ip bgp neighbor
```

```
BGP neighbor is 172.16.2.254, remote AS 3, internal link
BGP version 4, remote router ID 172.16.2.254
BGP state = Established, table version = 2, up for 22:36:09
Last read 00:00:10, hold time is 180, keepalive interval is 60 seconds
Minimum time between advertisement runs is 5 seconds
Received 1362 messages, 0 notifications, 0 in queue
Sent 1362 messages, 0 notifications, 0 in queue
Connections established 2; dropped 1
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 172.16.1.2, Local port: 11008
Foreign host: 172.16.2.254, Foreign port: 179
BGP neighbor is 192.68.12.1, remote AS 2, external link
BGP version 4, remote router ID 192.68.5.2
BGP state = Established, table version = 2, up for 22:13:01
Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
Minimum time between advertisement runs is 30 seconds
Received 1336 messages, 0 notifications, 0 in queue
```

# Verifying BGP Configuration

- If the router has not installed the BGP routes you expect, you can use the **show ip bgp** command to verify that BGP has learned these routes.

```
RTA#show ip bgp
```

```
BGP table version is 3, local router ID is 10.2.2.2
```

```
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

| Network            | Next Hop    | Metric | LocPrf | Weight    | Path |
|--------------------|-------------|--------|--------|-----------|------|
| * i1.0.0.0         | 192.168.1.6 | 0      | 100    | 0 200 400 | e    |
| *>i10.1.1.1/32     | 10.1.1.1    | 0      | 100    | 0         | i    |
| *>i172.16.1.0/24   | 10.1.1.1    | 0      | 100    | 0         | i    |
| * i192.168.1.32/27 | 192.168.1.6 | 0      | 100    | 0 200     | i    |

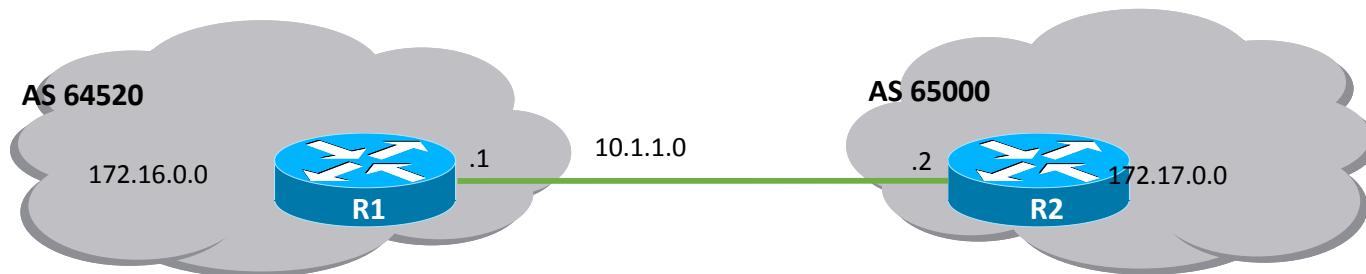
# Verifying BGP Configuration

- If an expected BGP route does not appear in the BGP table, you can use the **show ip bgp neighbors** command to verify that your router has established a BGP connection with its neighbors.

```
RTA#show ip bgp neighbors
```

```
BGP neighbor is 172.24.1.18, remote AS 200, external link
 BGP version 4, remote router ID 172.16.1.1
 BGP state = Established, up for 00:03:25
 Last read 00:00:25, hold time is 180, keepalive interval is 60 seconds
 Neighbor capabilities:
 Route refresh: advertised and received
 Address family IPv4 Unicast: advertised and received
 Received 7 messages, 0 notifications, 0 in queue
 Sent 8 messages, 0 notifications, 0 in queue
 Route refresh request: received 0, sent 0
 Minimum time between advertisement runs is 30 seconds
<output omitted>
```

# BGP Configuration Example #1



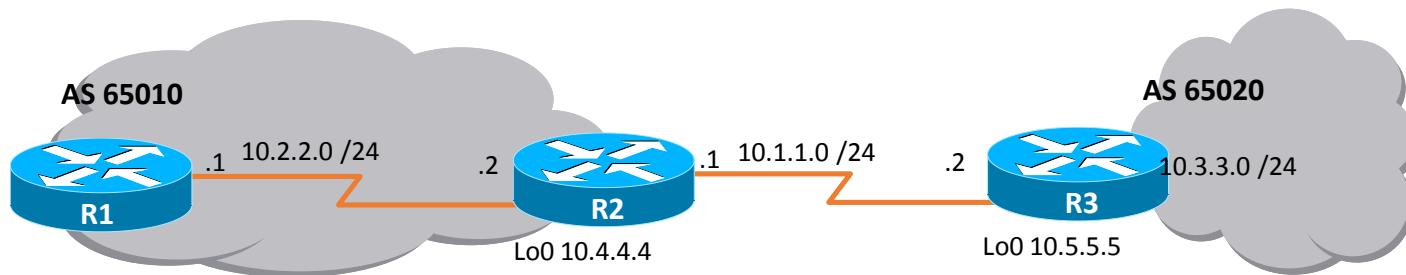
```
R1(config)# router bgp 64520
R1(config-router)# neighbor 10.1.1.2 remote-as 65000
R1(config-router)# network 172.16.0.0
R1(config-router)#

```

```
R2(config)# router bgp 65500
R2(config-router)# neighbor 10.1.1.1 remote-as 64520
R2(config-router)# network 172.17.0.0
R2(config-router)#

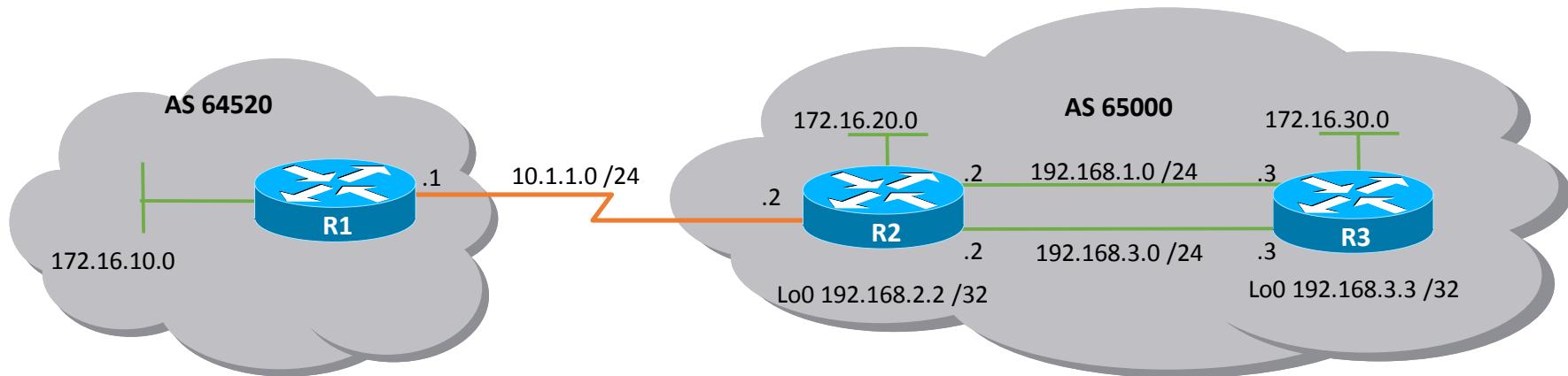
```

# BGP Configuration Example #2



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

# IBGP and EBGP Example



```
R2(config)# router bgp 65000
R2(config-router)# neighbor 10.1.1.1 remote-as 64520
R2(config-router)# neighbor 192.168.3.3 remote-as 65000
R2(config-router)# neighbor 192.168.3.3 update-source loopback 0
R2(config-router)# neighbor 192.168.3.3 next-hop-self
R2(config-router)# network 172.16.20.0 mask 255.255.255.0
R2(config-router)# network 192.168.1.0
R2(config-router)# network 192.168.3.0
R2(config-router)# no synchronization
R2(config-router)#

```

# BGP Peer Groups

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

# Defining a BGP Peer Group

- Create a peer group on the local router.

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

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

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

# Assign Neighbors to the Peer Group

- Assign neighbors as part of the peer group.

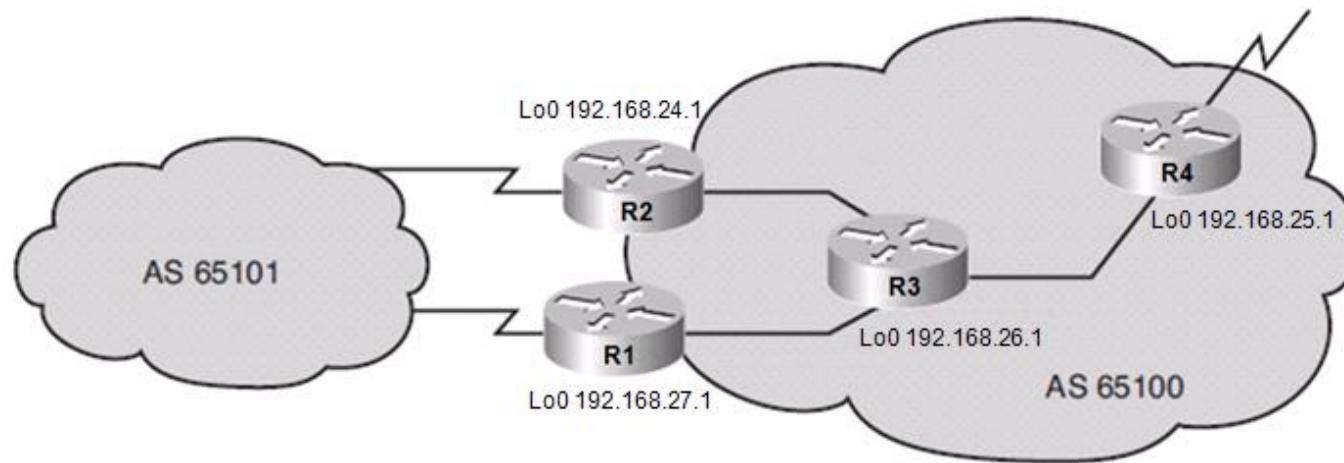
```
Router(config-router) #
```

```
neighbor ip-address peer-group peer-group-name
```

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

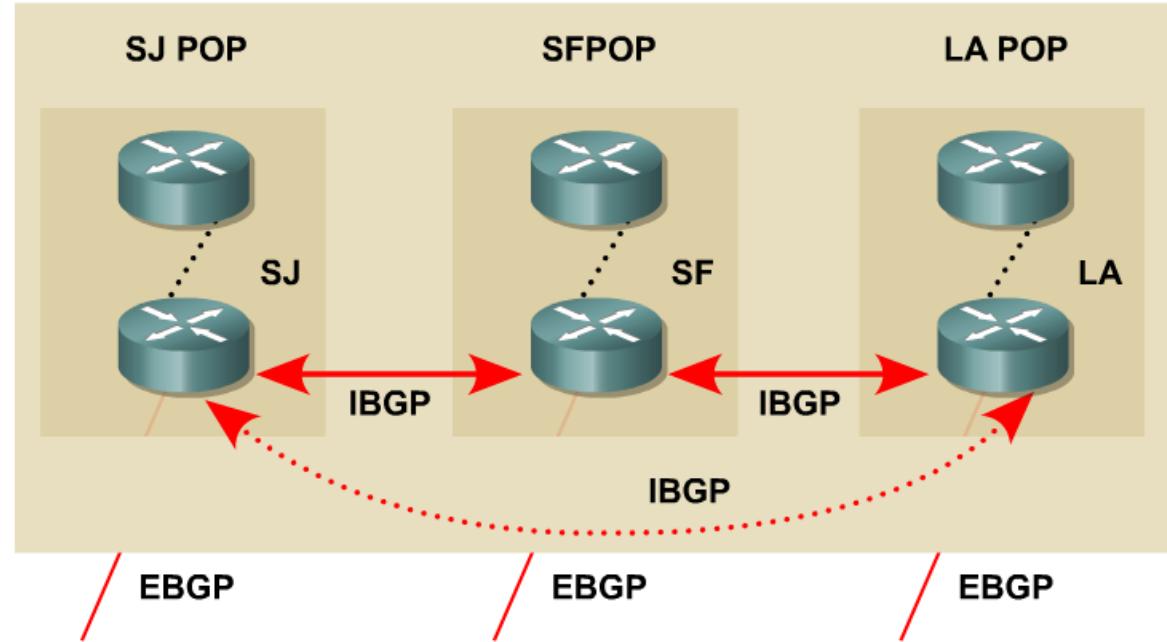


# BGP With Peer Group Example



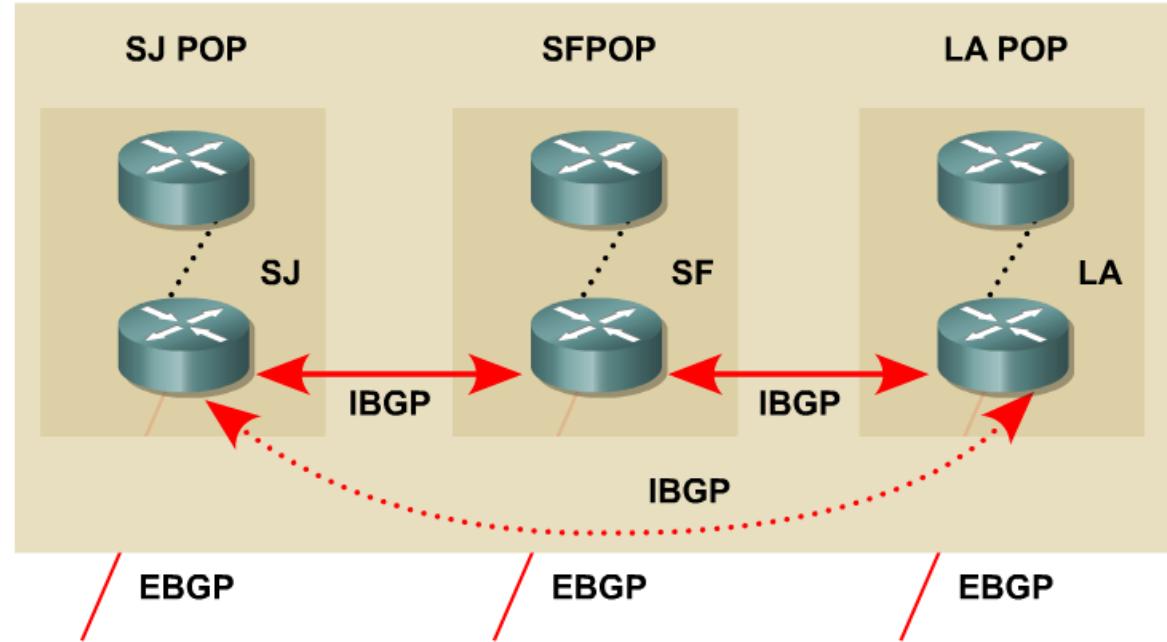
```
R1(config)# router bgp 65100
R1(config-router)# neighbor INTERNAL peer-group
R1(config-router)# neighbor INTERNAL remote-as 65100
R1(config-router)# neighbor INTERNAL update-source loopback 0
R1(config-router)# neighbor INTERNAL next-hop-self
R1(config-router)# neighbor INTERNAL distribute-list 20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

# BGP Peering



- Routes learned via IBGP peers are not propagated to other IBGP peers. – **BGP Split Horizon Rule**
- If they did, BGP routing inside the AS would present a dangerous potential for routing loops.
- For IBGP routers to learn about all BGP routes inside the AS, they must connect to every other IBGP router in a logical full IBGP mesh.
  - You can create a logical full mesh even if the routers aren't directly connected, as long as the IBGP peers can connect to each other using TCP/IP.

# BGP Peering

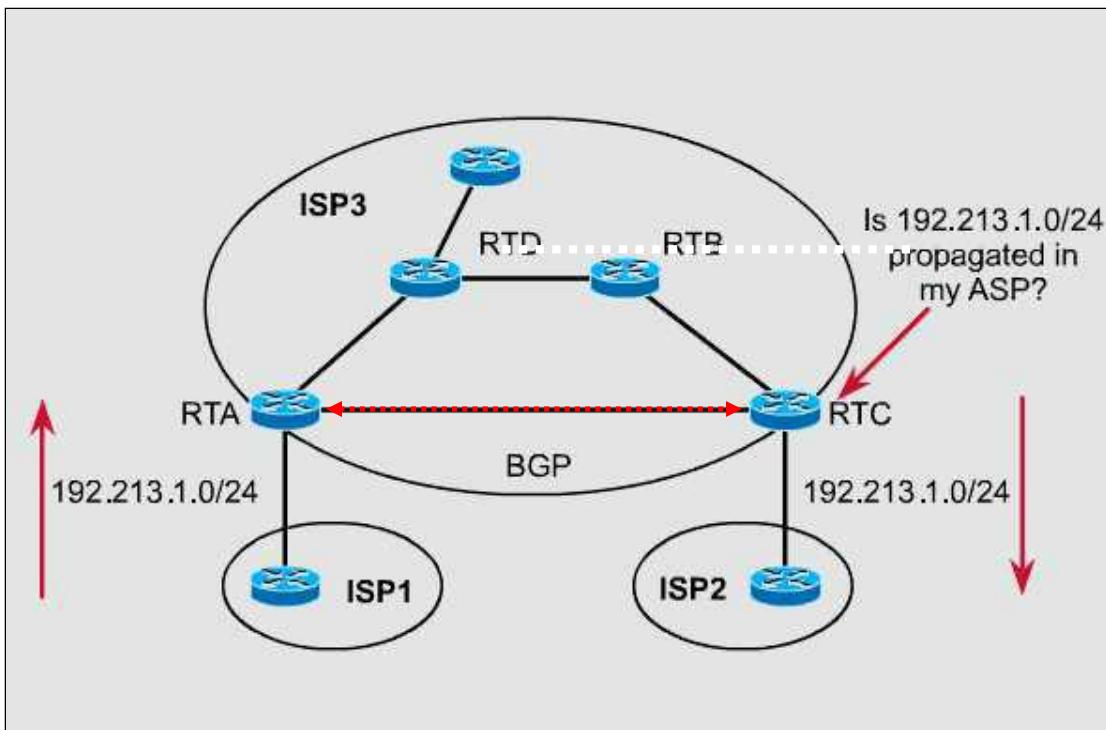


- Without dotted connection, routing in this scenario is not complete.
- EBGP routes learned by way of San Jose will not be given to Los Angeles, and EBGP routes learned by way of Los Angeles will not be given to San Jose.
- This is because the San Francisco router will not advertise IBGP routes between San Jose and Los Angeles.
- What is needed is an additional IBGP connection between San Jose and Los Angeles.
- This connection is shown as a dotted line.

# BGP Synchronization

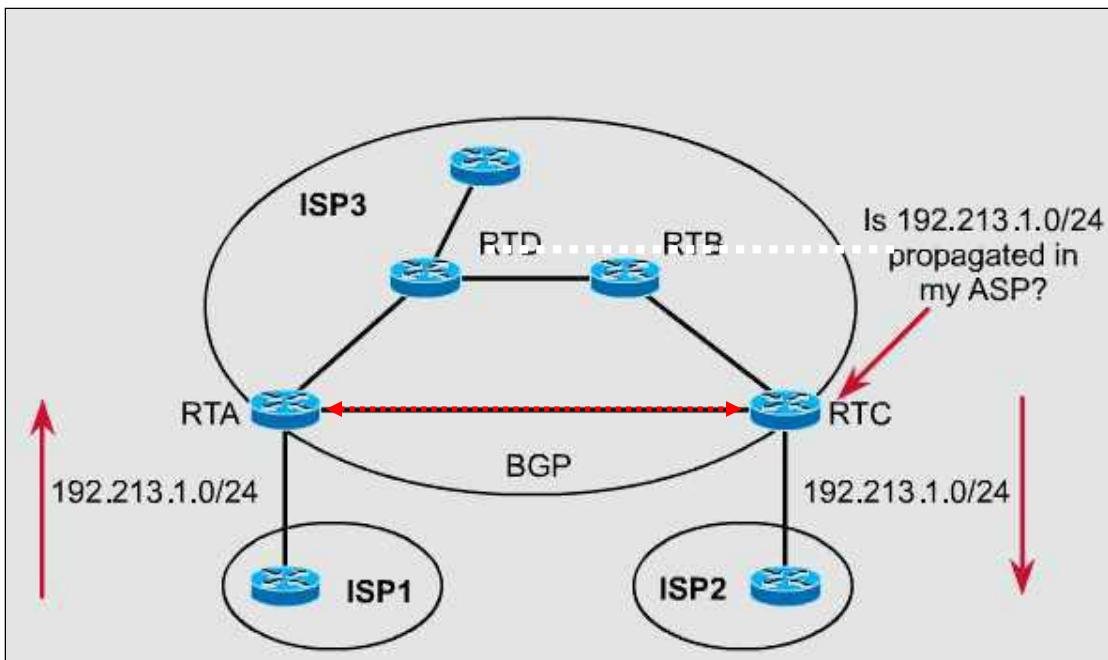
- Recall that the BGP synchronization rule states that:
  - “*A BGP router should not use, or advertise a route learned by IBGP, unless that route is local or is learned from the IGP.*”
- By default synchronization disabled, therefore BGP can use and advertise to an external BGP neighbor routes learned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

# BGP Synchronization



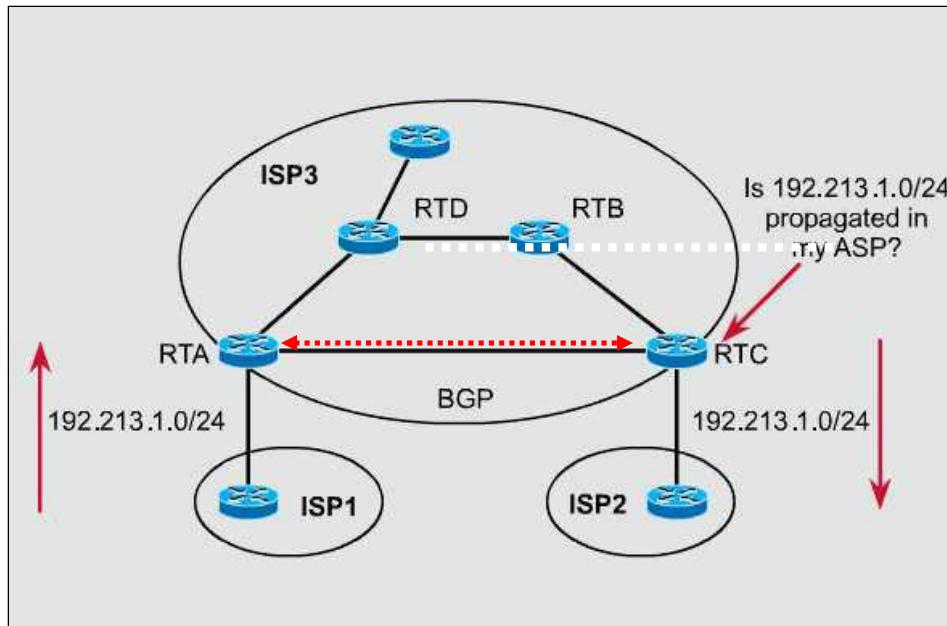
- When an IBGP router receives an update about a destination from an IBGP peer, it tries to verify reachability to that destination via an **IGP**, such as RIP or OSPF.
- If the IBGP router can't find the destination network in its **IGP** routing table, it **will not** advertise the destination to other BGP peers.

# BGP Synchronization



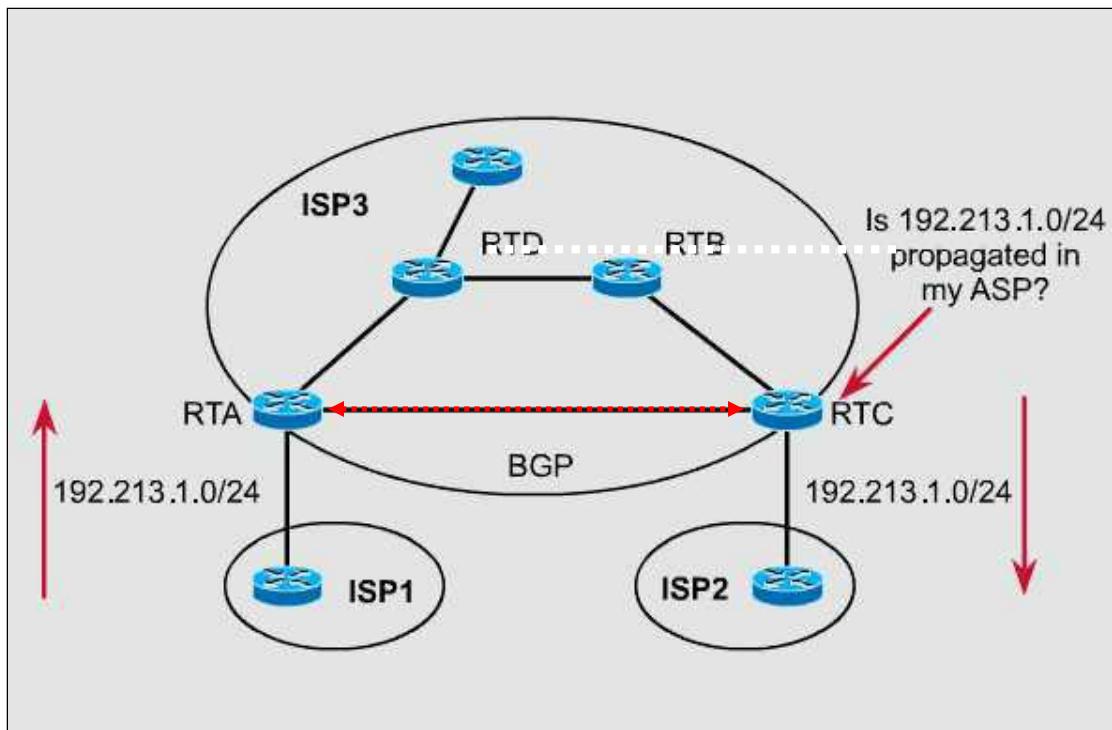
- If the route is not reachable through the **IGP** running within the AS, non-BGP routers will not be able to route traffic passing through the AS towards this destination.
- It is pointless to advertise destinations to external peers if traffic sent through this AS is going to be dropped by some non-BGP router within the AS anyway.

# BGP Synchronization



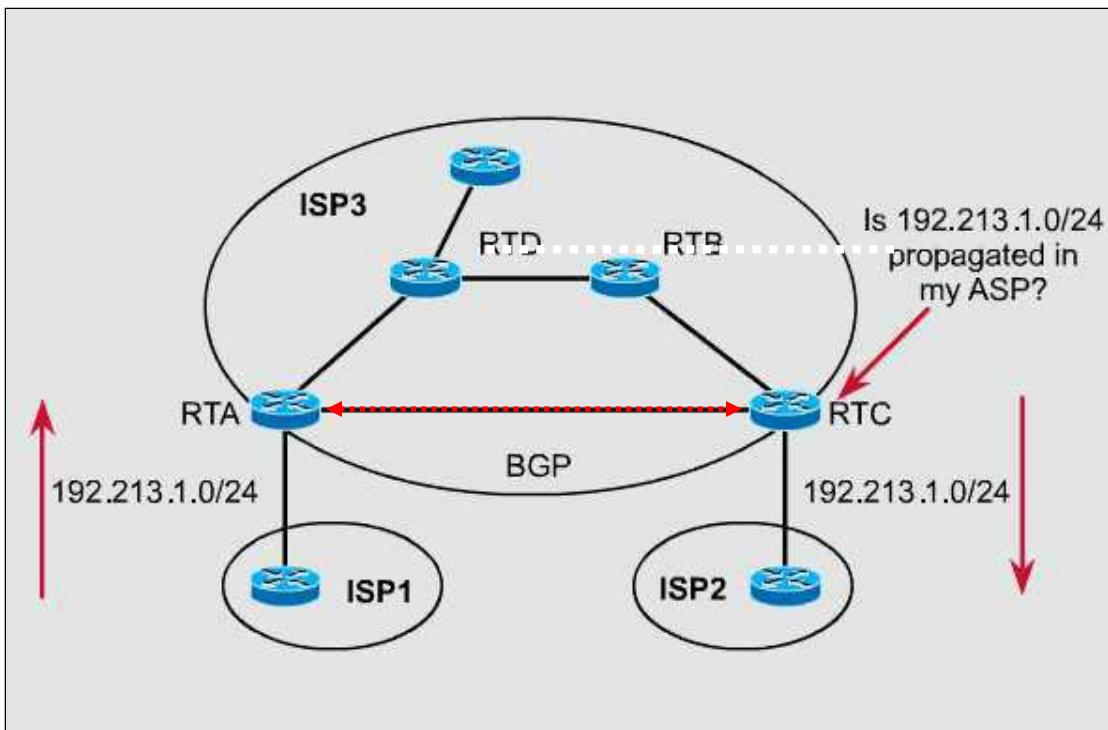
- The **BGP synchronization rule** states that a BGP router (**RTC**) should not advertise to external neighbors (**ISP2**) destinations (**192.213.1.0/24**) learned from inside BGP neighbors (**RTA**) unless those destinations are also known via an **IGP** (**RTD and RTB**).
- If a router knows about these destinations via an **IGP**, it assumes that the route has already been propagated inside the AS, and internal reachability is guaranteed.

# BGP Synchronization



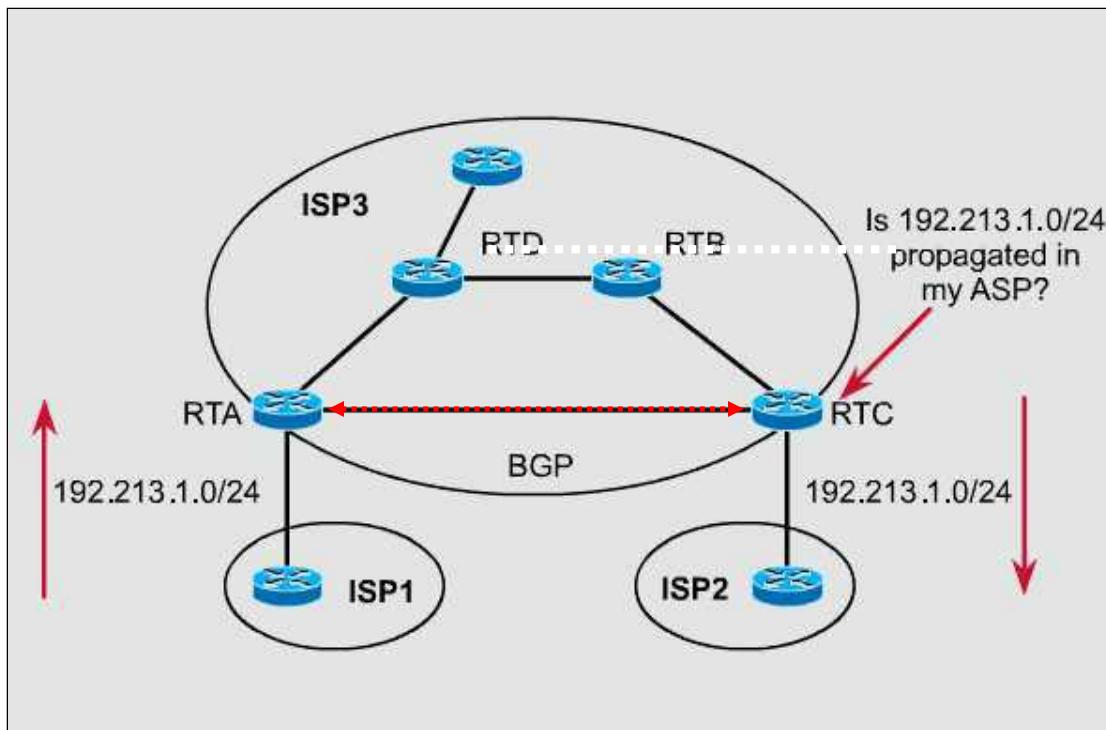
- If the IBGP router (**RTC**) does have an **IGP** route to this destination, the route is considered **synchronized**, and the router will announce it to other BGP peers (**ISP2**).
- *Otherwise*, the router will treat the route as not being synchronized with the **IGP** and will not advertise it.

# BGP Synchronization



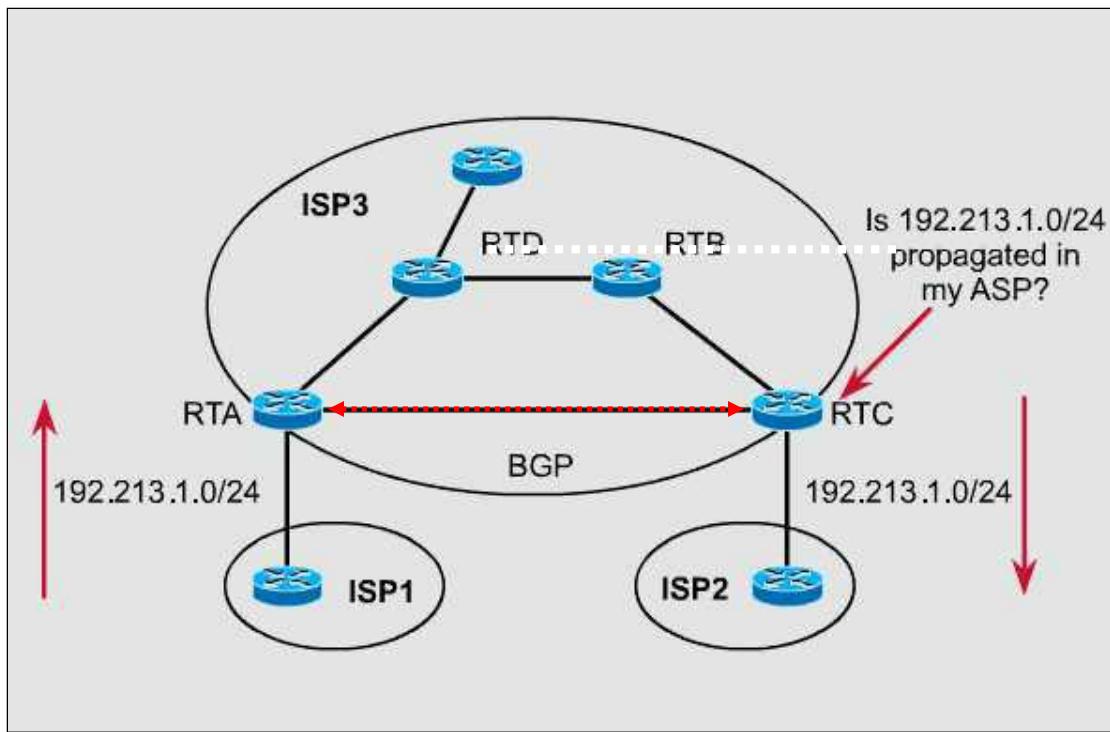
- The consequence of injecting BGP routes inside an AS is costly.
- Redistributing routes from BGP into the **IGP** will result in major overhead on the internal routers, which might not be equipped to handle that many routes.
- Besides, carrying all external routes inside an AS is not really necessary.

# BGP Synchronization



- The Cisco IOS offers an optional command called **no synchronization**.
- This command enables BGP to override the synchronization requirement, allowing the router to advertise routes learned via IBGP irrespective of an existence of an **IGP** route.

# BGP Synchronization



- In practice, two situations exist where synchronization can be safely turned off on border routers:
  - When all transit routers inside the AS are running fully meshed IBGP. Internal reachability is guaranteed because a route that is learned via EBGP on any of the border routers will automatically be passed on via IBGP to all other transit routers.
  - When the AS is not a transit AS.

# Verifying and Troubleshooting BGP

# Verifying and Troubleshooting BGP

| Command                                                         | Description                                                                                                                 |
|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <b>show ip bgp</b>                                              | Displays entries in the BGP table.<br>Specify a network number to get more specific information about a particular network. |
| <b>show ip bgp neighbors</b>                                    | Displays detailed information about the TCP and BGP connections to neighbors.                                               |
| <b>show ip bgp summary</b>                                      | Displays the status of all BGP connections.                                                                                 |
| <b>show ip bgp rib-failure</b>                                  | Displays BGP routes that were not installed in the RIB and the reason that they were not installed.                         |
| <b>debug ip bgp [dampening   events   keepalives   updates]</b> |                                                                                                                             |

# Monitoring Received BGP Routes

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

# Verifying BGP: show ip bgp

Display the BGP topology database (the BGP table).

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

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

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

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

This route is offered to the IP routing table.

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

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

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

R1# **show ip bgp**

BGP table version is 14, local router ID is 172.31.11.1

Status codes: s suppressed, d damped, h history, \* valid, > best, i - internal, r RIB-failure, S Stale

Origin codes: i - IGP, e - EGP, ? - incomplete

| Network          | Next Hop    | Metric | LocPrf | Weight | Path          |
|------------------|-------------|--------|--------|--------|---------------|
| *> 10.1.0.0/24   | 0.0.0.0     | 0      |        | 32768  | i             |
| * i              | 10.1.0.2    | 0      | 100    | 0      | i             |
| *> 10.1.1.0/24   | 0.0.0.0     | 0      |        | 32768  | i             |
| *>i10.1.2.0/24   | 10.1.0.2    | 0      | 100    | 0      | i             |
| *> 10.97.97.0/24 | 172.31.1.3  | 0      |        | 64998  | 64997 i       |
| *                | 172.31.11.4 | 0      |        | 64999  | 64997 i       |
| * i              | 172.31.11.4 | 0      | 100    | 0      | 64999 64997 i |
| *> 10.254.0.0/24 | 172.31.1.3  | 0      |        | 64998  | i             |
| *                | 172.31.11.4 | 0      |        | 64999  | 64998 i       |
| * i              | 172.31.1.3  | 0      | 100    | 0      | 64998 i       |
| r> 172.31.1.0/24 | 172.31.1.3  | 0      |        | 64998  | i             |
| r                | 172.31.11.4 | 0      |        | 64999  | 64998 i       |
| r i              | 172.31.1.3  | 0      | 100    | 0      | 64998 i       |
| *> 172.31.2.0/24 | 172.31.1.3  | 0      |        | 64998  | i             |

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

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

If blank the route is from the current autonomous system.

The last column displays the ORIGIN attribute).

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

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

# Verifying BGP: show ip rib-failure

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

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

# Verifying BGP: show ip bgp summary

Verify the BGP neighbor relationship.

```
R1# show ip bgp summary
BGP router identifier 10.1.1.1, local AS number 65001
BGP table version is 124, main routing table version 124
9 network entries using 1053 bytes of memory
22 path entries using 1144 bytes of memory
12/5 BGP path/bestpath attribute entries using 1488 bytes of memory
6 BGP AS-PATH entries using 144 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 3829 total bytes of memory
BGP activity 58/49 prefixes, 72/50 paths, scan interval 60 secs
```

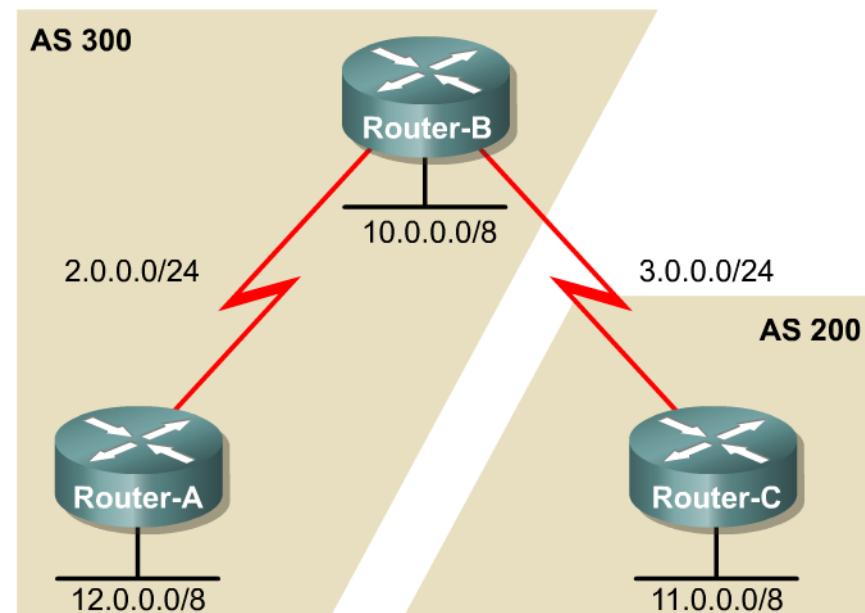
| Neighbor    | V | AS    | MsgRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down  | State/PfxRcd |
|-------------|---|-------|---------|---------|--------|-----|------|----------|--------------|
| 10.1.0.2    | 4 | 65001 | 11      | 11      | 124    | 0   | 0    | 00:02:28 | 8            |
| 172.31.1.3  | 4 | 64998 | 21      | 18      | 124    | 0   | 0    | 00:01:13 | 6            |
| 172.31.11.4 | 4 | 64999 | 11      | 10      | 124    | 0   | 0    | 00:01:11 | 6            |

# Verifying BGP: debug ip bgp updates

Verify the BGP neighbor relationship.

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

# Example



```
Router-A#show ip bgp
BGP table version is 12, local router ID is 12.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - inter
Origin codes: i - IGP, e - EGP, ? - incomplete
```

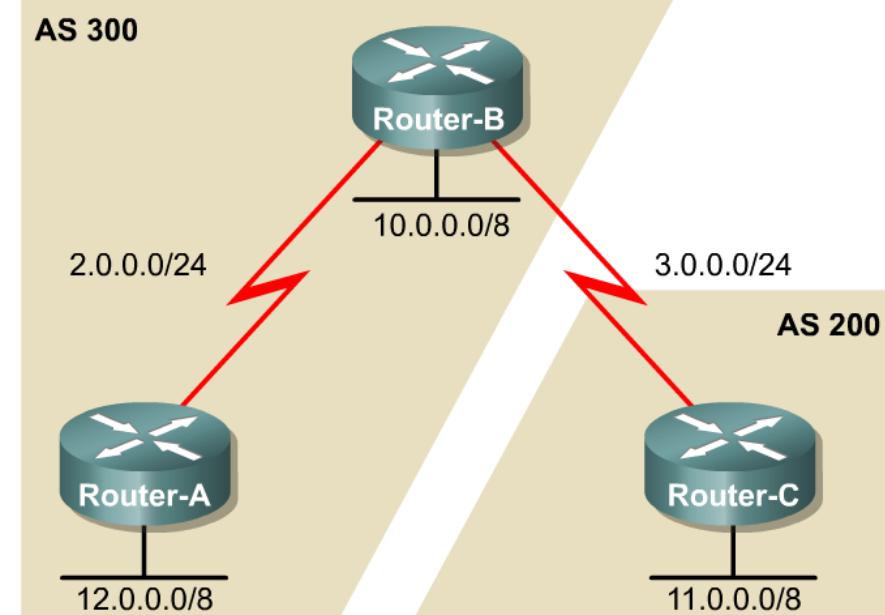
| Network     | Next Hop | Metric | LocPrf | Weight | Path  |
|-------------|----------|--------|--------|--------|-------|
| *>i10.0.0.0 | 2.0.0.1  | 0      | 100    | 0      | i     |
| *>i11.0.0.0 | 2.0.0.1  | 0      | 100    | 0      | 200 i |
| *> 12.0.0.0 | 0.0.0.0  | 0      |        | 32768  | i     |

```
Router-A#
```

```
Router-A#show ip bgp summary
```

| Neighbor | V | AS  | MsgRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down  | State/ |
|----------|---|-----|---------|---------|--------|-----|------|----------|--------|
| 2.0.0.1  | 4 | 300 | 201     | 197     | 12     | 0   | 0    | 00:54:23 |        |

# Example



```
Router-B#show ip bgp
BGP table version is 12, local router ID is 10.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - inte
Origin codes: i - IGP, e - EGP, ? - incomplete
```

| Network     | Next Hop | Metric | LocPrf | Weight | Path  |
|-------------|----------|--------|--------|--------|-------|
| *> 10.0.0.0 | 0.0.0.0  | 0      |        | 32768  | i     |
| *> 11.0.0.0 | 3.0.0.2  | 0      |        | 0      | 200 i |
| *>i12.0.0.0 | 2.0.0.2  | 0      | 100    | 0      | i     |

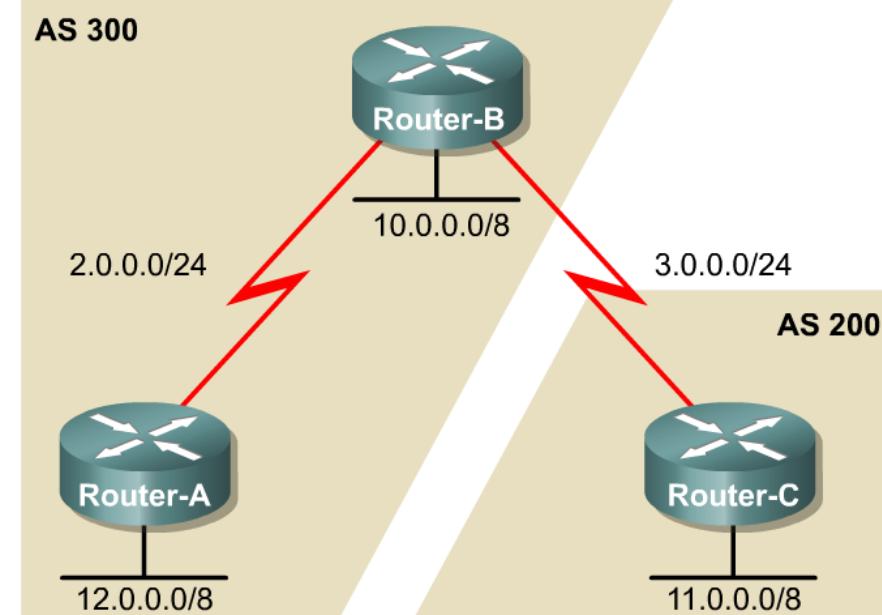
```
Router-B#
```

```
Router-B#show ip bgp summary
```

| Neighbor | V | AS  | MsgRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down  | State/ |
|----------|---|-----|---------|---------|--------|-----|------|----------|--------|
| 2.0.0.2  | 4 | 300 | 202     | 206     | 12     | 0   | 0    | 00:59:51 |        |
| 3.0.0.2  | 4 | 200 | 172     | 175     | 12     | 0   | 0    | 02:17:05 |        |

```
Router-B#
```

# Example



```
Router-C#show ip bgp
BGP table version is 6, local router ID is 11.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - inte
Origin codes: i - IGP, e - EGP, ? - incomplete
```

| Network     | Next Hop | Metric | LocPrf | Weight | Path  |
|-------------|----------|--------|--------|--------|-------|
| *> 10.0.0.0 | 3.0.0.1  | 0      |        | 0      | 300 i |
| *> 11.0.0.0 | 0.0.0.0  | 0      |        | 32768  | i     |
| *> 12.0.0.0 | 3.0.0.1  |        |        | 0      | 300 i |

```
Router-C#
```

---

```
Router-C#show ip bgp summary
```

| Neighbor | V | AS  | MsgRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down  | State/ |
|----------|---|-----|---------|---------|--------|-----|------|----------|--------|
| 3.0.0.1  | 4 | 300 | 178     | 176     | 6      | 0   | 0    | 02:20:15 |        |

```
Router-C#
```

# BGP Path Manipulation Using Route Maps

# The Route Map Command

```
RTA(config)#route-map MYMAP permit 10
RTA(config-route-map)#match ip address 1
RTA(config-route-map)#set metric 5
RTA(config-route-map)#exit
RTA(config)#access-list 1 permit 1.1.1.0 0.0.0.255
```

- Router(config)#**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*]
- BGP input and output policies are defined, generally, using route maps.
- Route maps are used with BGP to control and modify routing information and to define the conditions by which routes are redistributed between routing domains.
- Note that *map-tag* is a name that identifies the route map; the *sequence-number* indicates the position that an instance of the route map is to have in relation to other instances of the same route map.
- Instances are ordered sequentially, starting with the number 10 by default.

# Applying a Route Map to BGP

```
RTA(config)#route-map MYMAP permit 10
RTA(config-route-map)#match ip address 1
RTA(config-route-map)#set metric 5
RTA(config-route-map)#exit
RTA(config)#access-list 1 permit 1.1.1.0 0.0.0.255

RTA(config)#route bgp 100
RTA(config-route)#neighbor 172.16.20.2 remote-as 300
RTA(config-route)#neighbor 172.16.20.2 route-map MYMAP out
```

- Access list 1 identifies all routes of the form 1.1.1.x.
- A routing update of the form 1.1.1.x will match the access list and will be propagated with a metric set to five (5).
- This is because of the **permit** keyword in the access list.
- A route map can be applied on the incoming, using the keyword **in**, or the outgoing, using the keyword **out**, BGP updates.
- The route map MYMAP is applied on the outgoing updates toward BGP neighbor 172.16.20.2.

# Route Maps and BGP

- Policy Based Routing (PBR) was used for redistribution.
  - Route maps are implemented using the **redistribute** command.
- Route maps were used to define a routing policy other than basic destination-based routing using the routing table.
  - Route maps are implemented using the **ip policy route-map** command.
- Route maps will be used with BGP to assign or alter BGP attributes.
  - Route maps are implemented using the **neighbor route-map** command.

# Configuring Route Maps in BGP

Sample implementation plan:

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

# Implementing Route Maps in BGP

```
Router(config) #
```

```
route-map map-tag [permit | deny] [sequence-number]
```

- Defines the route map conditions.

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

```
match {criteria}
```

- Defines the criteria to match.

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

```
set {actions}
```

- Defines the action to be taken on a match.

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

```
neighbor {ip-address | peer-group-name} route-map map-name
{in | out}
```

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

# match Commands Used in BGP

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

\* Partial list

# match as-path Command

- Match a BGP autonomous system path access list.

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

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

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

# match ip-address Command

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

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

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

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

# set Commands Used in BGP

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

\* Partial list

# set weight Command

- Specify the BGP weight for the routing table.

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

```
 set weight number
```

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

# set local-preference

## Command

- Specify a preference value for the AS path.

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

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

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

# set as-path Command

- Modify an AS path for BGP routes.

Router(config-route-map) #

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

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

# set metric Command

- Specify a preference value for the AS path.

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

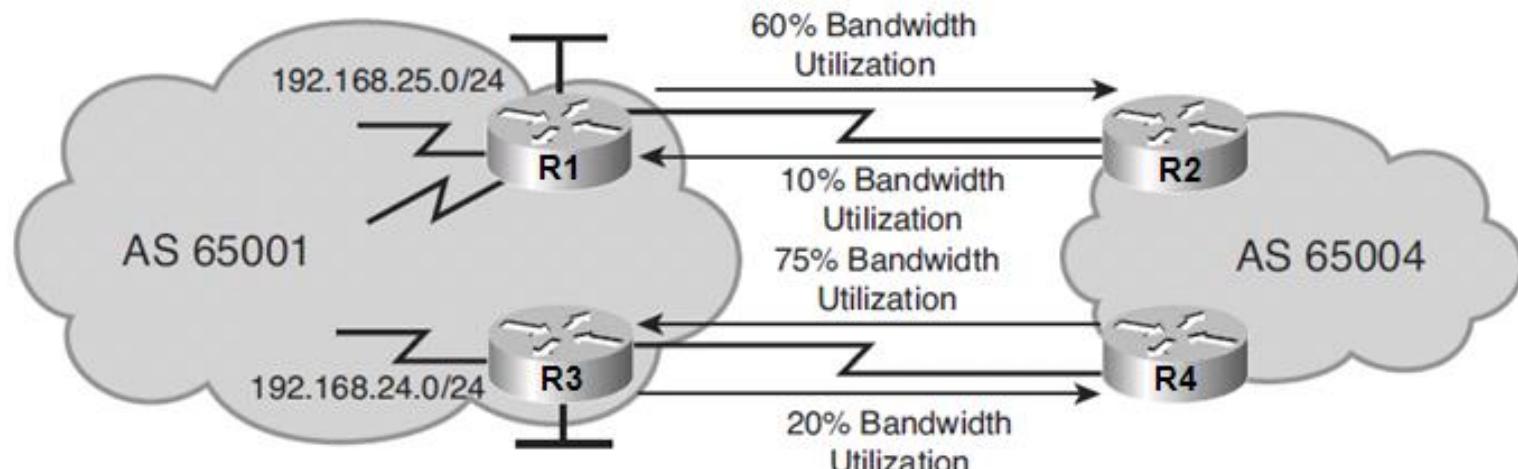
```
 set metric metric-value
```

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

# BGP Path Manipulation

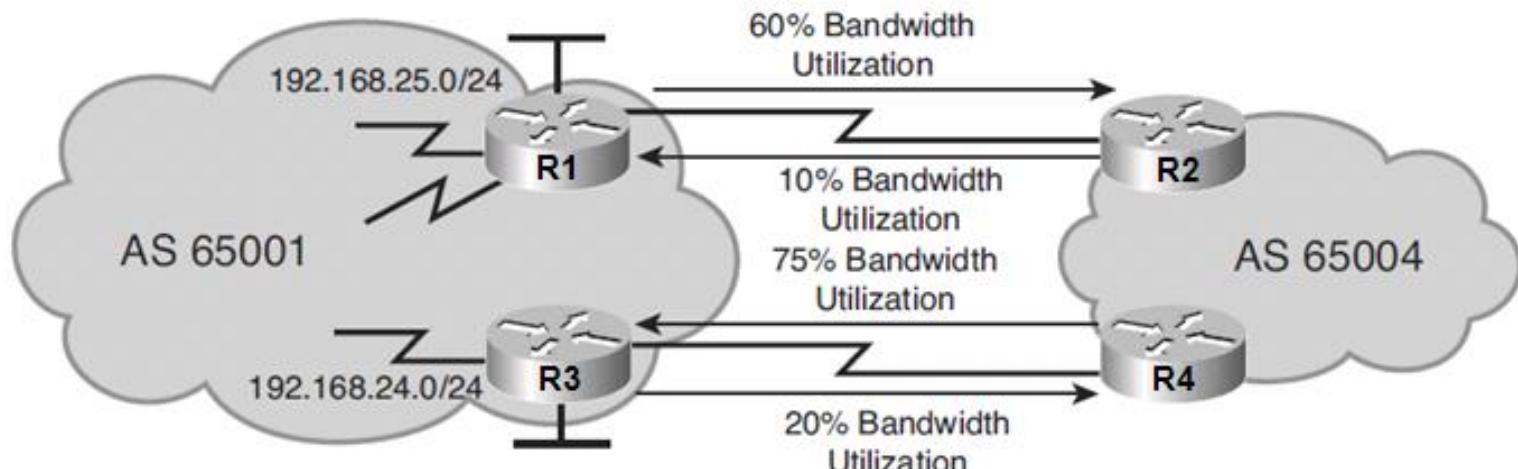
- Unlike IGPs, BGP was never designed to choose the quickest path.
- BGP was designed to manipulate traffic flow to maximize or minimize bandwidth use.

# BGP Without Routing Policy Example #1



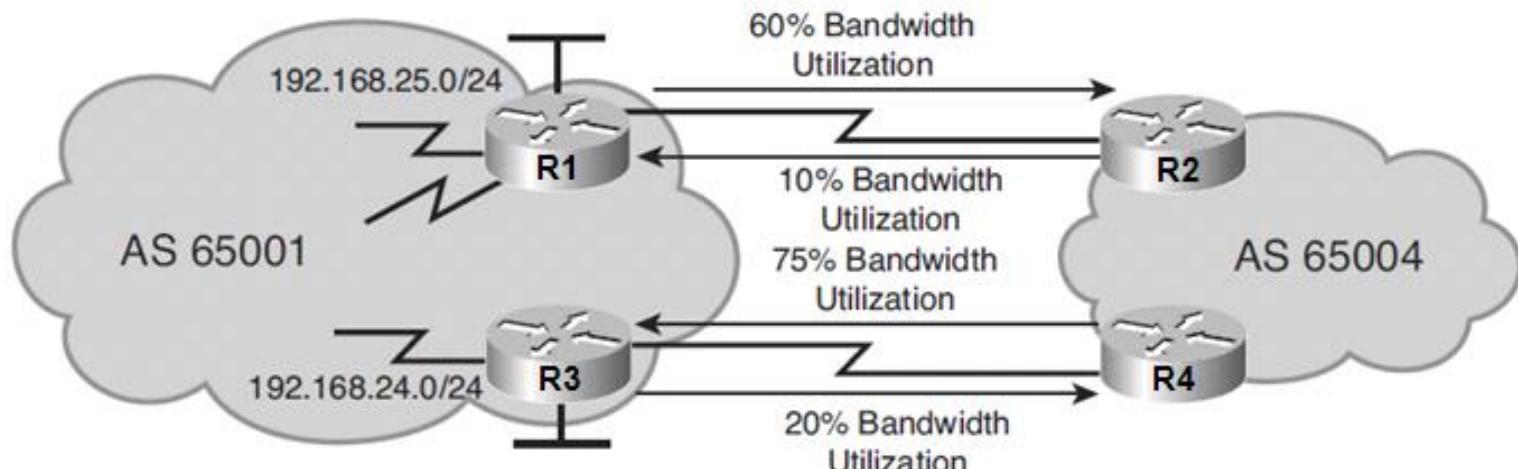
- In this example consider that:
  - R1 is using 60% of its outbound bandwidth to AS 65004.
  - R3 is using 20% of its outbound bandwidth to AS 65004.
  - R2 is using 10% of its outbound bandwidth to AS 65001.
  - R4 is using 75% of its outbound bandwidth to AS 65001.
- Traffic should be diverted using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

# Which traffic should be re-routed?



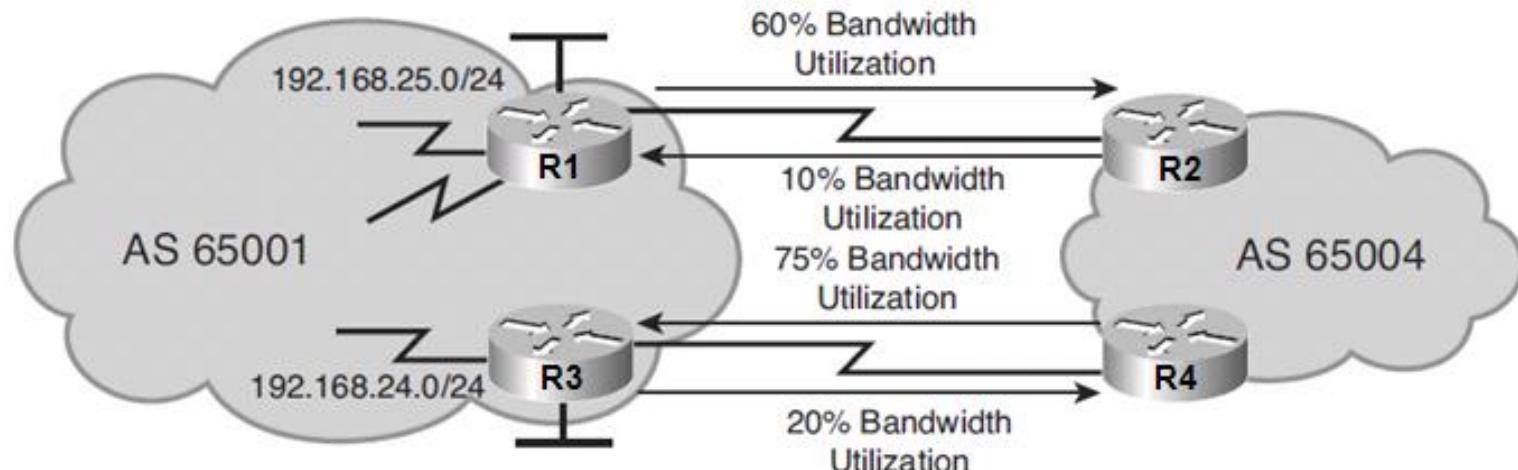
- To determine which path to manipulate, perform a traffic analysis on Internet-bound traffic by examining the most heavily visited addresses, web pages, or domain names.
  - Examine network management records or accounting information.
- If a heavily accessed traffic pattern is identified, a route map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

# BGP With Routing Policy Example #1



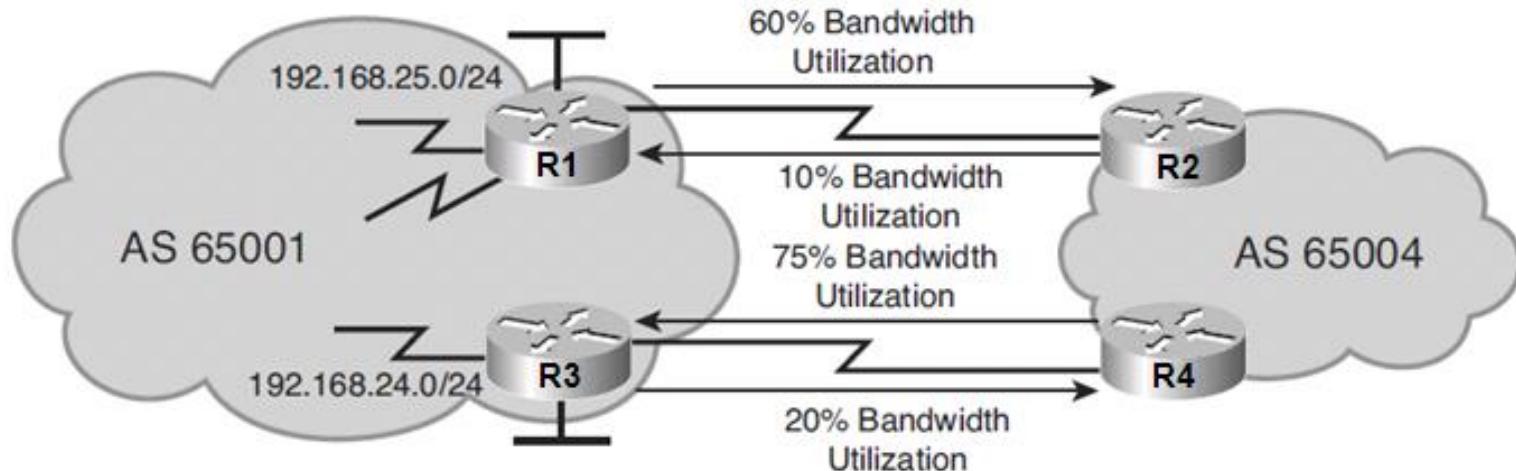
- For example, assume that 35% of all traffic from AS 65001 has been going to <http://www.cisco.com>.
  - The administrator does a reverse DNS lookup and obtains the Cisco IP address and AS number.
  - A route map can be used to change the local preference to manipulate packets destined to Cisco's network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

# BGP Routing Policy Example #2



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

# BGP Routing Policy Example #2



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

## BGP Route Selection Process

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

# Change the Weight

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

## BGP Route Selection Process

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

# Changing the Default Weight Example

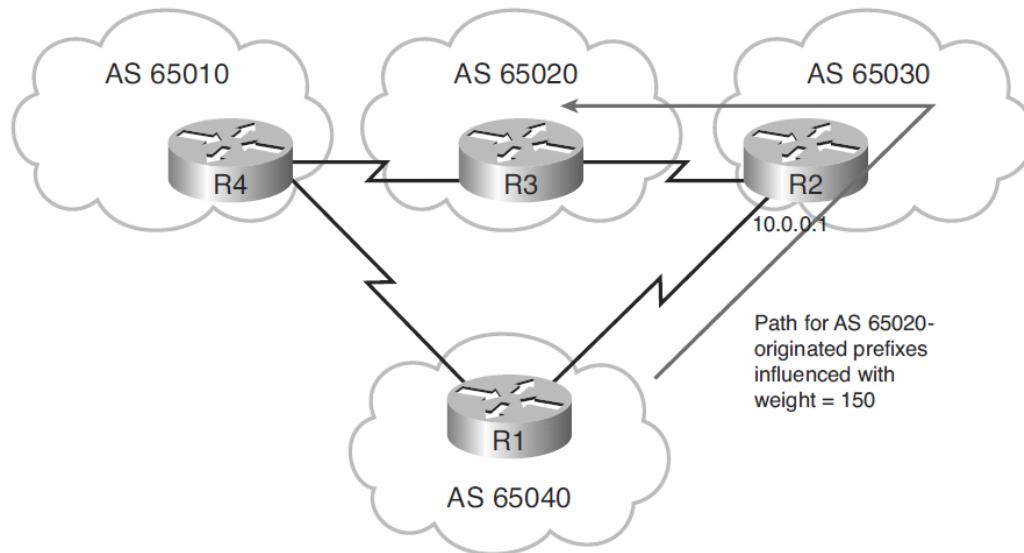
- Assign a default weight to all routes from a peer.

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

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

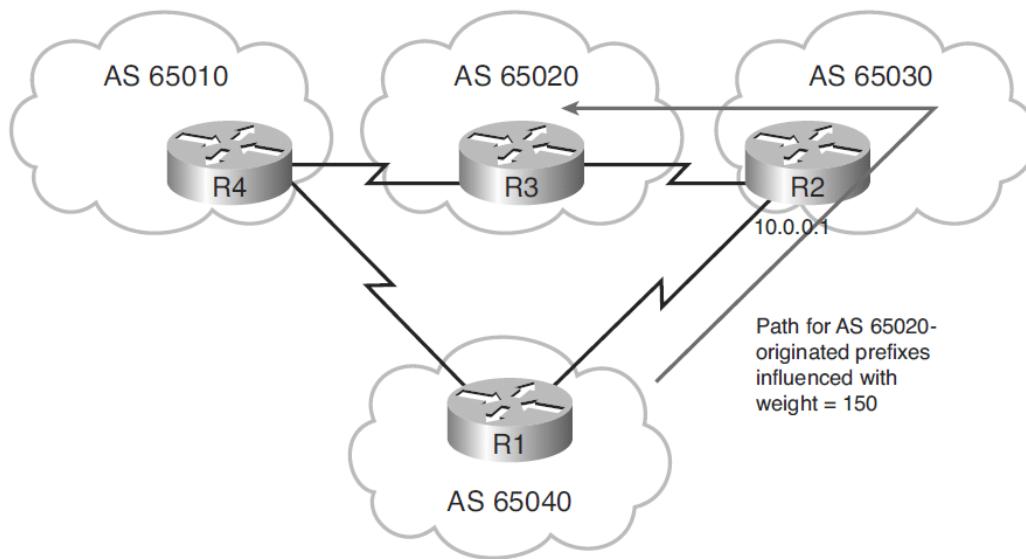
- Routes learned through another BGP peer have a default weight of 0 and routes sourced by the local router have a default weight of 32768.
- The *number* is the weight to assign.
  - Acceptable values are from 0 to 65535.
- The route with the highest weight will be chosen as the preferred route when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

# Changing Weight with Route Map Example



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

# Changing Weight with Route Map Example



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

# Configure an Autonomous System ACL

- Configure an autonomous system path filter.

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

```
ip as-path access-list acl-number {permit | deny}
 regexp
```

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

# Regular Expression Syntax

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

# Regular Expression Examples

| Regular Expression | Resulting Expression                                                       |
|--------------------|----------------------------------------------------------------------------|
| <b>a*</b>          | Expression indicates any occurrence of the letter "a", which includes none |
| <b>a+</b>          | indicates that at least one occurrence of the letter "a" must be present   |
| <b>ab?a</b>        | Expression matches "aa" or "aba".                                          |
| <b>_100_</b>       | Expression means via AS100.                                                |
| <b>_100\$</b>      | Expression indicates an origin of AS100.                                   |
| <b>^100 .*</b>     | Expression indicates transmission from AS100                               |
| <b>^\$</b>         | Expression indicates origination from this AS                              |

# Change the Local Preference

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

## BGP Route Selection Process

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

# Setting Default Local Preference Example

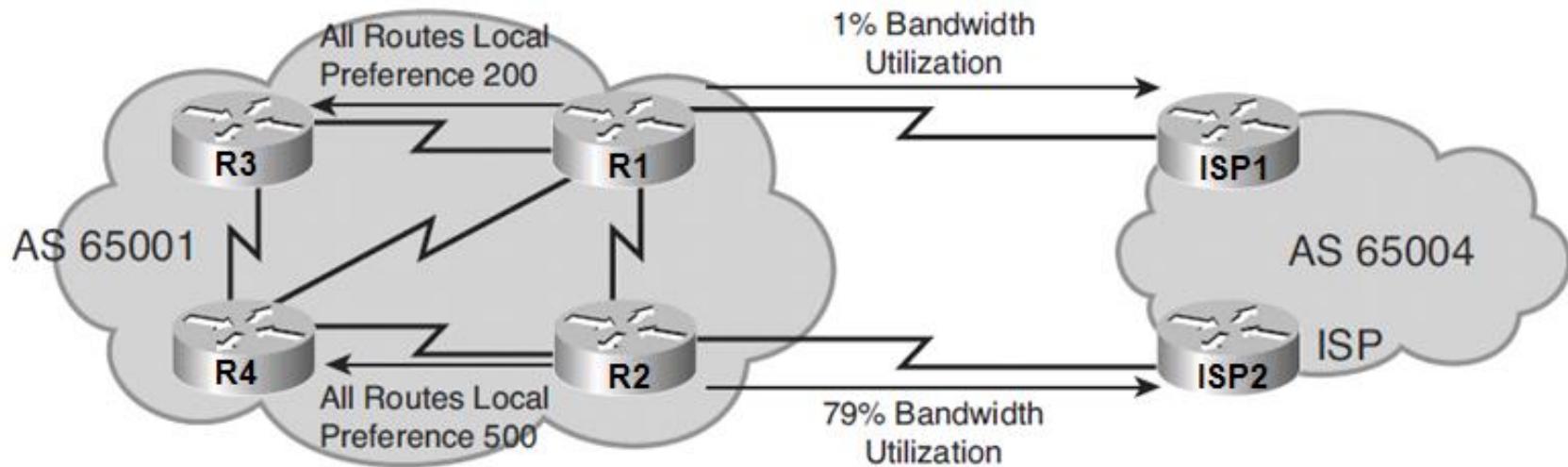
- Change the default local preference for outgoing routes.

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

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

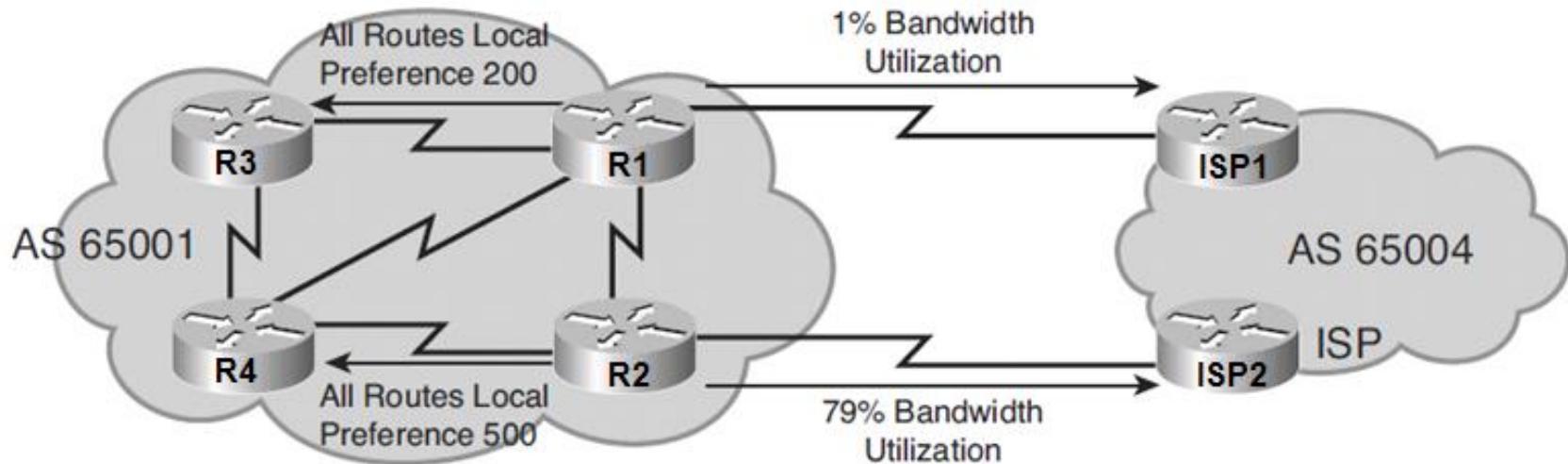
- The local preference attribute applies a degree of preference to a route during the BGP best path selection process.
  - The attribute is exchanged only between iBGP peers.
  - The route with the highest local preference is preferred.
- The *number* is the local preference value from 0 to 4294967295.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

# Setting Default Local Preference Example



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

# Setting Default Local Preference Example



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

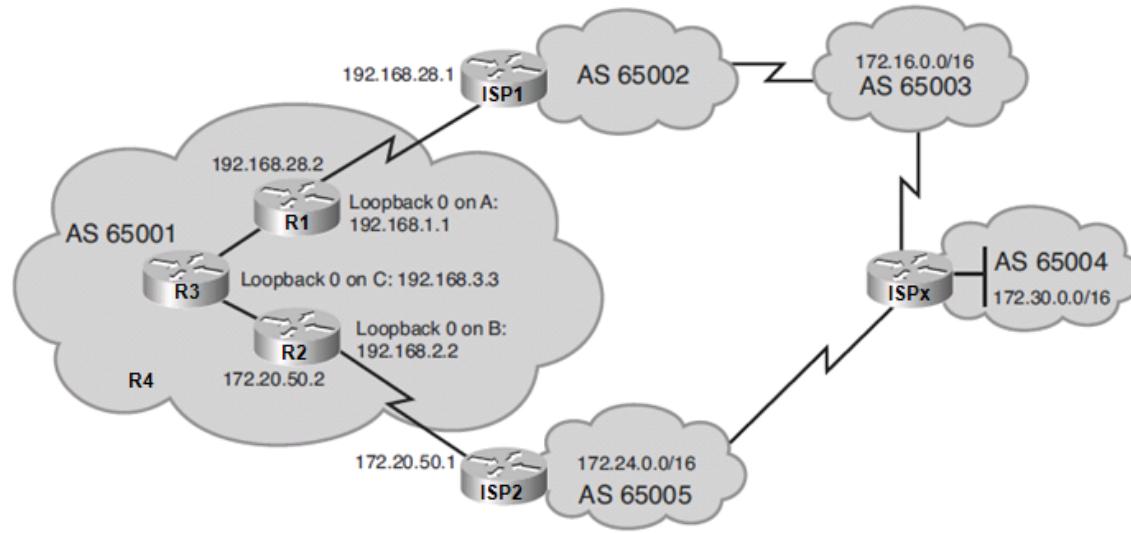
```

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

```

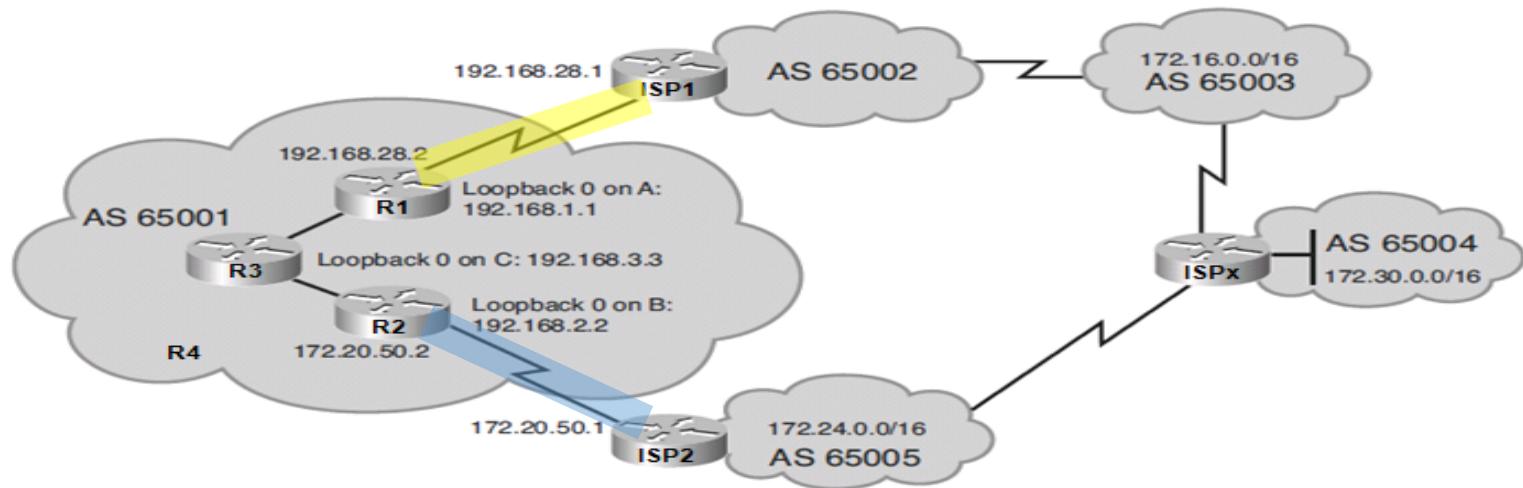
- The resulting configuration makes the IBGP routers in AS 65001 send all Internet bound traffic to R2, but the R1 to ISP1 link is underutilized.
- Route maps could be configured to select specific routes to have a higher local preference.

# Local Preference and Route Map Example



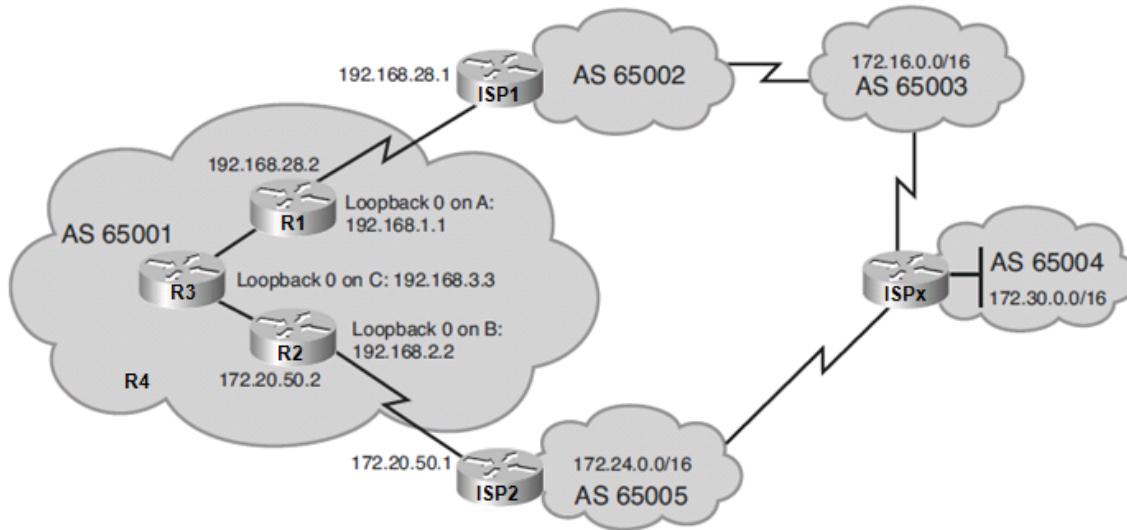
- The BGP routing policy results in the following:
  - All routes have a weight of 0 and a default local preference of 100.
  - BGP uses the shortest AS-path to select the best routes as follows:
    - For network 172.16.0.0, the shortest AS-path is through ISP1.
    - For network 172.24.0.0, the shortest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

# Local Preference and Route Map Example



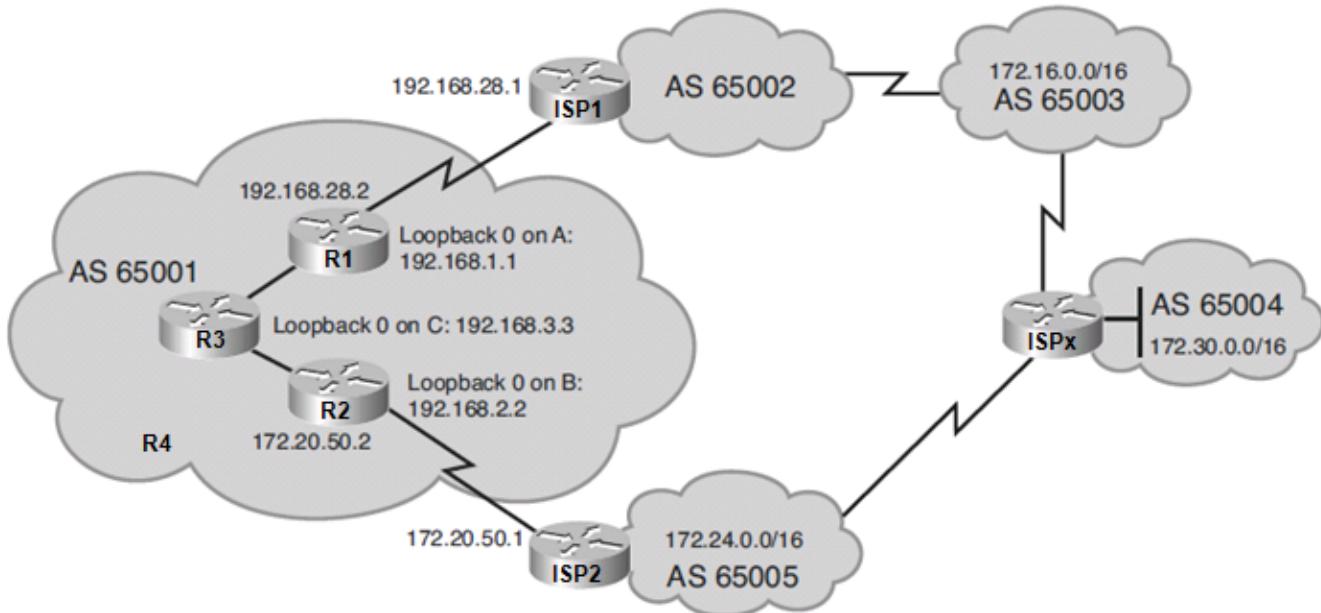
```
R3# show ip bgp
BGP table version is 7, local router ID is 192.168.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal, r
RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
Network Next Hop Metric LocPrf Weight Path
* i172.16.0.0 172.20.50.1 100 0 65005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

# Local Preference and Route Map Example



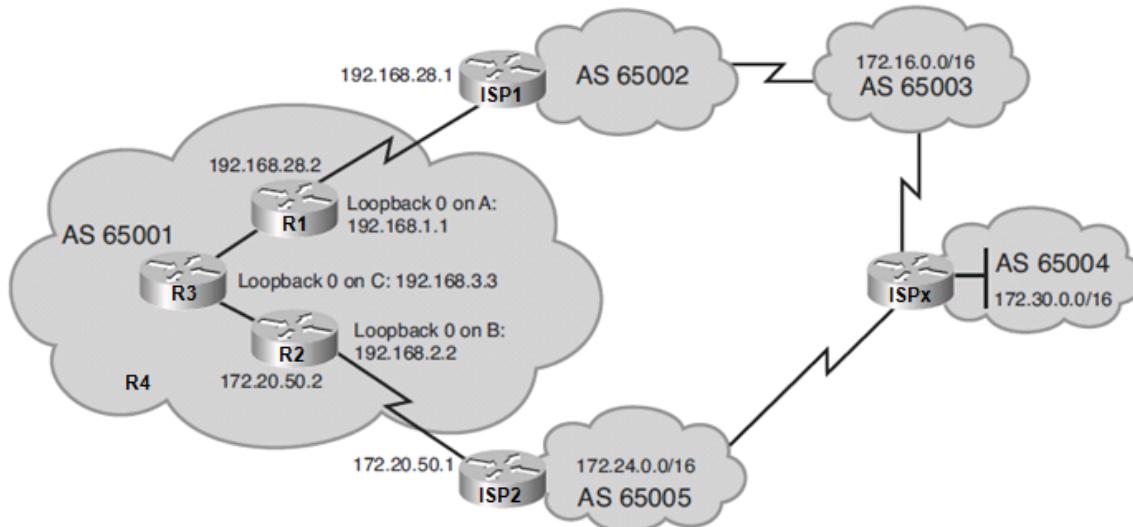
- A traffic analysis reveals the following traffic patterns:
  - 10% of traffic flows from R1 to ISP1 to network 172.16.0.0.
  - 50% of Internet traffic flow from R2 to ISP2 to networks network 172.24.0.0 and network 172.30.0.0.
  - The remaining 40 percent is going to other destinations.
- A solution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

# Local Preference and Route Map Example



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

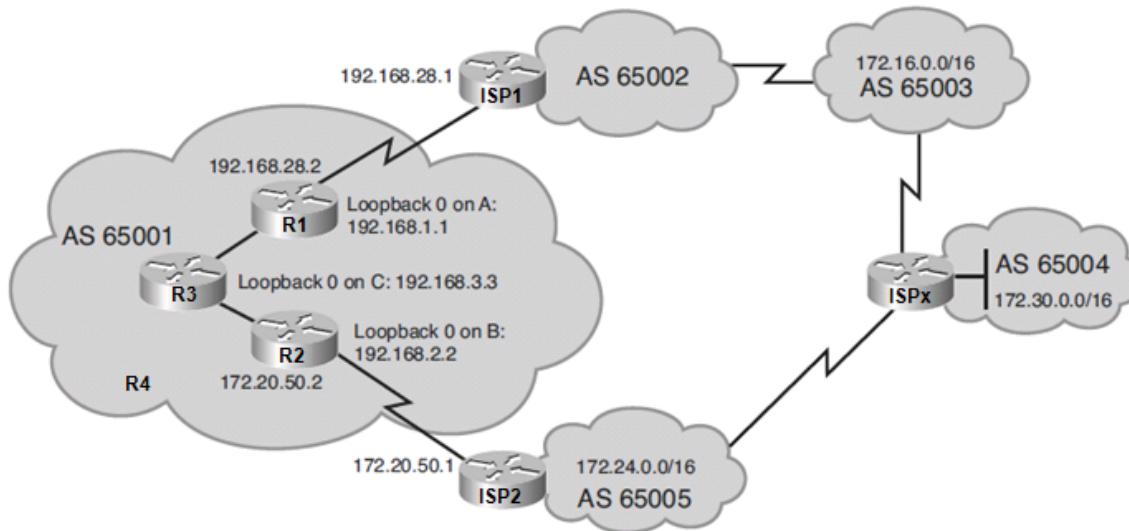
# Local Preference and Route Map Example



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

```

# Local Preference and Route Map Example



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

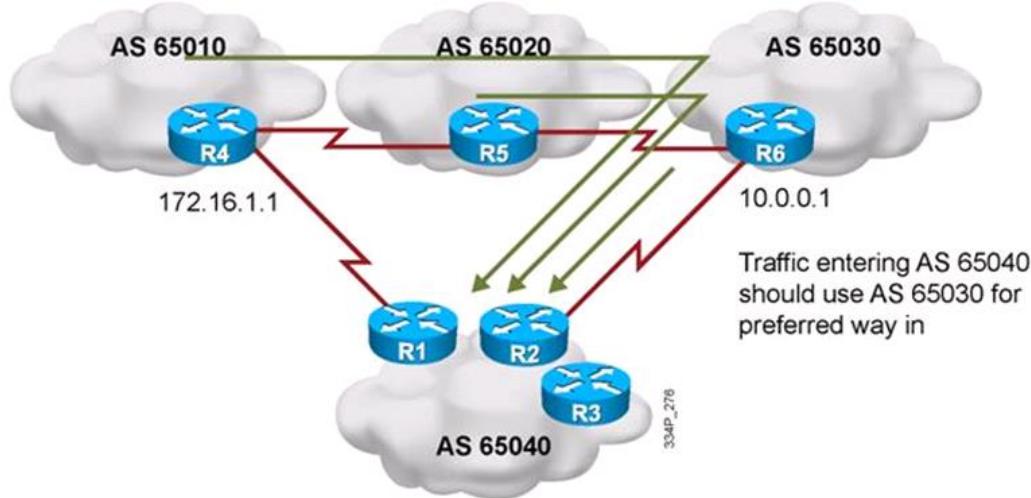
# Modifying the AS Path

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

## BGP Route Selection Process

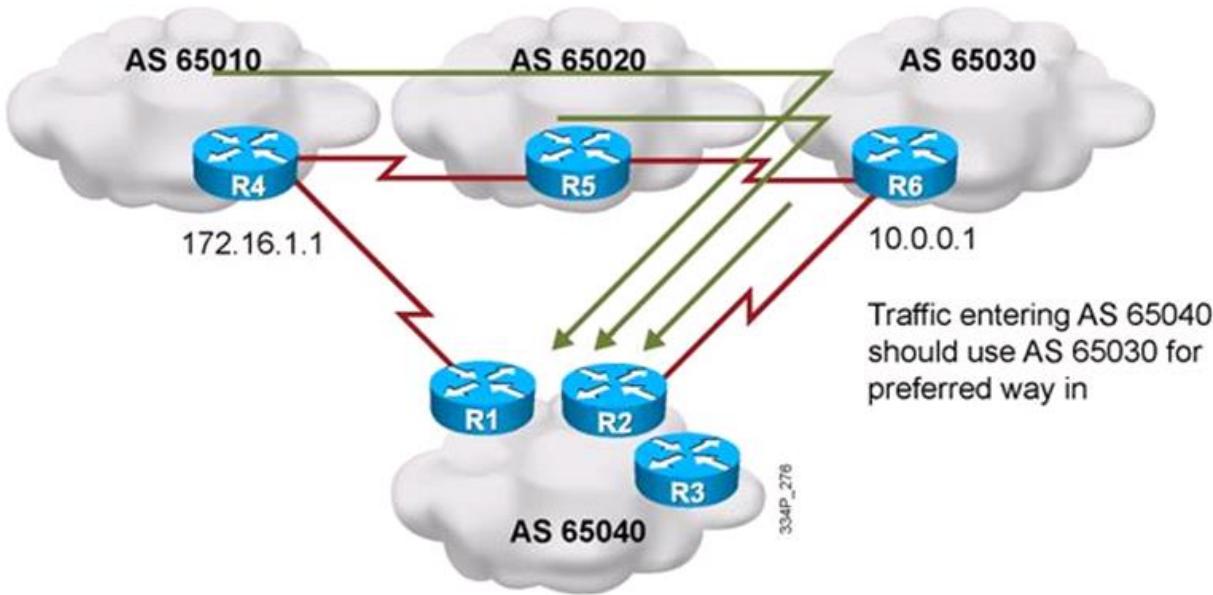
1. Prefer highest Weight
2. Prefer highest LOCAL\_PREF
3. Prefer locally generated routes
4. Prefer shortest AS\_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID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

# Modifying the AS Path Example



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

# Modifying the AS Path Example



```
R1(config)# route-map SET-AS-PATH permit 10
R1(config-route-map)# set as-path prepend 65040 65040 65040
R1(config-route-map)# exit
R1(config)# router bgp 65040
R1(config-router)# neighbor 172.16.1.1 remote-as 65010
R1(config-router)# neighbor 172.16.1.1 route-map SET-AS-PATH out
R1(config-router)# exit
R1(config)#
334P_276
```

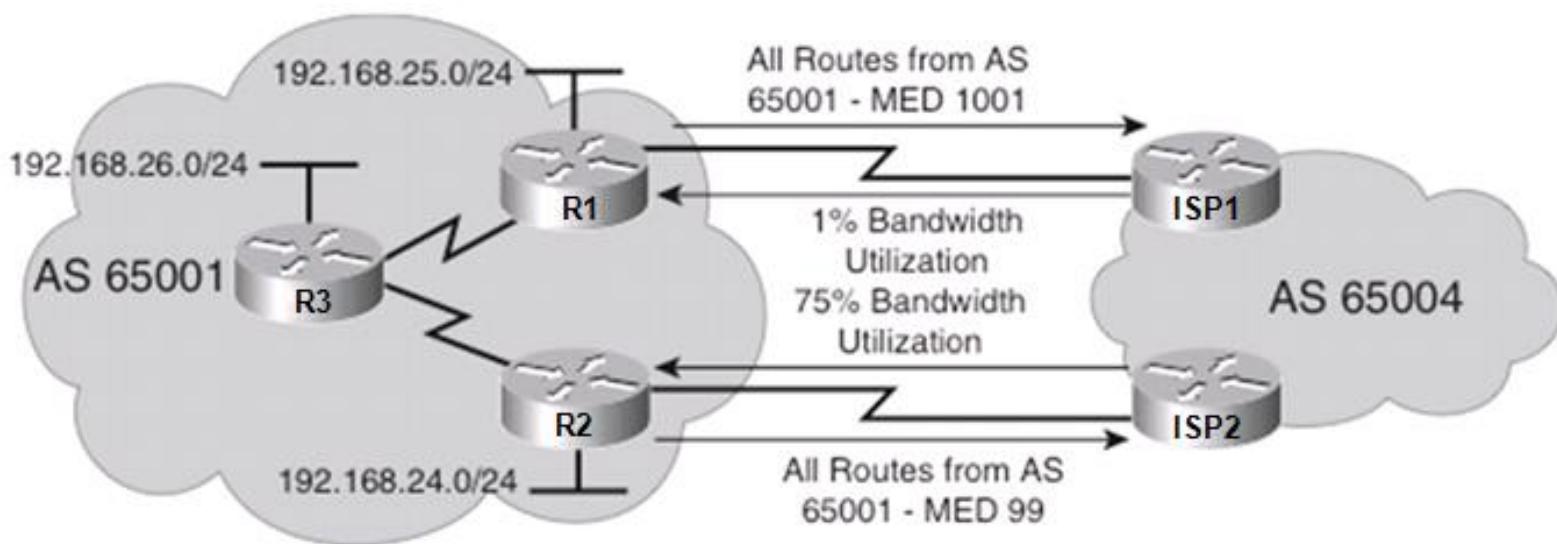
# Setting the MED

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

## BGP Route Selection Process

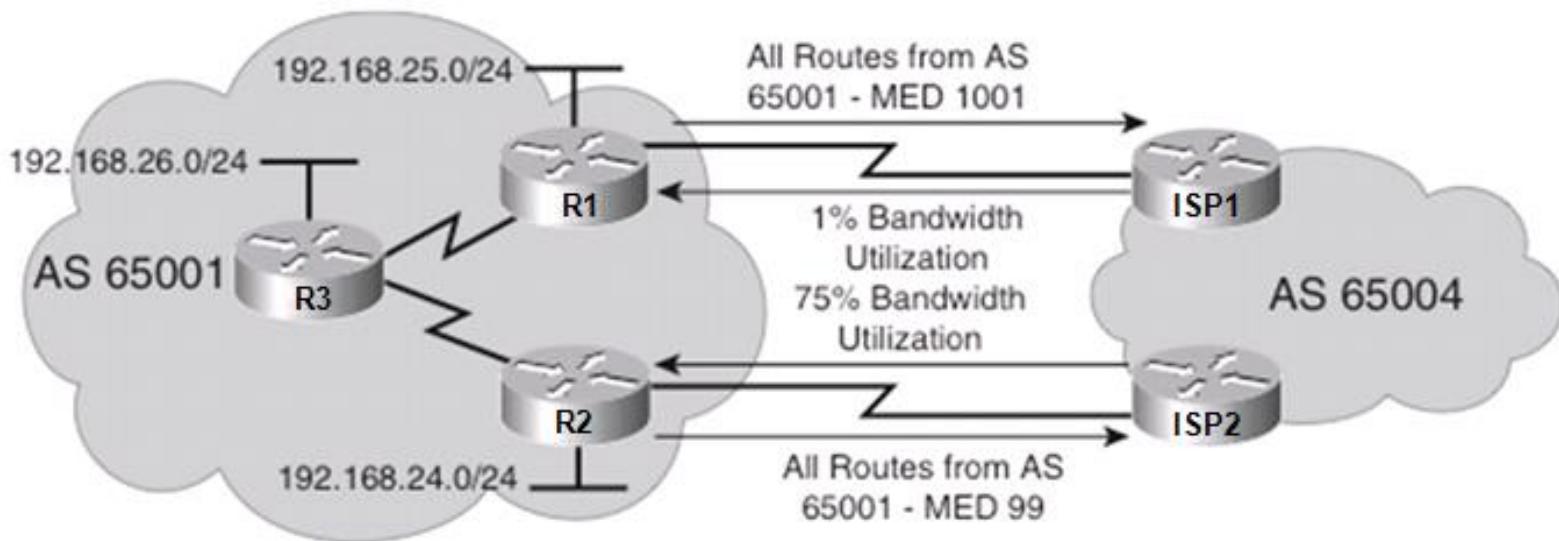
1. Prefer highest Weight
2. Prefer highest LOCAL\_PREF
3. Prefer locally generated routes
4. Prefer shortest AS\_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID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

# Setting the Default MED Example



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

# Setting the Default MED Example



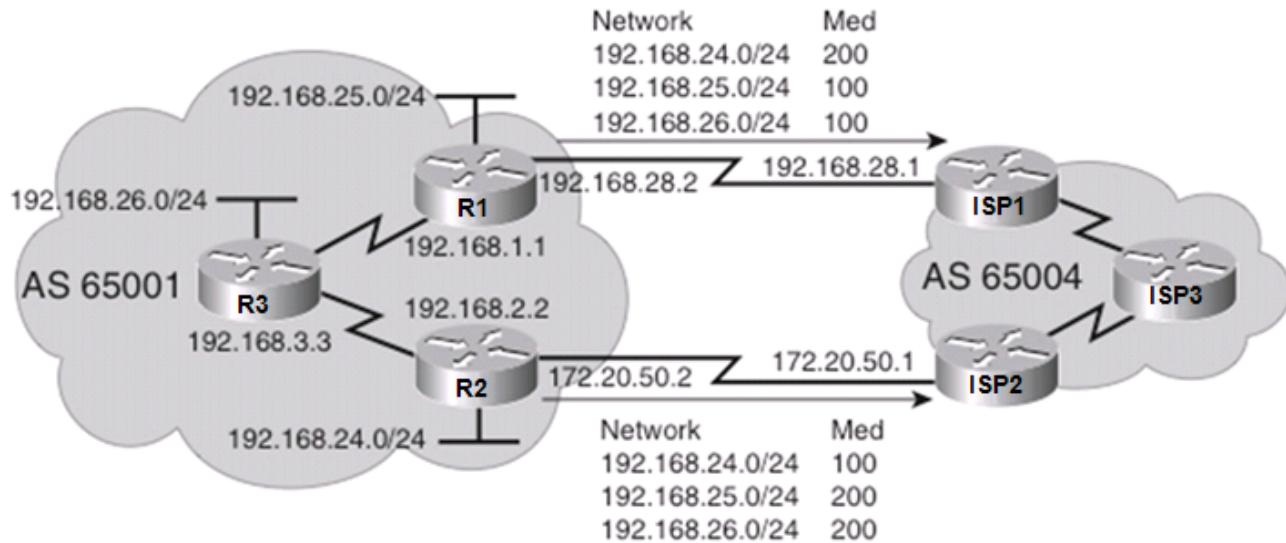
```
R1(config)# router bgp 65001
R1(config-router)# default metric 1001
R1(config-router)#{
```

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

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

# Setting the MED with Route Maps

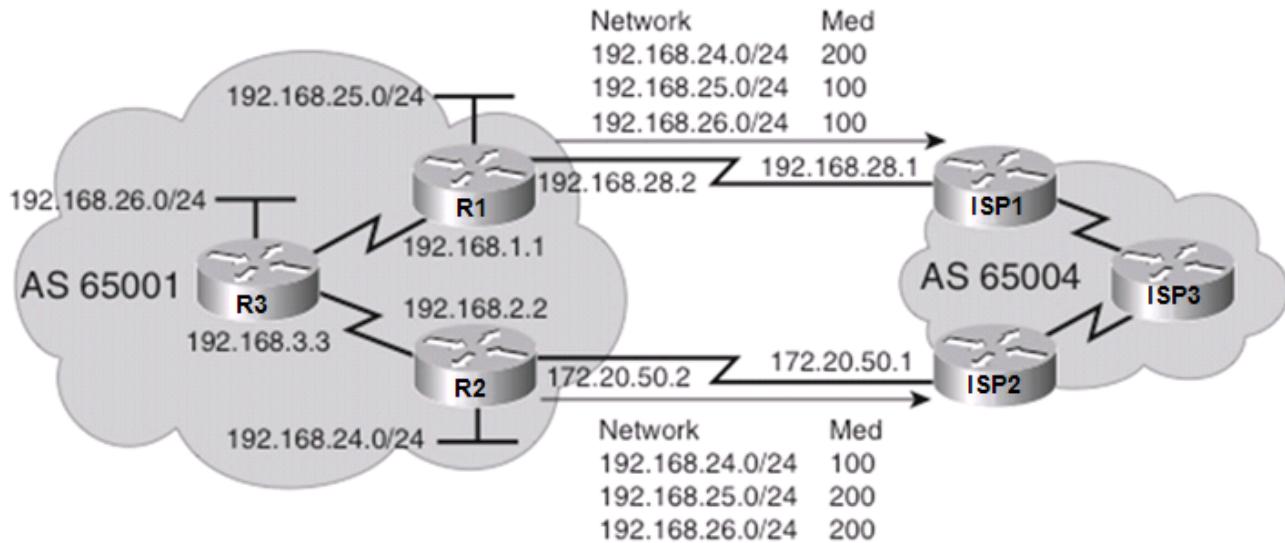
## Example



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

# Setting the MED with Route Maps

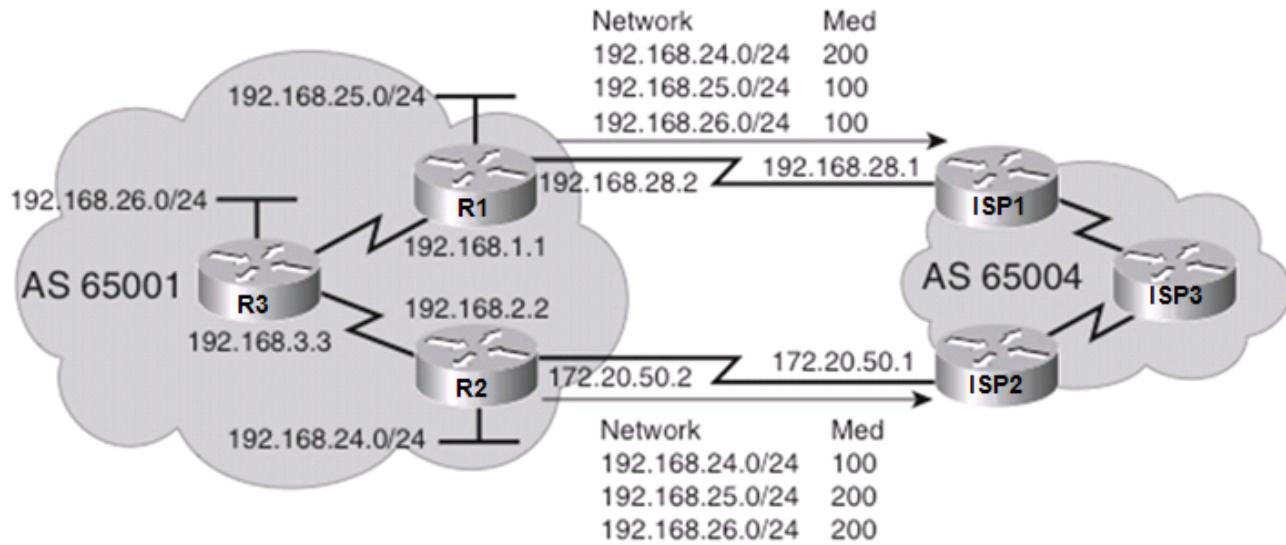
## Example



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

# Setting the MED with Route Maps

## Example

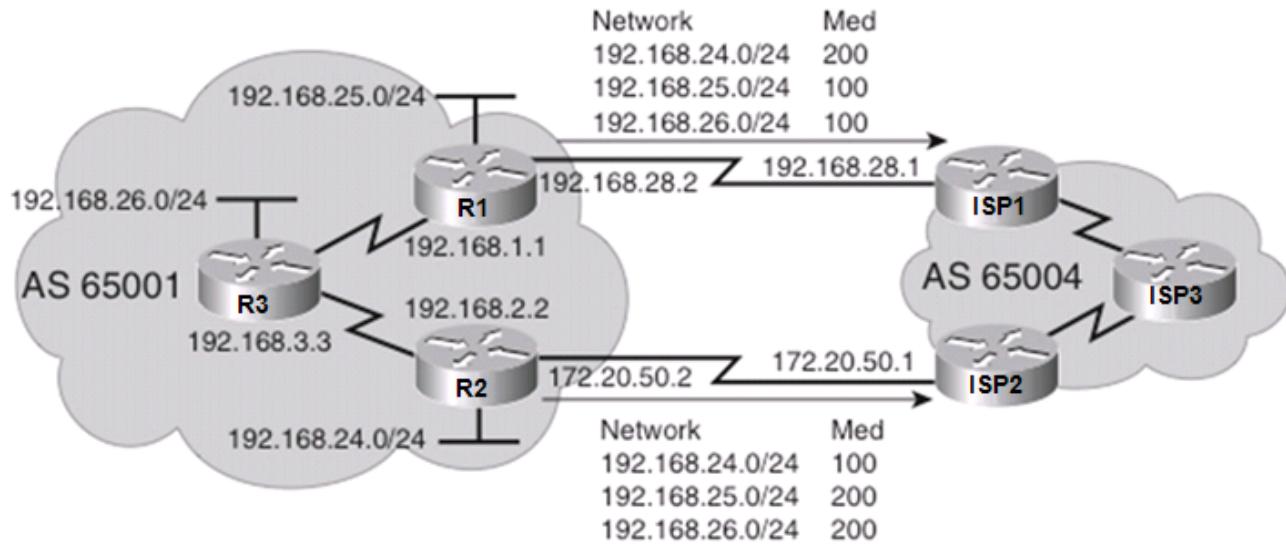


```
R2(config)# access-list 66 permit 192.168.24.0 0.0.0.255
R2(config)#
R2(config)# route-map MED-65004 permit 10
R2(config-route-map)# match ip address 66
R2(config-route-map)# set metric 100
R2(config-route-map)#
R2(config-route-map)# route-map MED-65004 permit 100
R2(config-route-map)# set metric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

```

# Setting the MED with Route Maps

## Example

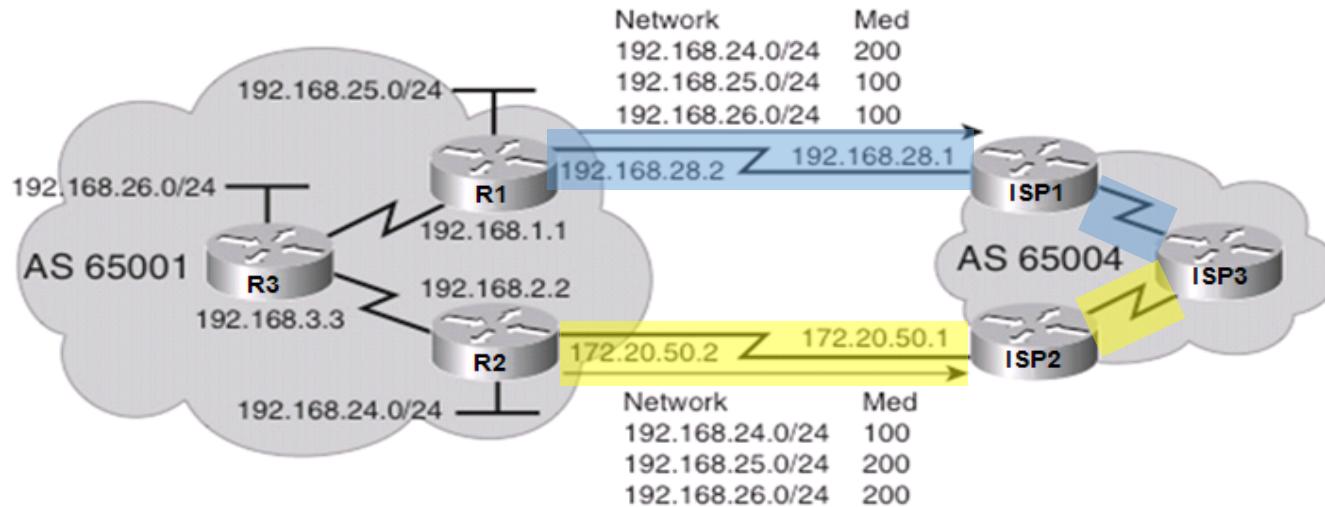


```
R2(config)# router bgp 65001
R2(config-router)# neighbor 192.168.1.1 remote-as 65001
R2(config-router)# neighbor 192.168.1.1 update-source loopback0
R2(config-router)# neighbor 192.168.3.3 remote-as 65001
R2(config-router)# neighbor 192.168.3.3 update-source loopback0
R2(config-router)# neighbor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

```

# Setting the MED with Route Maps

## Example



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

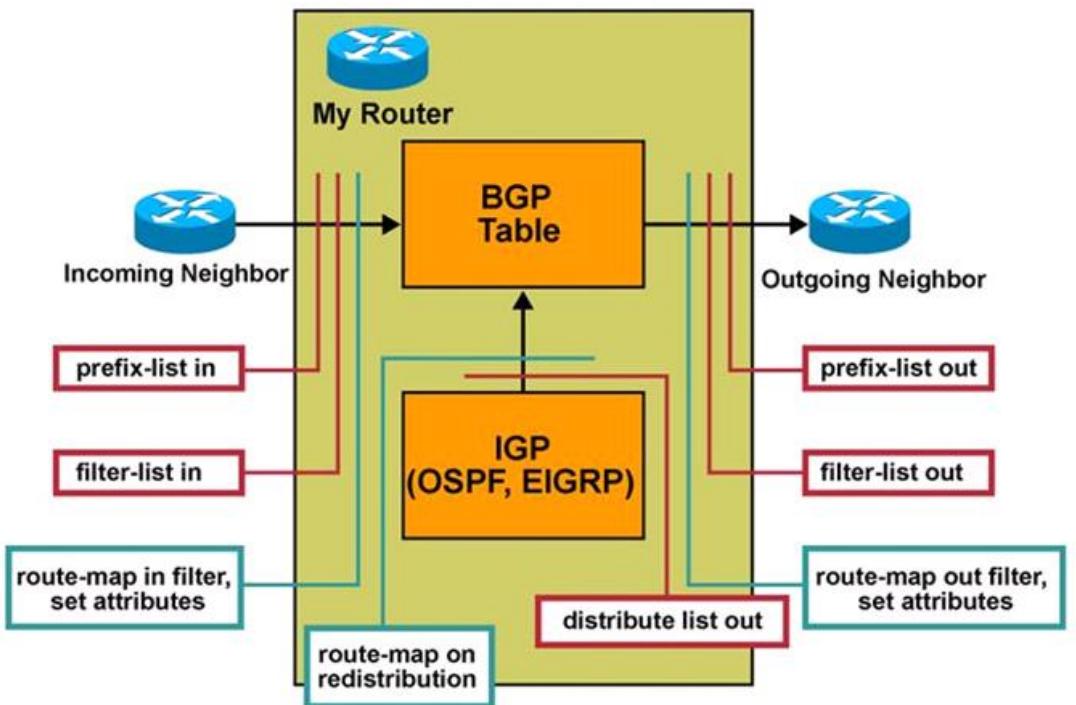
# Filtering BGP Routing Updates

# Filtering BGP Routing Updates

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

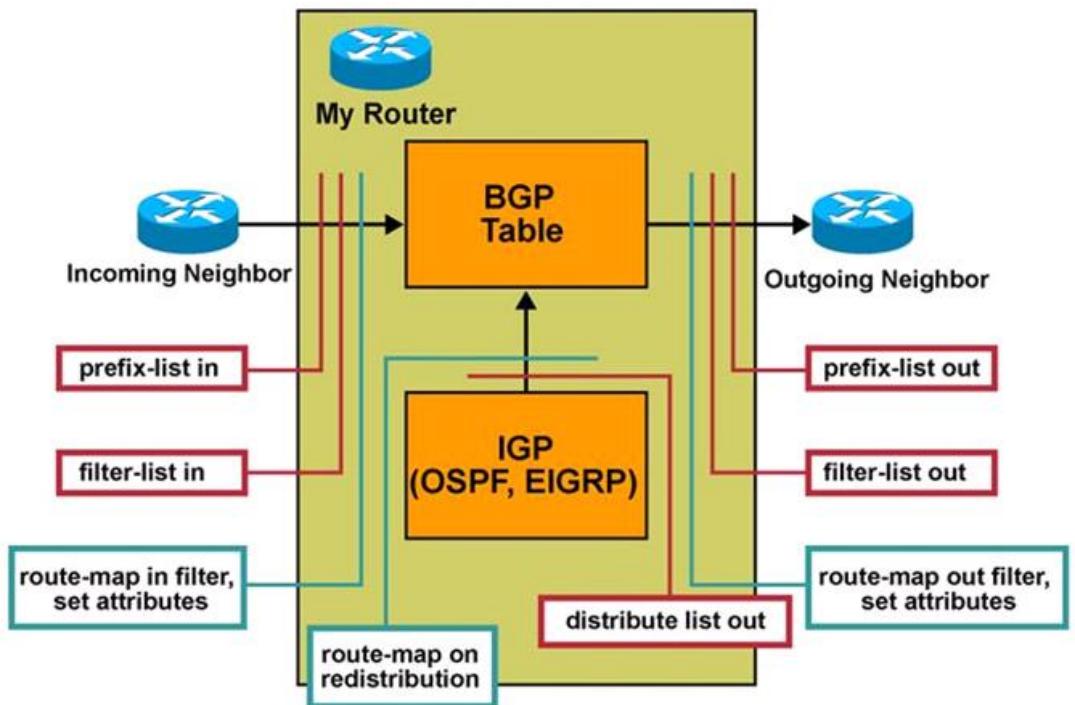
# Filtering BGP Routing Updates

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



# Filtering BGP Routing Updates

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



# Apply a BGP Filter To Routes

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

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

```
neighbor {ip-address | peer-group-name} filter-list
access-list-number {in | out}
```

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

# Planning BGP Filtering Using Prefix Lists

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

# Configure a Prefix List

- Define a prefix list.

```
Router(config) #
```

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

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

# Apply a Prefix List

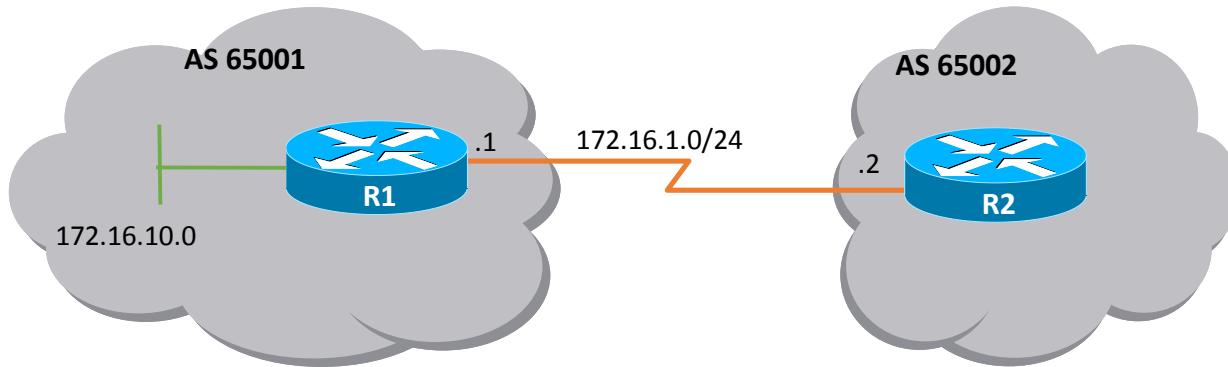
- Apply a prefix list to routes from or to a neighbor.

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

```
neighbor {ip-address | peer-group-name} prefix-list prefix-list-name
{in | out}
```

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

# BGP Filtering Using Prefix Lists Example

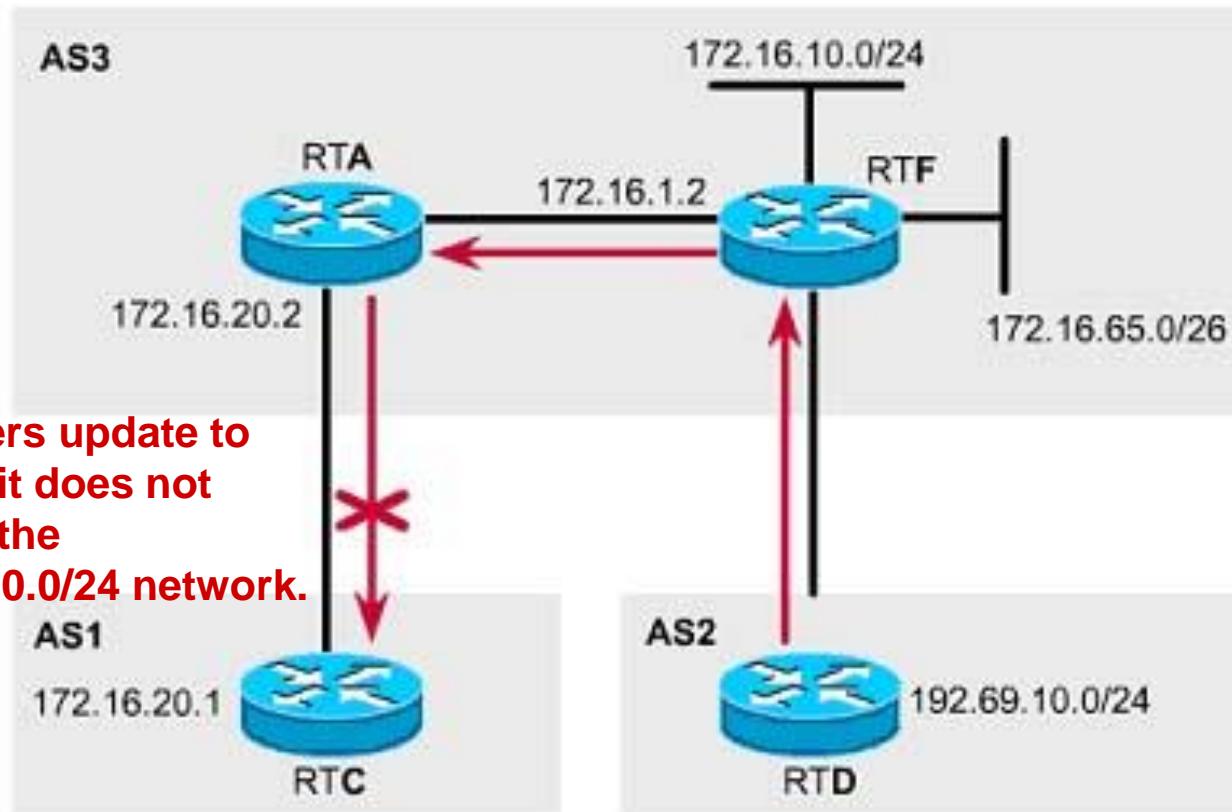


```
R2(config)# ip prefix-list ANY-8to24-NET permit 0.0.0.0/0 ge 8 le 24
R2(config)# router bgp 65001
R2(config-router)# neighbor 172.16.1.2 remote-as 65002
R2(config-router)# neighbor 172.16.1.2 prefix-list ANY-8to24-NET in
R2(config-router)# end
R2#
R2# show ip prefix-list detail ANY-8to24-NET
ip prefix-list ANY-8to24-NET:
Description: test-list
count: 1, range entries: 1, sequences: 10 - 10, refcount: 3
seq 10 permit 0.0.0.0/0 ge 8 le 24 (hit count: 0, refcount: 1)
```

# BGP Route Filtering

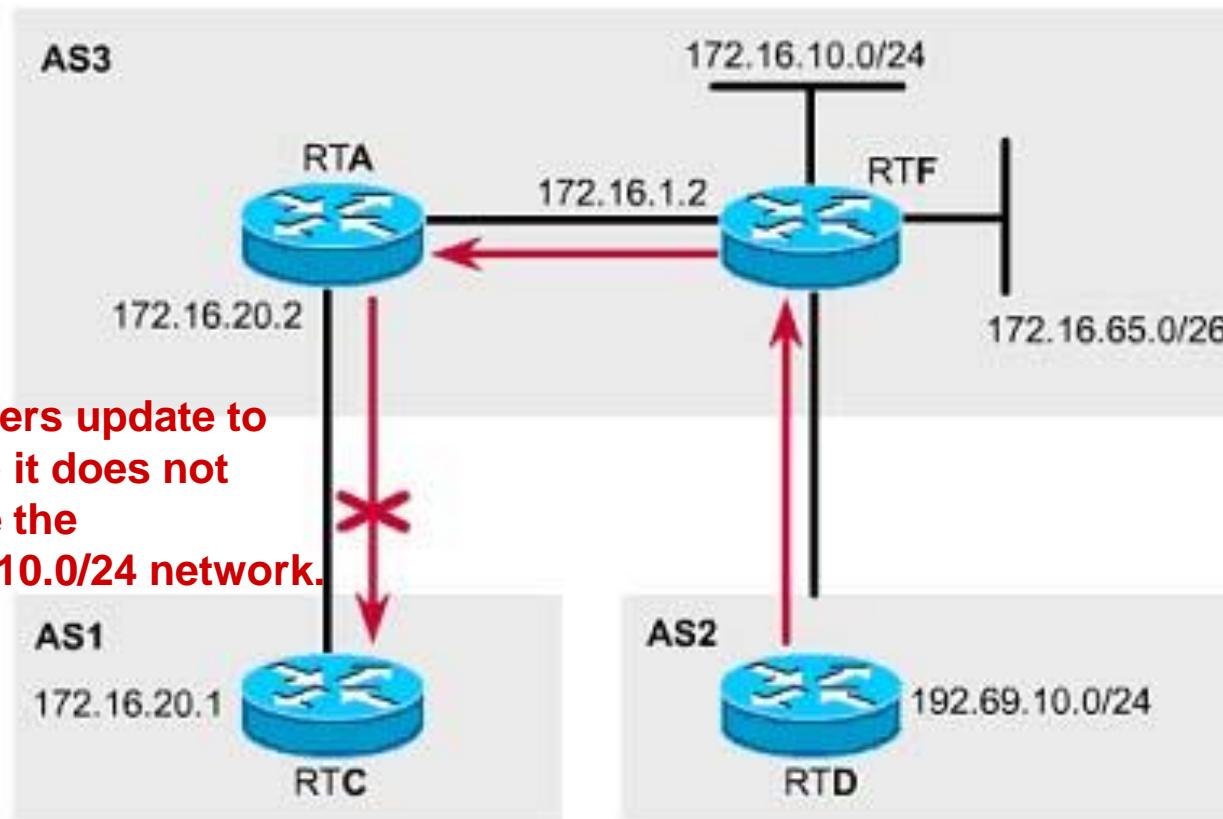
- Route filtering empowers a BGP speaker to **choose what routes to exchange with any of its BGP peers.**
- Route filtering is the cornerstone of policy routing.
  - An AS can identify inbound traffic it is willing to accept by filtering its outbound advertisements
  - An AS can control what routes its outbound traffic uses by specifying the routes to accept from EBGP neighbors
- Even more precise policies can be defined via route filters.
- For example, **BGP routes passing through a filter can have their attributes manipulated to affect the best-path decision process.**
- You can apply route filters to or from a particular neighbor by using the **distribute-list** command.

# BGP Route Filtering



**RTA filters update to RTC so it does not include the 192.69.10.0/24 network.**

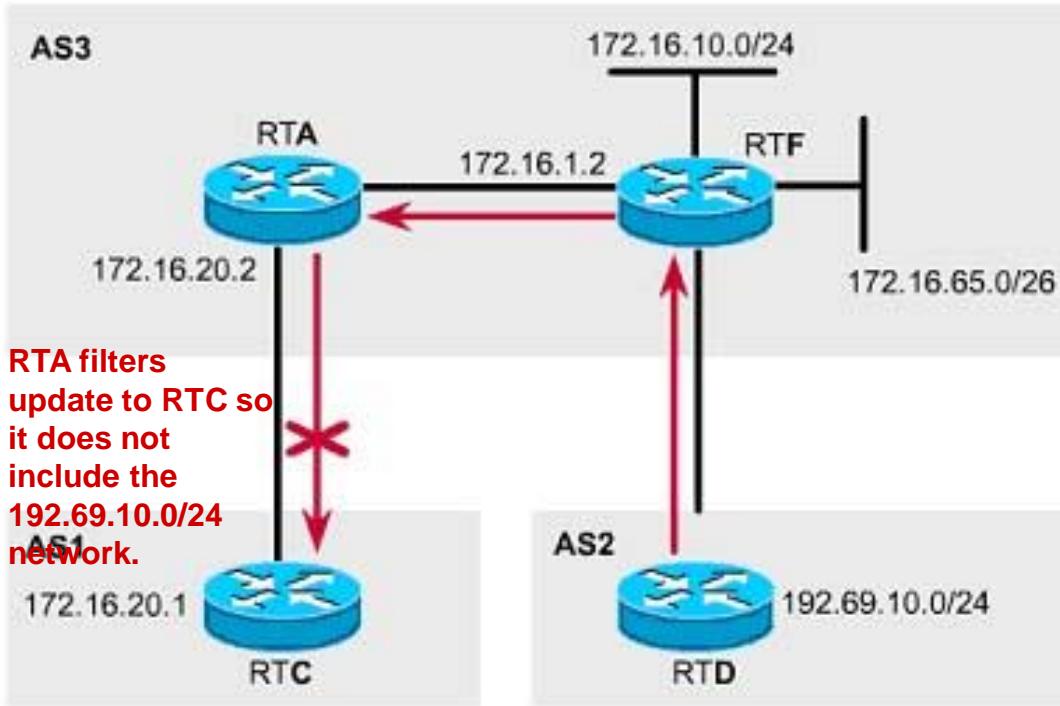
**The distribute-list command can be used to filter updates so that AS1 does not receive transit traffic to network 192.69.10.0 /24.**



```

RTA(config)#router bgp 3
RTA(config-router)#neighbor 172.16.1.2 remote-as 3
RTA(config-router)#neighbor 172.16.20.1 remote-as 1
RTA(config-router)#neighbor 172.16.20.1 distribute-list 1 out
RTA(config-router)#exit
RTA(config)#access-list 1 deny 192.69.10.0 0.0.0.255
RTA(config)#access-list 1 permit any

```



- The **distribute-list** keyword, used as part of a BGP **neighbor** statement, prevents RTA from advertising prefix 192.69.10.0/24 to RTC.
- The access list is used to identify the prefixes to be filtered, and the **distribute-list** and **out** keywords apply the filter to outgoing updates.
- Whereas configuring BGP **neighbor** statements to include the **distribute-list** keyword is effective for filtering specific routes, controlling supernets can be a bit trickier.

```

RTA(config)#router bgp 3
RTA(config-router)#neighbor 172.16.1.2 remote-as 3
RTA(config-router)#neighbor 172.16.20.1 remote-as 1
RTA(config-router)#neighbor 172.16.20.1 distribute-list 1 out
RTA(config-router)#exit
RTA(config)#access-list 1 deny 192.69.10.0 0.0.0.255
RTA(config)#access-list 1 permit any

```

# BGP Route Filtering

- Configuring a distribute list relies on creating an access list.
- If we use a **standard access list**, we are afforded only limited functionality.
- **What if you want to advertise an aggregate address of 172.16.0.0 /16, but not the individual subnets themselves?**
- **A standard access list would not work** because it permits more than is desired, since it filters based on the network address only.
- For example, this access list would permit not only the 172.16.0.0/16 summary, but also all the components of that summary as well:

```
access-list 1 permit 172.16.0.0 0.0.255.255
```

- To restrict the update to the 172.16.0.0/16 summary, you can use an **extended access list**.
- In the case of a BGP route filter, an extended list matches,
  - first, the network address,
  - second, the subnet mask of the prefix.
- Both **network** and **mask** are paired with their own wildcard bitmask, using the following syntax:

```
Router(config) #access-list number permit|deny network

network-wildcard mask mask-wildcard
```

- Using this configuration, RTA would not send a subnet route (such as 172.16.0.0 /17 or 172.16.10.0 /24) in an update to AS1.

```
RTA(config) #router bgp 3

RTA(config-router) #neighbor 172.16.1.2 remote-as 3

RTA(config-router) #neighbor 172.16.20.1 remote-as 1

RTA(config-router) #neighbor 172.16.20.1 distribute-list 101 out

RTA(config-router) #exit

RTA(config) #access-list 101 permit ip 172.16.0.0 0.0.255.255

255.255.0.0 0.0.0.0
```

# BGP Route Filtering - Prefix lists

- If using an extended access list to accomplish this type of filtering seems confusing to you, you are not alone.
- **Improved user-friendliness** was one of the factors that motivated Cisco to include the **ip prefix-list** command in IOS 12.0.
- You can use prefix lists as an alternative to access lists with many BGP route-filtering commands.
- You must define a prefix list before you can apply it as a route filter.
- The Cisco IOS allows a very flexible configuration procedure, where each statement can be assigned its own sequence numbers.
- There is an **implicit deny at the end of each prefix list**.
- To define a prefix list, use the **ip prefix-list command**, which has the following syntax:

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

# BGP Route Filtering - Prefix lists

| Parameter          | Description                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------|
| <i>list-name</i>   | Specifies the name of a prefix list.                                                            |
| <b>seq</b>         | (Optional) Applies the sequence number to the prefix list entry being created or deleted.       |
| <i>seq-value</i>   | (Optional) Specifies the sequence number for the prefix list entry.                             |
| <b>deny</b>        | Denies access to matching conditions.                                                           |
| <b>permit</b>      | Permits access for matching conditions.                                                         |
| <i>network/len</i> | (Mandatory) The network number and length (in bits) of the network mask.                        |
| <b>ge</b>          | (Optional) Applies <i>ge-value</i> to the range specified.                                      |
| <i>ge-value</i>    | (Optional) Specifies the lesser value of a range (the "from" portion of the range description). |
| <b>le</b>          | (Optional) Applies <i>le-value</i> to the range specified.                                      |
| <i>le-value</i>    | (Optional) Specifies the greater value of a range (the "to" portion of the range description).  |

# Example

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

- Restricts the update to the 172.16.0.0/16 summary
- Using this configuration, RTA would not send a subnet route (such as 172.16.0.0 /17 or 172.16.10.0 /24) in an update to AS1.

# BGP Route Filtering - Prefix lists

- The real power of the **ip prefix-list** command is in its optional parameters.
- The keywords **ge** and **le** can be used to specify the range of the **prefix length** to be matched for prefixes that are more specific than the *network/len* value.
- The prefix-length range is assumed to be from *ge-value* to 32 if only the **ge** attribute is specified, and from *len* to *le-value* if only the **le** attribute is specified.
- For example, to accept a mask length of up to 24 bits in routes with the prefix **192.0.0.0/8**, (ie.192.1.0.0/16, 192.2.10.0/24) and deny more specific routes (192.168.10.128/25), use the commands as shown in.

```
RTA(config) #ip prefix-list GROVER permit 192.0.0.0/8 le 24
RTA(config) #ip prefix-list GROVER deny 192.0.0.0/8 ge 25
```

# BGP Route Filtering - Prefix lists

- The **le** and **ge** keywords can be used together, in the same statement:

```
RTA(config)#ip prefix-list OSCAR permit 10.0.0.0/8 ge
16 le 24
```

- This list permits all prefixes in the **10.0.0.0/8 address space** that have a **mask of between 16 and 24 bits**.

**Examples** - The following examples show how a **prefix** list can be used.

- To **deny** the default route 0.0.0.0/0:

```
ip prefix-list abc deny 0.0.0.0/0
```

- To **permit** the prefix 35.0.0.0/8:

```
ip prefix-list abc permit 35.0.0.0/8
```

The following examples show how to specify a group of prefixes.

- To accept a mask length of up to 24 bits in routes with the prefix 192/8:

```
ip prefix-list abc permit 192.0.0.0/8 le 24
```

- To **deny** mask lengths greater than 25 bits in routes with a prefix of 192/8:

```
ip prefix-list abc deny 192.0.0.0/8 ge 25
```

- To **permit** mask lengths from 8 to 24 bits in all address space:

```
ip prefix-list abc permit 0.0.0.0/0 ge 8 le 24
```

- To **deny** mask lengths greater than 25 bits in all address space:

```
ip prefix-list abc deny 0.0.0.0/0 ge 25
```

# BGP Route Filtering - Prefix lists

- Each prefix list entry is assigned a sequence number, either by default or manually by an administrator.
- By numbering the prefix list statements, new entries can be inserted at any point in the list, which is important because routers test for prefix list matches from lowest sequence number to highest.
- By default, the entries of a prefix-list will have sequence values of 5,10, 15, etc.
- To disable this: RTR(config)# no ip prefix-list sequence-number
- Sequence numbers can be created using the command:

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

```
RTA#show ip prefix-list
ip prefix-list ELMO: 3 entries
 seq 5 deny 0.0.0.0/0
 seq 10 permit 172.16.0.0/16
 seq 15 permit 192.168.0.0/16 le 24
```

# Default Routes

- It is important to control default information in BGP because improper configuration can cause serious Internet routing problems.
- For example, a misconfigured BGP speaker could end up flooding a default route to all of its neighbors and quickly find itself consumed with default routed traffic from surrounding autonomous systems.
- To protect against misadvertisements, the Cisco IOS provides a way to target default information at a specific neighbor by using the **default-originate** option with the **neighbor** command:

```
RTC(config)#router bgp 3
```

```
RTC(config-router)#neighbor 172.16.20.1 remote-as 1
```

```
RTC(config-router)#neighbor 172.16.20.1 default-originate
```

- If RTC is configured as shown in this configuration, it will send default information only to the specified neighbor.

# Default Routes

- If a BGP router is to be configured to advertise a default to all of its peers, use the **network** command shown as follows:
- **Note:** Both neighbors, 172.16.20.1 and 172.17.1.1, will receive a default route from RTC.

```
RTC(config)#router bgp 3
RTC(config-router)#neighbor 172.16.20.1 remote-as 1
RTC(config-router)#neighbor 172.17.1.1 remote-as 2
RTC(config-router)#network 0.0.0.0
```

- Many network administrators choose to filter dynamically learned default routes to avoid situations in which traffic ends up where it is not supposed to be.
- Without dynamically learned default routes, a router must be statically configured with default information.
- Statically configured default routes typically provide more control over routing within an AS.

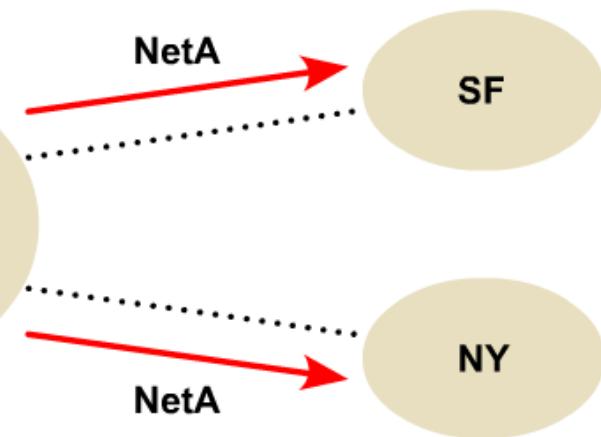
# Symmetry

- Symmetry is achieved when traffic leaving the AS from one exit point comes back through the same point.
- Symmetry always exists if an AS maintains a single connection to outside networks.
- However, the need for redundancy often results in multihoming an AS. If an AS has many different links to the outside world, traffic tends to flow asymmetrically.
- An asymmetrical traffic flow can result in increased delay and other routing problems.
- In general, customers and providers would like to see their traffic come back by way of the same point or close to the same point that it left the AS.
- To promote symmetry, choose a primary path and configure routing policies that force traffic to flow along this path.
- A default route with a low administrative distance or a high Local Preference might serve to control the flow of outbound traffic, but inbound traffic can be more complex to manipulate.
- Through appropriate planning and use of BGP attributes and route filters, an AS can control which paths the outside world finds most desirable.

# Load Balancing

## Outbound Decision

Which way should the outbound traffic go to reach NetA, by way of SF or NY?



- Load balancing is the capability to divide data traffic over multiple connections.
- A BGP speaker may learn two identical EBGP paths for a prefix from a neighboring AS.
- If this happens, it will choose the path with the lowest route ID as the best path.
- This best path is installed in the IP routing table.
- To enable BGP load balancing over equal cost paths, use the **maximum-paths** command, which has the following syntax:

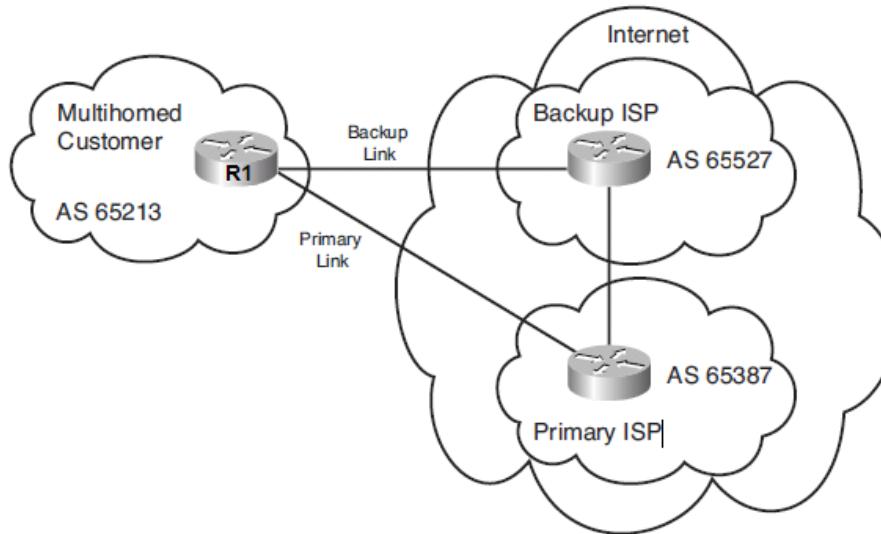
```
Router (config-router) # maximum-paths number
```

- BGP supports a maximum of six paths per destination, but only if they are sourced from the same AS.
- By default, BGP will install only one path to the IP routing table.

# Planning BGP Filtering Using Route Maps

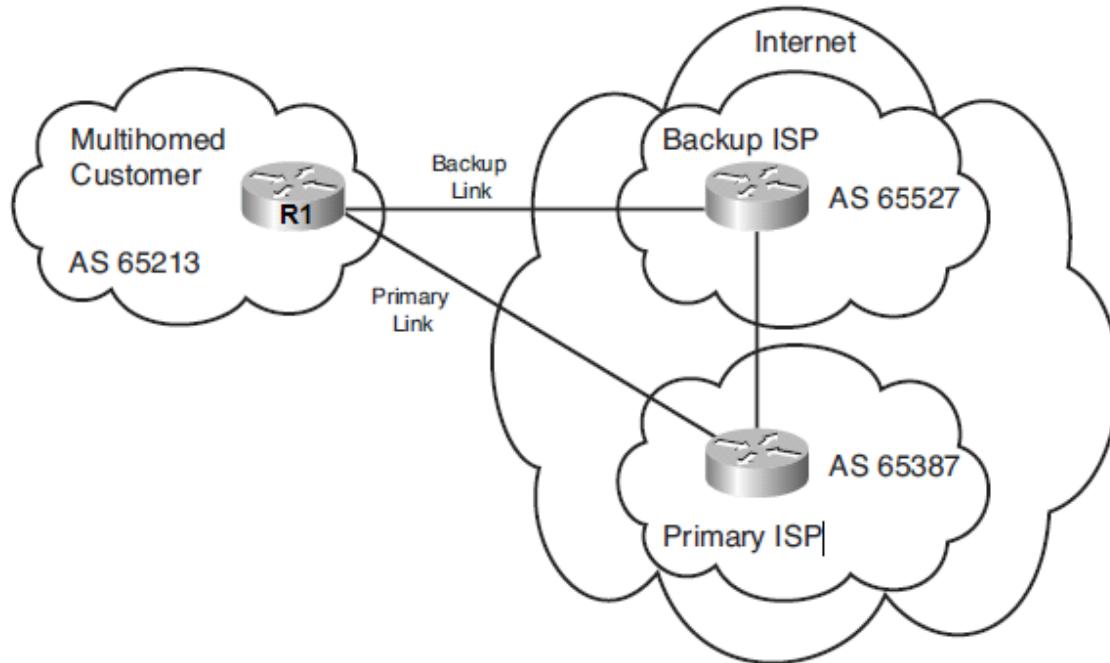
- When planning BGP filter configuration using route maps, the following steps should be documented:
  - Define the route map, including:
    - The **match** statements
    - The **set** statements
  - Configure route filtering using the route map.

# BGP Filtering Using Route Maps



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

# BGP Filtering Using Route Maps



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

```

# BGP4 Case Studies/Tutorial

**Sam Halabi-cisco Systems**

The purpose of this paper is to introduce the reader to the latest in BGP4 terminology and design issues. It is targeted to the novice as well as the experienced user. For any clarification or comments please send e-mail to [shalabi@cisco.com](mailto:shalabi@cisco.com).

Copyright 1995 ©Cisco Systems Inc.

|        |                                         |    |
|--------|-----------------------------------------|----|
| 1.0    | Introduction.....                       | 4  |
| 1.1    | How does BGP work .....                 | 4  |
| 1.2    | What are peers (neighbors) .....        | 4  |
| 1.3    | Information exchange between peers..... | 4  |
| 2.0    | EBGP and IBGP .....                     | 5  |
| 3.0    | Enabling BGP routing.....               | 6  |
| 3.1    | BGP Neighbors/Peers .....               | 7  |
| 4.0    | BGP and Loopback interfaces .....       | 10 |
| 5.0    | EBGP Multihop .....                     | 11 |
| 5.1    | EBGP Multihop (Load Balancing) .....    | 12 |
| 6.0    | Route Maps .....                        | 13 |
| 7.0    | Network command.....                    | 17 |
| 7.1    | Redistribution.....                     | 18 |
| 7.2    | Static routes and redistribution .....  | 20 |
| 8.0    | Internal BGP .....                      | 22 |
| 9.0    | The BGP decision algorithm.....         | 23 |
| 10.0   | As_path Attribute.....                  | 24 |
| 11.0   | Origin Attribute.....                   | 25 |
| 12.0   | BGP Nexthop Attribute.....              | 27 |
| 12.1   | BGP Nexthop (Multiaccess Networks)..... | 29 |
| 12.2   | BGP Nexthop (NBMA) .....                | 30 |
| 12.3   | Next-hop-self .....                     | 31 |
| 13.0   | BGP Backdoor .....                      | 32 |
| 14.0   | Synchronization .....                   | 34 |
| 14.1   | Disabling synchronization .....         | 35 |
| 15.0   | Weight Attribute.....                   | 37 |
| 16.0   | Local Preference Attribute.....         | 39 |
| 17.0   | Metric Attribute .....                  | 41 |
| 18.0   | Community Attribute .....               | 44 |
| 19.0   | BGP Filtering .....                     | 45 |
| 19.1   | Route Filtering .....                   | 45 |
| 19.2   | Path Filtering.....                     | 47 |
| 19.2.1 | AS-Regular Expression .....             | 49 |
| 19.3   | BGP Community Filtering .....           | 50 |
| 20.0   | BGP Neighbors and Route maps .....      | 53 |
| 20.1   | Use of set as-path prepend .....        | 55 |
| 20.2   | BGP Peer Groups.....                    | 56 |
| 21.0   | CIDR and Aggregate Addresses .....      | 58 |

|      |                                              |    |
|------|----------------------------------------------|----|
| 21.1 | Aggregate Commands.....                      | 59 |
| 21.2 | CIDR example 1 .....                         | 61 |
| 21.3 | CIDR example 2 (as-set).....                 | 63 |
| 22.0 | BGP Confederation.....                       | 65 |
| 23.0 | Route Reflectors.....                        | 68 |
| 23.1 | Multiple RRs within a cluster .....          | 71 |
| 23.2 | RR and conventional BGP speakers .....       | 73 |
| 23.3 | Avoiding looping of routing information..... | 74 |
| 24.0 | Route Flap Dampening .....                   | 75 |
| 25.0 | How BGP selects a Path .....                 | 79 |
| 26.0 | Practical design example: .....              | 80 |

# **1.0 Introduction**

The Border Gateway Protocol (BGP), defined in RFC 1771, allows you to create loop free interdomain routing between autonomous systems. An autonomous system is a set of routers under a single technical administration. Routers in an AS can use multiple interior gateway protocols to exchange routing information inside the AS and an exterior gateway protocol to route packets outside the AS.

## **1.1 How does BGP work**

BGP uses TCP as its transport protocol (port 179). Two BGP speaking routers form a TCP connection between one another (peer routers) and exchange messages to open and confirm the connection parameters.

BGP routers will exchange network reachability information, this information is mainly an indication of the full paths (BGP AS numbers) that a route should take in order to reach the destination network. This information will help in constructing a graph of ASs that are loop free and where routing policies can be applied in order to enforce some restrictions on the routing behavior.

## **1.2 What are peers (neighbors)**

Any two routers that have formed a TCP connection in order to exchange BGP routing information are called peers, they are also called neighbors.

## **1.3 Information exchange between peers**

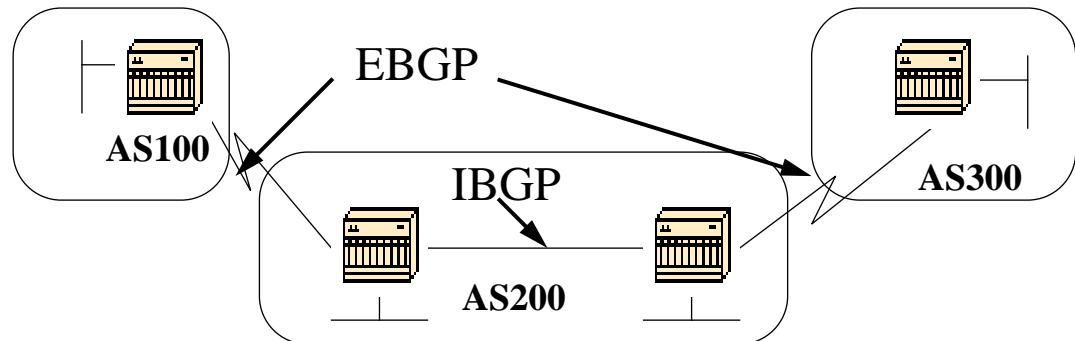
BGP peers will initially exchange their full BGP routing tables. From then on incremental updates are sent as the routing table changes. BGP keeps a version number of the BGP table and it should be the same for all of its BGP peers. The version number will change whenever BGP updates the table due to some routing information changes. Keepalive packets are sent to ensure that the connection is alive between the BGP peers and notification packets are sent in response to errors or special conditions.

## 2.0 EBGP and IBGP

If an Autonomous System has multiple BGP speakers, it could be used as a transit service for other ASs. As you see below, AS200 is a transit autonomous system for AS100 and AS300.

It is necessary to ensure reachability for networks within an AS before sending the information to other external ASs. This is done by a combination of Internal BGP peering between routers inside an AS and by redistributing BGP information to Internal Gateway protocols running in the AS.

As far as this paper is concerned, when BGP is running between routers belonging to two different ASs we will call it EBGP (Exterior BGP) and for BGP running between routers in the same AS we will call it IBGP (Interior BGP).



## 3.0 Enabling BGP routing

Here are the steps needed to enable and configure BGP.

Let us assume you want to have two routers RTA and RTB talk BGP. In the first example RTA and RTB are in different autonomous systems and in the second example both routers belong to the same AS.

We start by defining the router process and define the AS number that the routers belong to:

The command used to enable BGP on a router is:

```
router bgp autonomous-system
```

```
RTA#
router bgp 100
```

```
RTB#
router bgp 200
```

The above statements indicate that RTA is running BGP and it belongs to AS100 and RTB is running BGP and it belongs to AS200 and so on.

The next step in the configuration process is to define BGP neighbors. The neighbor definition indicates which routers we are trying to talk to with BGP.

The next section will introduce you to what is involved in forming a valid peer connection.

### 3.1 BGP Neighbors/Peers

Two BGP routers become neighbors or peers once they establish a TCP connection between one another. The TCP connection is essential in order for the two peer routers to start exchanging routing updates.

Two BGP speaking routers trying to become neighbors will first bring up the TCP connection between one another and then send open messages in order to exchange values such as the AS number, the BGP version they are running (version 3 or 4), the BGP router ID and the keepalive hold time, etc. After these values are confirmed and accepted the neighbor connection will be established. Any state other than established is an indication that the two routers did not become neighbors and hence the BGP updates will not be exchanged.

The neighbor command used to establish a TCP connection is:

```
neighbor ip-address remote-as number
```

The remote-as number is the AS number of the router we are trying to connect to via BGP.

The ip-address is the next hop directly connected address for EBGP<sup>1</sup> and any IP address<sup>2</sup> on the other router for IBGP.

It is essential that the two IP addresses used in the neighbor command of the peer routers be able to reach one another. One sure way to verify reachability is an extended ping between the two IP addresses, the extended ping forces the pinging router to use as source the IP address specified in the neighbor command rather than the IP address of the interface the packet is going out from.

---

1.A special case (EBGP multihop) will be discussed later when the external BGP peers are not directly connected.

2.A special case for loopback interfaces is discussed later.

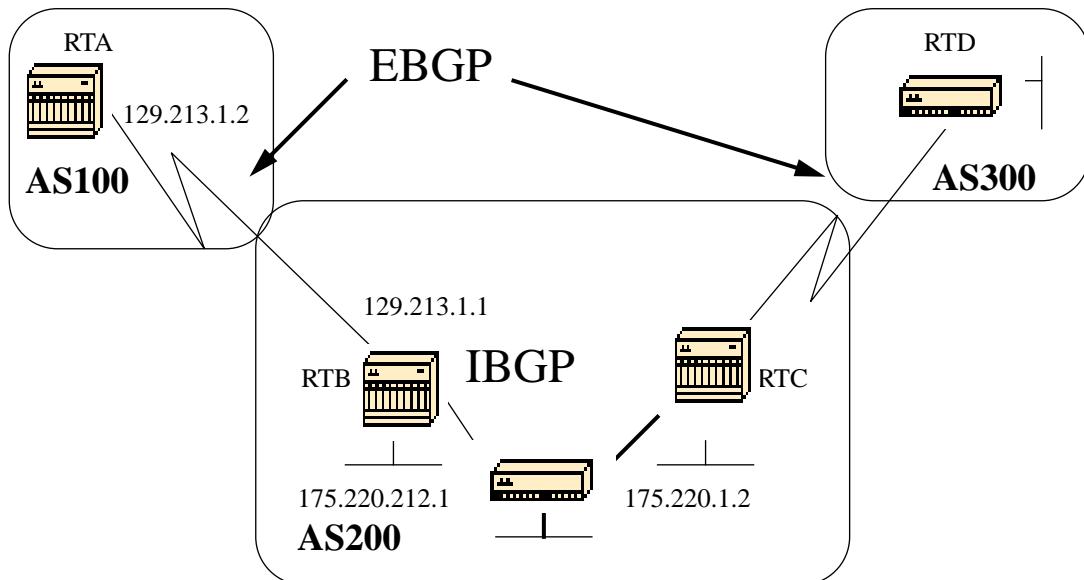
It is important to reset the neighbor connection in case any bgp configuration changes are made in order for the new parameters to take effect.

```
clear ip bgp address (where address is the neighbor address)
clear ip bgp * (clear all neighbor connections)
```

By default, BGP sessions begin using BGP Version 4 and negotiating downward to earlier versions if necessary. To prevent negotiations and force the BGP version used to communicate with a neighbor, perform the following task in router configuration mode:

```
neighbor {ip address|peer-group-name} version value
```

An example of the neighbor command configuration follows:



```
RTA#
router bgp 100
neighbor 129.213.1.1 remote-as 200
```

```
RTB#
router bgp 200
neighbor 129.213.1.2 remote-as 100
neighbor 175.220.1.2 remote-as 200
```

```
RTC#
router bgp 200
neighbor 175.220.212.1 remote-as 200
```

In the above example RTA and RTB are running EBGP. RTB and RTC are running IBGP. The difference between EBGP and IBGP is manifested by having the remote-as number pointing to either an external or an internal AS.

Also, the EBGP peers are directly connected and the IBGP peers are not. IBGP routers do not have to be directly connected, as long as there is some IGP running that allows the two neighbors to reach one another.

The following is an example of the information that the command "sh ip bgp neighbors" will show you, pay special attention to the BGP state. Anything other than state established indicates that the peers are not up. You should also note the BGP is version 4, the remote router ID (highest IP address on that box or the highest loopback interface in case it exists) and the table version (this is the state of the table. Any time new information comes in, the table will increase the version and a version that keeps incrementing indicates that some route is flapping causing routes to keep getting updated).

```
#SH IP BGP N

BGP neighbor is 129.213.1.1, remote AS 200, external link
 BGP version 4, remote router ID 175.220.212.1
BGP state = Established, table version = 3, up for 0:10:59
 Last read 0:00:29, hold time is 180, keepalive interval is 60 seconds
 Minimum time between advertisement runs is 30 seconds
 Received 2828 messages, 0 notifications, 0 in queue
 Sent 2826 messages, 0 notifications, 0 in queue
 Connections established 11; dropped 10
```

In the next section we will discuss special situations such as EBGP multihop and loopback addresses.

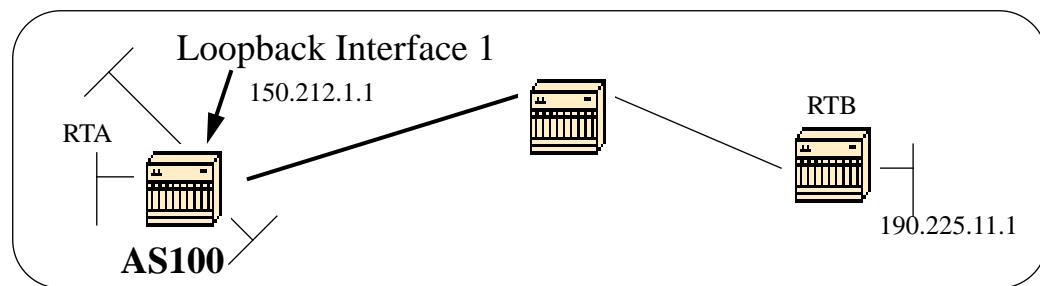
## 4.0 BGP and Loopback interfaces

Using a loopback interface to define neighbors is commonly used with IBGP rather than EBGP. Normally the loopback interface is used to make sure that the IP address of the neighbor stays up and is independent of an interface that might be flaky. In the case of EBGP, most of the time the peer routers are directly connected and loopback does not apply.

If the IP address of a loopback interface is used in the neighbor command, some extra configuration needs to be done on the neighbor router. The neighbor router needs to tell BGP that it is using a loopback interface rather than a physical interface to initiate the BGP neighbor TCP connection. The command used to indicate a loopback interface is:

```
neighbor ip-address update-source interface
```

The following example should illustrate the use of this command.



```
RTA#
router bgp 100
neighbor 190.225.11.1 remote-as 100
neighbor 190.225.11.1 update-source int loopback 1
```

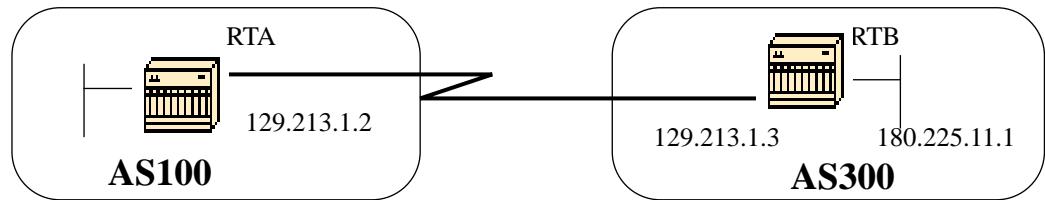
```
RTB#
router bgp 100
neighbor 150.212.1.1 remote-as 100
```

In the above example, RTA and RTB are running internal BGP inside autonomous system 100. RTB is using in its neighbor command the loopback interface of RTA (150.212.1.1); in this case RTA has to force BGP to use the loopback IP address as the source in the TCP neighbor connection. RTA will do so by adding the update-source int loopback configuration (neighbor 190.225.11.1 update-source int loopback 1) and this statement forces BGP to use the IP address of its loopback interface when talking to neighbor 190.225.11.1.

Note that RTA has used the physical interface IP address (190.225.11.1) of RTB as a neighbor and that is why RTB does not need to do any special configuration.

## 5.0 EBGP Multihop

In some special cases, there could be a requirement for EBGP speakers to be not directly connected. In this case EBGP multihop is used to allow the neighbor connection to be established between two non directly connected external peers. **The multihop is used only for external BGP and not for internal BGP.** The following example gives a better illustration of EBGP multihop.



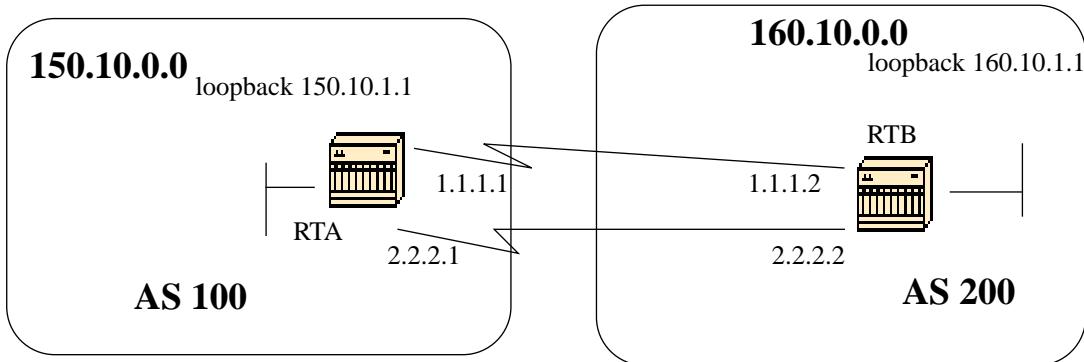
```
RTA#
router bgp 100
neighbor 180.225.11.1 remote-as 300
neighbor 180.225.11.1 ebgp-multihop
```

```
RTB#
router bgp 300
neighbor 129.213.1.2 remote-as 100
```

RTA is indicating an external neighbor that is not directly connected. RTA needs to indicate that it will be using ebgp-multihop. On the other hand, RTB is indicating a neighbor that is directly connected (129.213.1.2) and that is why it does not need the ebgp-multihop command. Some IGP or static routing should also be configured in order to allow the non directly connected neighbors to reach one another.

The following example shows how to achieve load balancing with BGP in a particular case where we have EBGP over parallel lines.

## 5.1 EBGP Multihop (Load Balancing)



```
RTA#
int loopback 0
ip address 150.10.1.1 255.255.255.0

router bgp 100
neighbor 160.10.1.1 remote-as 200
neighbor 160.10.1.1 ebgp-multihop
neighbor 160.10.1.1 update-source loopback 0
network 150.10.0.0

ip route 160.10.0.0 255.255.0.0 1.1.1.2
ip route 160.10.0.0 255.255.0.0 2.2.2.2
```

```
RTB#
int loopback 0
ip address 160.10.1.1 255.255.255.0

router bgp 200
neighbor 150.10.1.1 remote-as 100
neighbor 150.10.1.1 update-source loopback 0
neighbor 150.10.1.1 ebgp-multihop
network 160.10.0.0
```

```
ip route 150.10.0.0 255.255.0.0 1.1.1.1
ip route 150.10.0.0 255.255.0.0 2.2.2.1
```

The above example illustrates the use of loopback interfaces, update-source and ebgp-multihop. This is a workaround in order to achieve load balancing between two EBGP speakers over parallel serial lines. In normal situations, BGP will pick one of the lines to send packets on and load balancing would not take place. By introducing loopback interfaces, the next hop for EBGP will be the loopback interface. Static routes (it could be some IGP also) are used to introduce two equal cost paths to reach the destination. RTA will have two choices to reach next hop 160.10.1.1: one via 1.1.1.2 and the other one via 2.2.2.2 and the same for RTB.

## 6.0 Route Maps

At this point I would like to introduce route maps because they will be used heavily with BGP. In the BGP context, route map is a method used to control and modify routing information. This is done by defining conditions for redistributing routes from one routing protocol to another or controlling routing information when injected in and out of BGP. The format of the route map follows:

```
route-map map-tag [[permit | deny] | [sequence-number]]
```

The map-tag is just a name you give to the route-map. Multiple instances of the same route map (same name-tag) can be defined. The sequence number is just an indication of the position a new route map is to have in the list of route maps already configured with the same name.

For example, if I define two instances of the route map, let us call it MYMAP, the first instance will have a sequence-number of 10, and the second will have a sequence number of 20.

```
route-map MYMAP permit 10
(first set of conditions goes here.)
```

```
route-map MYMAP permit 20
(second set of conditions goes here.)
```

When applying route map MYMAP to incoming or outgoing routes, the first set of conditions will be applied via instance 10. If the first set of conditions is not met then we proceed to a higher instance of the route map.

The conditions that we talked about are defined by the **match** and **set** configuration commands. Each route map will consist of a list of match and set configuration. The match will specify a **match** criteria and set specifies a **set** action if the criteria enforced by the match command are met.

For example, I could define a route map that checks outgoing updates and if there is a match for IP address 1.1.1.1 then the metric for that update will be set to 5. The above can be illustrated by the following commands:

```
match ip address 1.1.1.1
set metric 5
```

Now, if the match criteria are met and we have a **permit** then the routes will be redistributed or controlled as specified by the set action and we break out of the list.

If the match criteria are met and we have a **deny** then the route will not be redistributed or controlled and we break out of the list.

If the match criteria are not met and we have a **permit** or **deny** then the next instance of the route map (instance 20 for example) will be checked, and so on until we either break out or finish all the instances of the route map. If we finish the list without a match then the route we are looking at will **not be accepted nor forwarded**.

One restriction on route maps is that when used for filtering BGP updates (as we will see later) rather than when redistributing between protocols, you can NOT filter on the inbound when using a "match" on the ip address. Filtering on the outbound is OK.

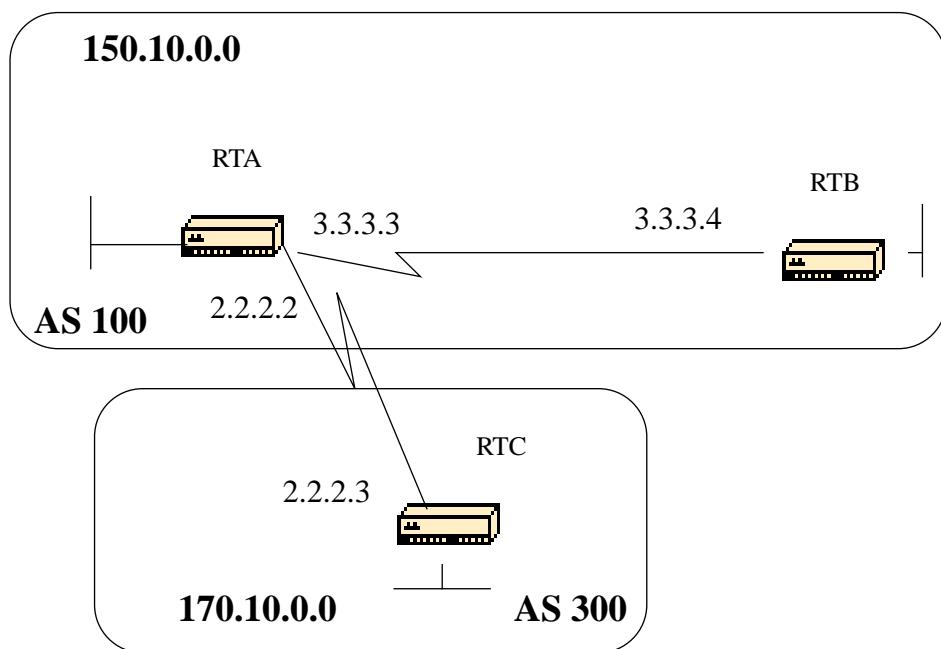
The related commands for **match** are:

```
match as-path
match community
match clns
match interface
match ip address
match ip next-hop
match ip route-source
match metric
match route-type
match tag
```

The related commands for **set** are:

```
set as-path
set automatic-tag
set community
set clns
set interface
set default interface
set ip next-hop
set ip default next-hop
set ip precedence
set tos
set level
set local-preference
set metric
set metric-type
set next-hop
set origin
set tag
set weight
```

Let's look at some route-map examples:



Example 1:

Assume RTA and RTB are running rip; RTA and RTC are running BGP.  
 RTA is getting updates via BGP and redistributing them to rip.  
 If RTA wants to redistribute to RTB routes about 170.10.0.0 with a metric of 2 and all other routes with a metric of 5 then we might use the following configuration:

```

RTA#
router rip
network 3.0.0.0
network 2.0.0.0
network 150.10.0.0
passive-interface Serial0
redistribute bgp 100 route-map SETMETRIC

router bgp 100
neighbor 2.2.2.3 remote-as 300
network 150.10.0.0

route-map SETMETRIC permit 10
match ip-address 1
set metric 2

route-map SETMETRIC permit 20
set metric 5

access-list 1 permit 170.10.0.0 0.0.255.255

```

In the above example if a route matches the IP address 170.10.0.0 it will have a metric of 2 and then we break out of the route map list. If there is no match then we go down the route map list which says, set everything else to metric 5. **It is always very important to ask the question, what will happen to routes that do not match any of the match statements because they will be dropped by default.**

#### Example 2:

Suppose in the above example we did not want AS100 to accept updates about 170.10.0.0. Since route maps cannot be applied on the inbound when matching based on an ip address, we have to use an outbound route map on RTC:

RTC#

```
router bgp 300
network 170.10.0.0
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 route-map STOPUPDATES out

route-map STOPUPDATES permit 10
match ip address 1

access-list 1 deny 170.10.0.0 0.0.255.255
access-list 1 permit 0.0.0.0 255.255.255.255
```

Now that you feel more comfortable with how to start BGP and how to define a neighbor, let's look at how to start exchanging network information.

There are multiple ways to send network information using BGP. I will go through these methods one by one.

## 7.0 Network command

The format of the network command follows:

```
network network-number [mask network-mask]
```

The network command controls what networks are originated by this box. This is a different concept from what you are used to configuring with IGRP and RIP. With this command we are not trying to run BGP on a certain interface, rather we are trying to indicate to BGP what networks it should originate from this box. The mask portion is used because BGP4 can handle subnetting and supernetting. A maximum of 200 entries of the network command are accepted.

**The network command will work if the network you are trying to advertise is known to the router, whether connected, static or learned dynamically.**

An example of the network command follows:

```
RTA#
router bgp 1
network 192.213.0.0 mask 255.255.0.0

ip route 192.213.0.0 255.255.0.0 null 0
```

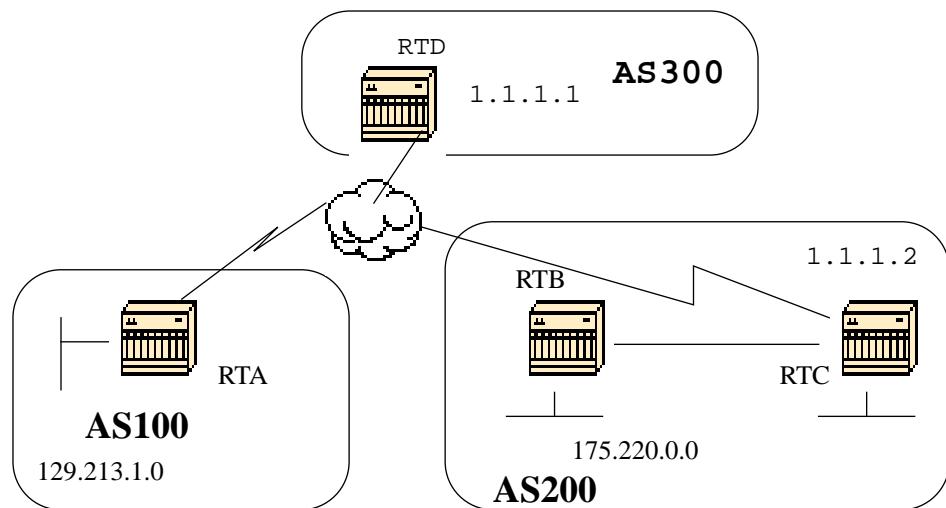
The above example indicates that router A, will generate a network entry for 192.213.0.0/16. The /16 indicates that we are using a supernet of the class C address and we are advertising the first two octets (the first 16 bits).

Note that we need the static route to get the router to generate 192.213.0.0 because the static route will put a matching entry in the routing table.

## 7.1 Redistribution

The network command is one way to advertise your networks via BGP. Another way is to redistribute your IGP (IGRP, OSPF, RIP, EIGRP, etc.) into BGP. This sounds scary because now you are dumping all of your internal routes into BGP, some of these routes might have been learned via BGP and you do not need to send them out again. Careful filtering should be applied to make sure you are sending to the internet only routes that you want to advertise and not everything you have. Let us look at the example below.

RTA is announcing 129.213.1.0 and RTC is announcing 175.220.0.0. Look at RTC's configuration:



If you use a network command you will have:

```
RTC#
router eigrp 10
network 175.220.0.0
redistribute bgp 200
default-metric 1000 100 250 100 1500

router bgp 200
neighbor 1.1.1.1 remote-as 300
network 175.220.0.0 mask 255.255.0.0 (this will limit the networks
originated by your AS to 175.220.0.0)
```

If you use redistribution instead you will have:

```
RTC#
router eigrp 10
network 175.220.0.0
redistribute bgp 200
default-metric 1000 100 250 100 1500

router bgp 200
neighbor 1.1.1.1 remote-as 300
redistribute eigrp 10 (eigrp will inject 129.213.1.0 again into BGP)
```

This will cause 129.213.1.0 to be originated by your AS. This is misleading because you are not the source of 129.213.1.0 but AS100 is. So you would have to use filters to prevent that network from being sourced out by your AS. The correct configuration would be:

```
RTC#
router eigrp 10
network 175.220.0.0
redistribute bgp 200
default-metric 1000 100 250 100 1500

router bgp 200
neighbor 1.1.1.1 remote-as 300
neighbor 1.1.1.1 distribute-list 1 out
redistribute eigrp 10

access-list 1 permit 175.220.0.0 0.0.255.255
```

The access-list is used to control what networks are to be originated from AS200.

## 7.2 Static routes and redistribution

You could always use static routes to originate a network or a subnet. The only difference is that BGP will consider these routes as having an origin of incomplete (unknown). In the above example the same could have been accomplished by doing:

```
RTC#
router eigrp 10
network 175.220.0.0
redistribute bgp 200
default-metric 1000 100 250 100 1500

router bgp 200
neighbor 1.1.1.1 remote-as 300
redistribute static

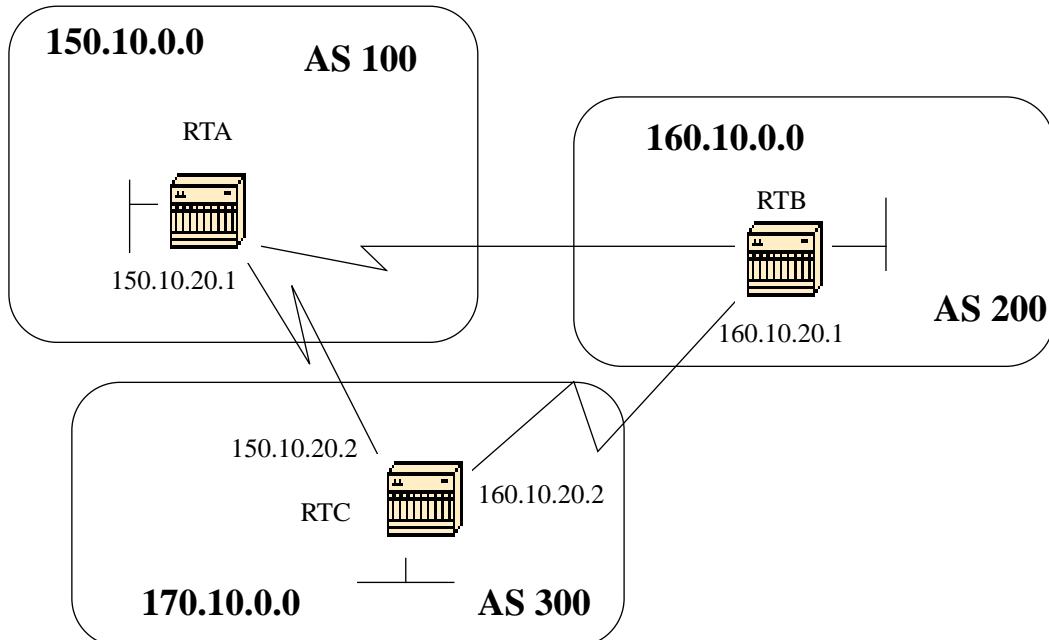
ip route 175.220.0.0 255.255.255.0 null0
```

The null 0 interface means to disregard the packet. So if I get the packet and there is a more specific match than 175.220.0.0 (which exists of course) the router will send it to the specific match otherwise it will disregard it. This is a nice way to advertise a supernet.

We have discussed how we can use different methods to originate routes out of our autonomous system. Please remember that these routes are generated in addition to other BGP routes that BGP has learned via neighbors (internal or external). BGP passes on information that it learns from one peer to other peers. The difference is that routes generated by the network command, or redistribution or static, will indicate your AS as the origin for these networks.

Injecting BGP into IGP is always done by redistribution.

Example:



```
RTA#
router bgp 100
neighbor 150.10.20.2 remote-as 300
network 150.10.0.0
```

```
RTB#
router bgp 200
neighbor 160.10.20.2 remote-as 300
network 160.10.0.0
```

```
RTC#
router bgp 300
neighbor 150.10.20.1 remote-as 100
neighbor 160.10.20.1 remote-as 200
network 170.10.0.0
```

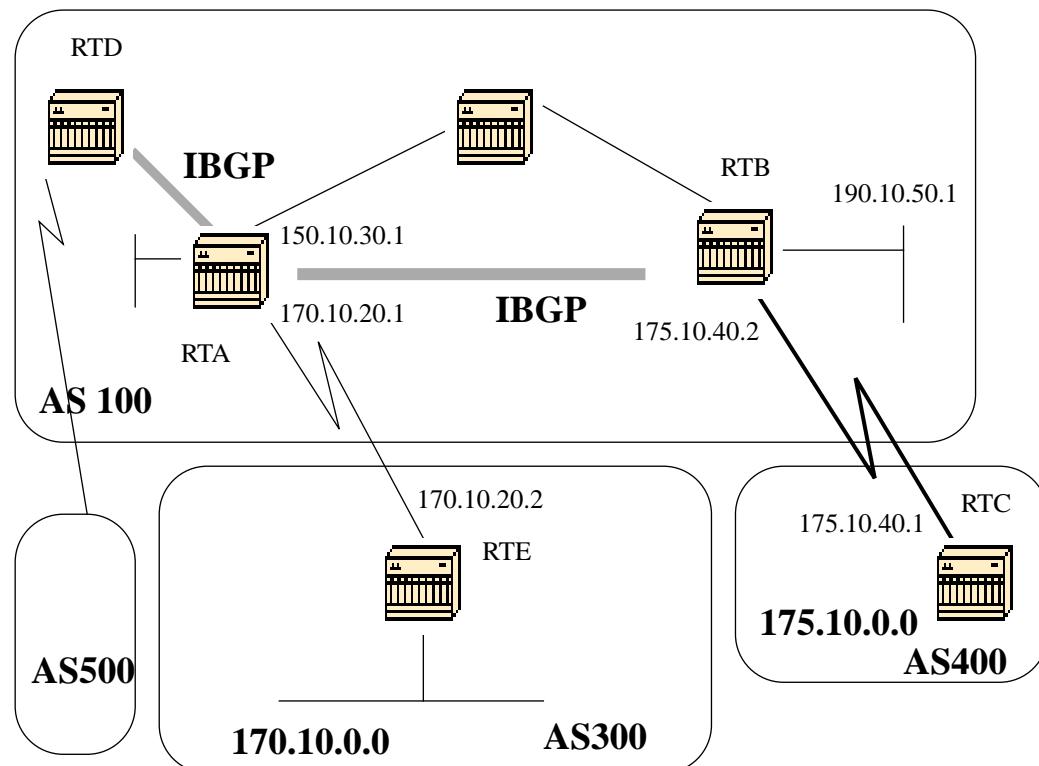
Note that you do not need network 150.10.0.0 or network 160.10.0.0 in RTC unless you want RTC to also generate these networks on top of passing them on as they come in from AS100 and AS200. Again the difference is that the network command will add an extra advertisement for these same networks indicating that AS300 is also an origin for these routes.

An important point to remember is that BGP will not accept updates that have originated from its own AS. This is to insure a loop free interdomain topology.

For example, assume AS200 above had a direct BGP connection into AS100. RTA will generate a route 150.10.0.0 and will send it to AS300, then RTC will pass this route to AS200 with the origin kept as AS100, RTB will pass 150.10.0.0 to AS100 with origin still AS100. RTA will notice that the update has originated from its own AS and will ignore it.

## 8.0 Internal BGP

IBGP is used if an AS wants to act as a transit system to other ASs. You might ask, why can't we do the same thing by learning via EBGP redistributing into IGP and then redistributing again into another AS? We can, but IBGP offers more flexibility and more efficient ways to exchange information within an AS; for example IBGP provides us with ways to control what is the best exit point out of the AS by using local preference (will be discussed later).



```
RTA#
router bgp 100
neighbor 190.10.50.1 remote-as 100
neighbor 170.10.20.2 remote-as 300
network 150.10.0.0
```

```
RTB#
router bgp 100
neighbor 150.10.30.1 remote-as 100
neighbor 175.10.40.1 remote-as 400
network 190.10.50.0
```

```
RTC#
router bgp 400
neighbor 175.10.40.2 remote-as 100
network 175.10.0.0
```

An important point to remember, is that when a BGP speaker receives an update from other BGP speakers in its own AS (IBGP), the receiving BGP speaker will not redistribute that information to other BGP speakers in its own AS. The receiving BGP speaker will redistribute that information to other BGP speakers outside of its AS. That is why it is important to sustain a full mesh between the IBGP speakers within an AS.

In the above diagram, RTA and RTB are running IBGP and RTA and RTD are running IBGP also. The BGP updates coming from RTB to RTA will be sent to RTE (outside of the AS) but not to RTD (inside of the AS). This is why an IBGP peering should be made between RTB and RTD in order not to break the flow of the updates.

## 9.0 The BGP decision algorithm

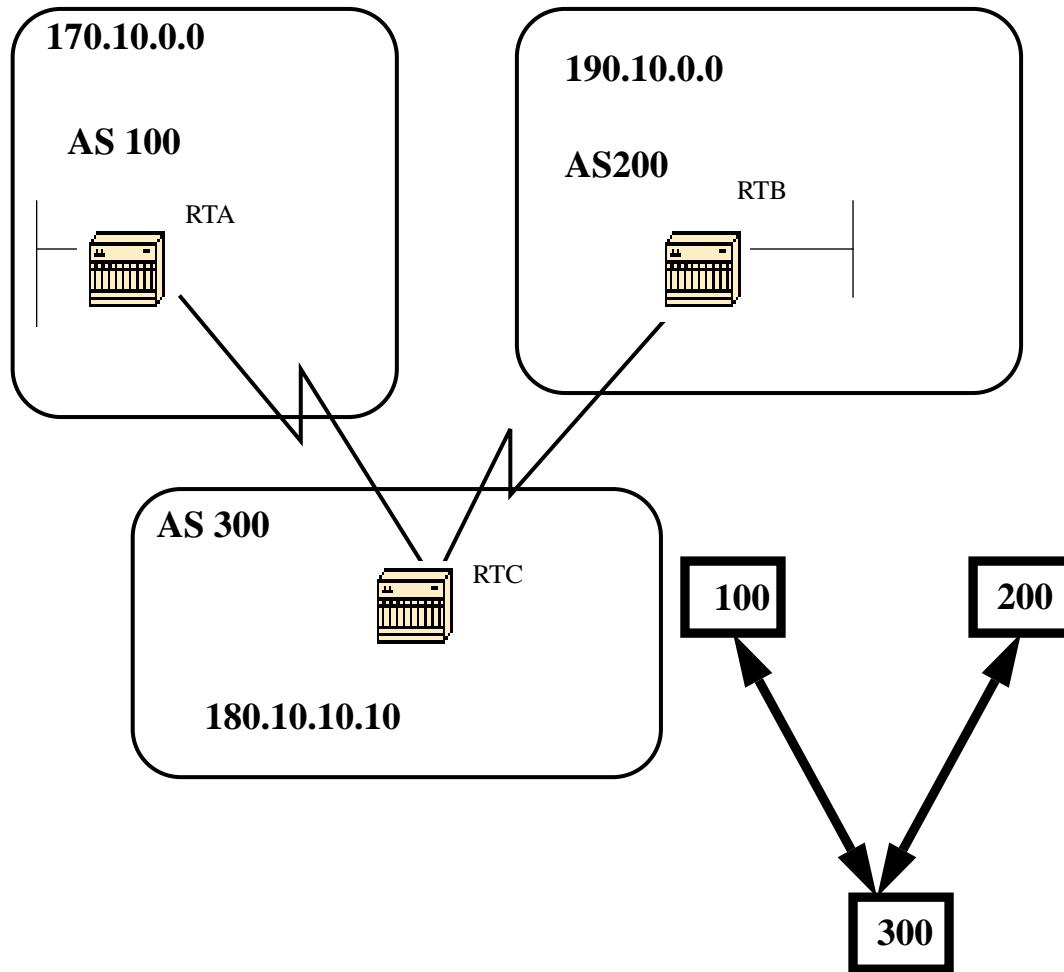
After BGP receives updates about different destinations from different autonomous systems, the protocol will have to decide which paths to choose in order to reach a specific destination. **BGP will choose only a single path to reach a specific destination.**

The decision process is based on different **attributes**, such as next hop, administrative weights, local preference, the route origin, path length, origin code, metric and so on.

**BGP will always propagate the best path to its neighbors.**

In the following section I will try to explain these attributes and show how they are used. We will start with the path attribute.

## 10.0 As\_path Attribute



Whenever a route update passes through an AS, the AS number is prepended to that update. The **AS\_path** attribute is actually the list of AS numbers that a route has traversed in order to reach a destination. An **AS-SET** is an ordered mathematical set {} of all the ASs that have been traversed. An example of AS-SET is given later.

In the above example, network 190.10.0.0 is advertised by RTB in AS200, when that route traverses AS300 and RTC will append its own AS number to it.

So when 190.10.0.0 reaches RTA it will have two AS numbers attached to it: first 200 then 300. So as far as RTA is concerned the path to reach 190.10.0.0 is (300,200).

The same applies for 170.10.0.0 and 180.10.0.0. RTB will have to take path (300,100) i.e. traverse AS300 and then AS100 in order to reach 170.10.0.0. RTC will have to traverse path (200) in order to reach 190.10.0.0 and path (100) in order to reach 170.10.0.0.

## 11.0 Origin Attribute

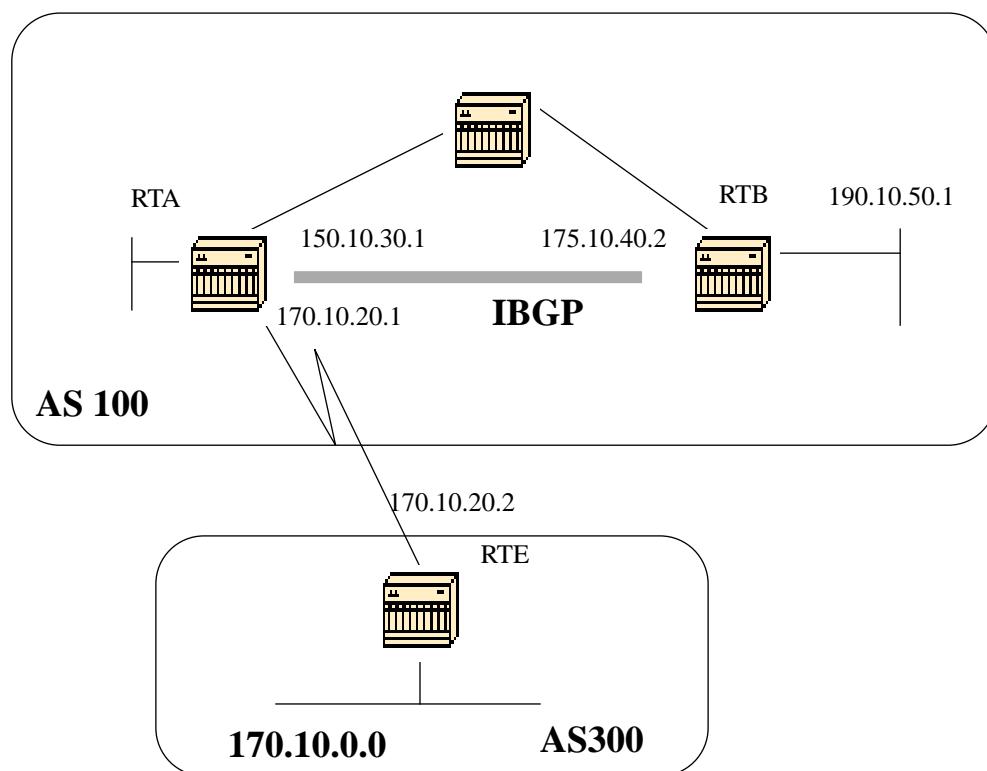
The origin is a mandatory attribute that defines the origin of the path information. The origin attribute can assume three values:

IGP: Network Layer Reachability Information (NLRI) is interior to the originating AS. This normally happens when we use the bgp network command or when IGP is redistributed into BGP, then the origin of the path info will be IGP. This is indicated with an "i" in the BGP table.

EGP: NLRI is learned via EGP (Exterior Gateway Protocol). This is indicated with an "e" in the BGP table.

INCOMPLETE: NLRI is unknown or learned via some other means. This usually occurs when we redistribute a static route into BGP and the origin of the route will be incomplete. This is indicated with an "?" in the BGP table.

Example:



```
RTA#
router bgp 100
neighbor 190.10.50.1 remote-as 100
neighbor 170.10.20.2 remote-as 300
network 150.10.0.0
redistribute static

ip route 190.10.0.0 255.255.0.0 null0
```

```
RTB#
router bgp 100
neighbor 150.10.30.1 remote-as 100
network 190.10.50.0
```

```
RTE#
router bgp 300
neighbor 170.10.20.1 remote-as 100
network 170.10.0.0
```

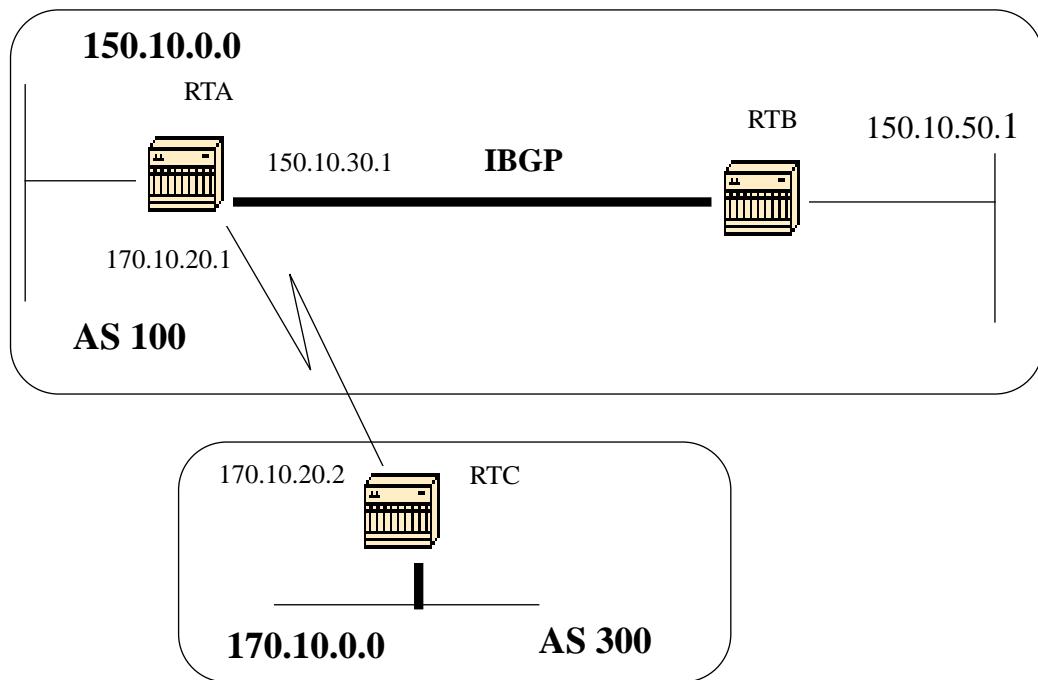
RTA will reach 170.10.0.0 via: 300 i (which means the next AS path is 300 and the origin of the route is IGP).

RTA will also reach 190.10.50.0 via: i (which means, the entry is in the same AS and the origin is IGP).

RTE will reach 150.10.0.0 via: 100 i (the next AS is 100 and the origin is IGP).

RTE will also reach 190.10.0.0 via: 100 ? (the next AS is 100 and the origin is incomplete "?", coming from a static route).

## 12.0 BGP Nexthop Attribute



The BGP nexthop attribute is the next hop IP address that is going to be used to reach a certain destination.

For EBGP, the next hop is always the IP address of the neighbor specified in the neighbor command<sup>1</sup>. In the above example, RTC will advertise 170.10.0.0 to RTA with a next hop of 170.10.20.2 and RTA will advertise 150.10.0.0 to RTC with a next hop of 170.10.20.1. For IBGP, the protocol states **that the next hop advertised by EBGP should be carried into IBGP**. Because of that rule, RTA will advertise 170.10.0.0 to its IBGP peer RTB with a next hop of 170.10.20.2. So according to RTB, the next hop to reach 170.10.0.0 is 170.10.20.2 and **NOT** 150.10.30.1.

You should make sure that RTB can reach 170.10.20.2 via IGP, otherwise RTB will drop packets destined to 170.10.0.0 because the next hop address would be inaccessible. For example, if RTB is running igrp you could also run IGRP on RTA network 170.10.0.0. You would want to make IGRP passive on the link to RTC so BGP is only exchanged.

1.This is not true if the next hop is on a multiaccess media, in which case the nexthop will be the ip address of the router that is closest to the destination. This is described in the following sections.

Example:

```
RTA#
router bgp 100
neighbor 170.10.20.2 remote-as 300
neighbor 150.10.50.1 remote-as 100
network 150.10.0.0

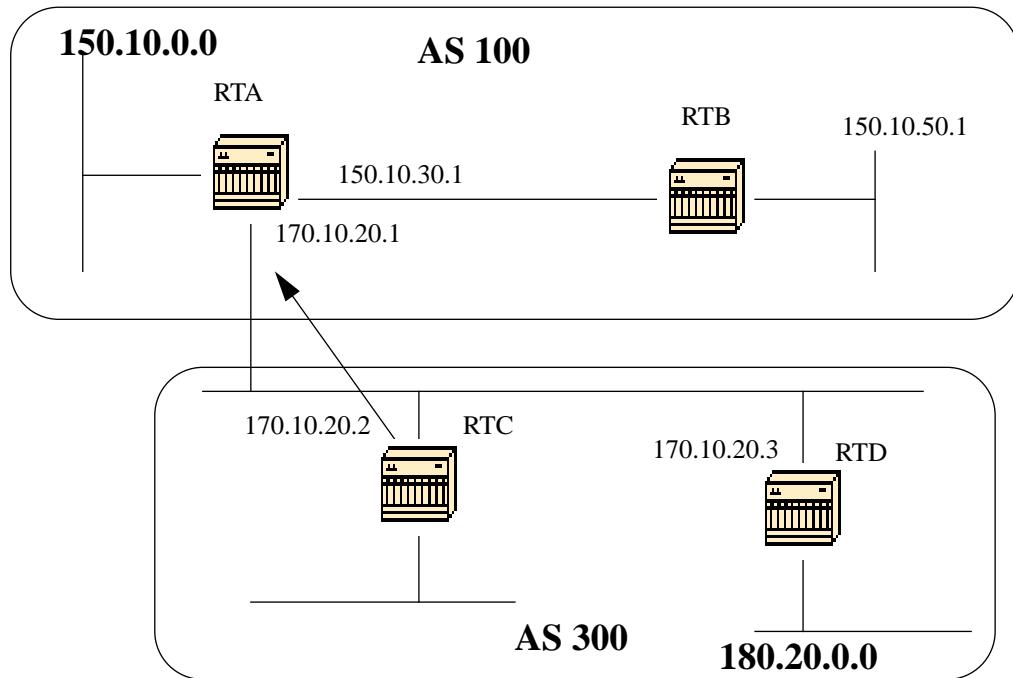
RTB#
router bgp 100
neighbor 150.10.30.1 remote-as 100

RTC#
router bgp 300
neighbor 170.10.20.1 remote-as 100
network 170.10.0.0

*RTC will advertise 170.10.0.0 to RTA with a NextHop = 170.10.20.2
*RTA will advertise 170.10.0.0 to RTB with a NextHop=170.10.20.2
(The external NextHop via EBGP is sent via IBGP)
```

Special care should be taken when dealing with multiaccess and NBMA networks as described in the following sections.

## 12.1 BGP Nexthop (Multiaccess Networks)



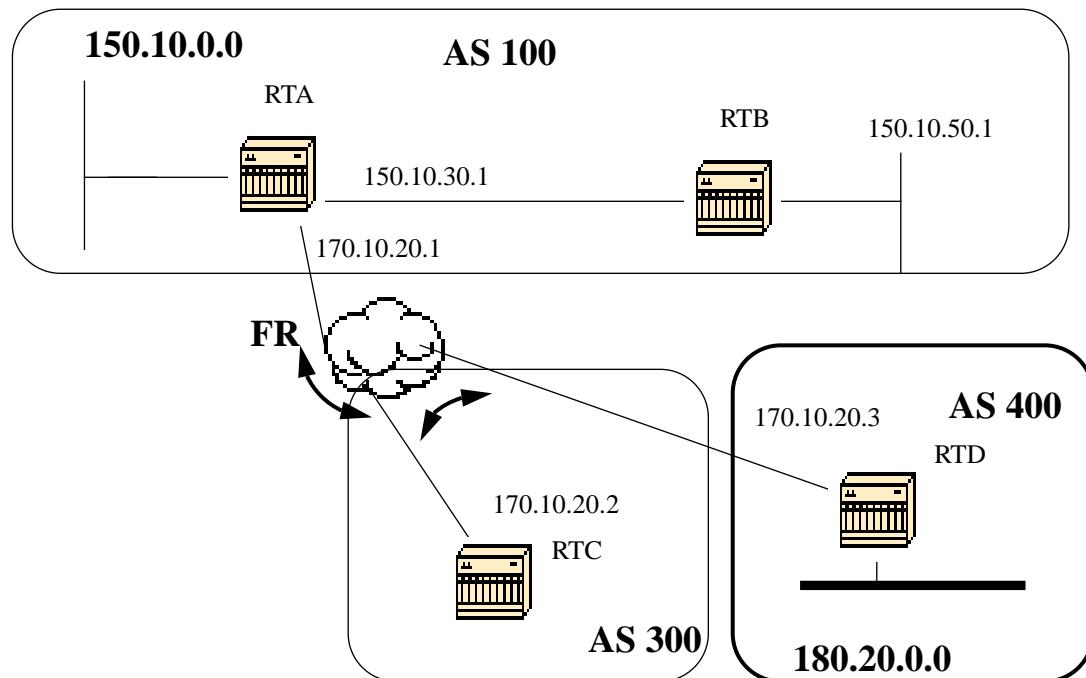
The following example shows how the nexthop will behave on a multiaccess network such as ethernet.

Assume that RTC and RTD in AS300 are running OSPF. RTC is running BGP with RTA. RTC can reach network 180.20.0.0 via 170.10.20.3. When RTC sends a BGP update to RTA regarding 180.20.0.0 it will use as next hop 170.10.20.3 and not its own IP address (170.10.20.2). This is because the network between RTA, RTC and RTD is a multiaccess network and it makes more sense for RTA to use RTD as a next hop to reach 180.20.0.0 rather than making an extra hop via RTC.

\*RTC will advertise 180.20.0.0 to RTA with a NextHop = 170.10.20.3.

If the common media to RTA, RTC and RTD was not multiaccess, but NBMA (Non Broadcast Media Access) then further complications will occur.

## 12.2 BGP Nexthop (NBMA)



If the common media as you see in the shaded area above is a frame relay or any NBMA cloud then the exact behavior will occur as if we were connected via ethernet. RTC will advertise 180.20.0.0 to RTA with a next hop of 170.10.20.3.

The problem is that RTA does not have a direct PVC to RTD, and cannot reach the next hop. In this case routing will fail.

In order to remedy this situation a command called `NextHopSelf` is created.

## 12.3 Next-hop-self

Because of certain situations with the nexthop as we saw in the previous example, a command called **next-hop-self** is created.

the syntax is:

```
neighbor {ip-address|peer-group-name1} next-hop-self
```

The next-hop-self command will allow us to force BGP to use a specified IP address as the next hop rather than letting the protocol choose the nexthop.

In the previous example the following will solve our problem:

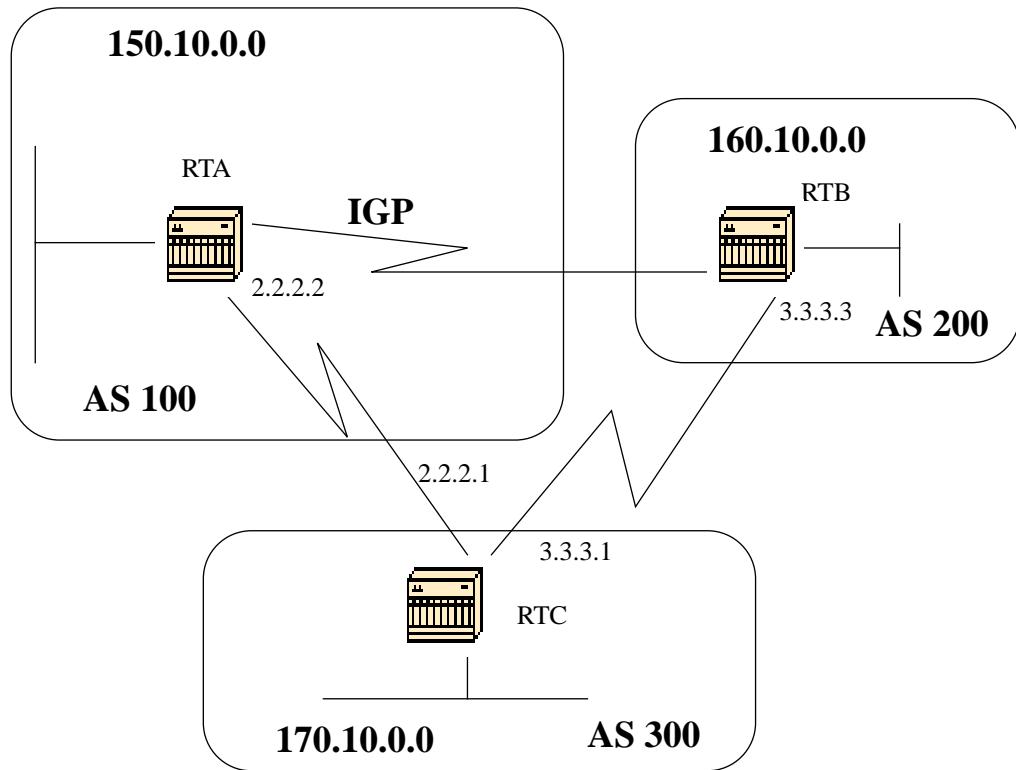
```
RTC#
router bgp 300
neighbor 170.10.20.1 remote-as 100
neighbor 170.10.20.1 next-hop-self
```

```
RTC will advertise 180.20.0.0 with a NextHop = 170.10.20.2
```

---

1.We will discuss peer-group-names later on

## 13.0 BGP Backdoor



Consider the above diagram, RTA and RTC are running EBGP and RTB and RTC are running EBGP. RTA and RTB are running some kind of IGP (RIP, IGRP, etc.).

By definition, EBGP updates have a distance of 20 which is lower than the IGP distances. Default distance is 120 for RIP, 100 for IGRP, 90 for EIGRP and 110 for OSPF.

RTA will receive updates about 160.10.0.0 via two routing protocols: EBGP with a distance of 20 and IGP with a distance higher than 20.

By default, BGP has the following distances, but that could be changed by the `distance` command:

```
distance bgp external-distance internal-distance local-distance
```

```
external-distance:20
internal-distance:200
local-distance:200
```

RTA will pick EBGP via RTC because of the lower distance.

If we want RTA to learn about 160.10.0.0 via RTB (IGP), then we have two options:

1- Change EBGP's external distance or IGP's distance which is NOT recommended.

2- Use BGP backdoor

BGP backdoor will make the IGP route, the preferred route.

Use the following command: **network address backdoor**.

The configured network is the network that we would like to reach via IGP. For BGP this network will be treated as a locally assigned network except it will not be advertised in bgp updates.

Example:

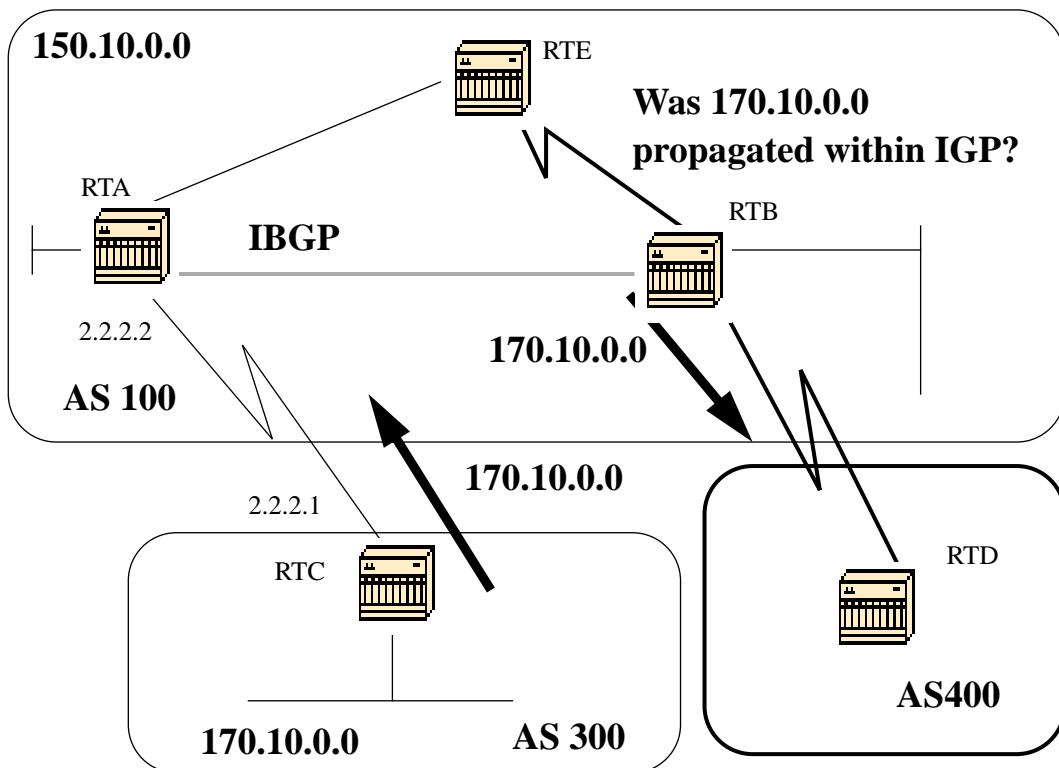
```
RTA#
router eigrp 10
network 160.10.0.0

router bgp 100
neighbor 2.2.2.1 remote-as 300
network 160.10.0.0 backdoor
```

Network 160.10.0.0 will be treated as a local entry but will not be advertised as a normal network entry would.

RTA will learn 160.10.0.0 from RTB via EIGRP with distance 90, and will also learn it from RTC via EBGP with distance 20. Normally EBGP will be preferred, but because of the backdoor command EIGRP will be preferred.

## 14.0 Synchronization



Before we discuss synchronization let us look at the following scenario. RTC in AS300 is sending updates about 170.10.0.0. RTA and RTB are running IBGP, so RTB will get the update and will be able to reach 170.10.0.0 via next hop 2.2.2.1 (remember that the next hop is carried via IBGP). In order to reach the next hop, RTB will have to send the traffic to RTE.

Assume that RTA has not redistributed network 170.10.0.0 into IGP, so at this point RTE has no idea that 170.10.0.0 even exists.

If RTB starts advertising to AS400 that he can reach 170.10.0.0 then traffic coming from RTD to RTB with destination 170.10.0.0 will flow in and get dropped at RTE.

Synchronization states: If your autonomous system is passing traffic from another AS to a third AS, BGP should not advertise a route before all routers in your AS have learned about the route via IGP<sup>1</sup>. BGP will wait until IGP has propagated the route within the AS and then will advertise it to external peers. This is called synchronization.

---

1. As far as the router is concerned, we will check to see if we have a route in the ip routing table. This could be done by defining a static route.

In the above example, RTB will wait to hear about 170.10.0.0 via IGP before it starts sending the update to RTD. We can fool RTB into thinking that IGP has propagated the information by adding a static route in RTB pointing to 170.10.0.0. Care should be taken to make sure that other routers can reach 170.10.0.0 otherwise we will have a problem reaching that network.

## 14.1 Disabling synchronization

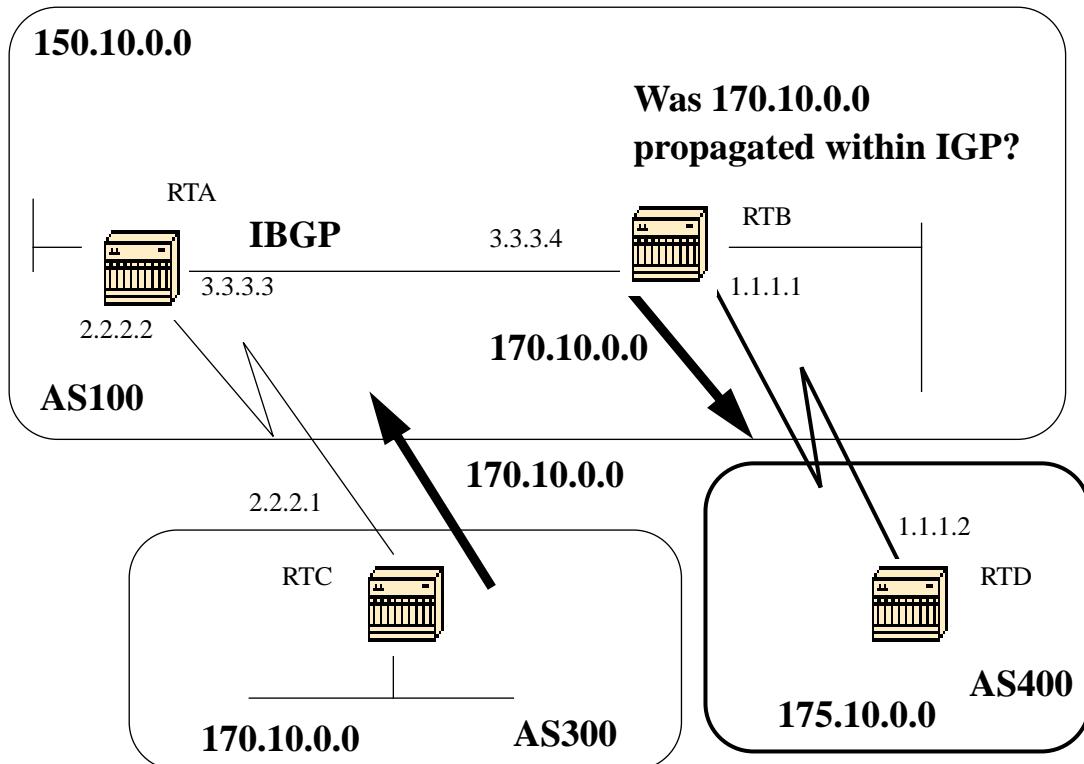
In some cases you do not need synchronization. If you will not be passing traffic from a different autonomous system through your AS, or if all routers in your AS will be running BGP, you can disable synchronization. Disabling this feature can allow you to carry fewer routes in your IGP and allow BGP to converge more quickly.

Disabling synchronization is not automatic, if you have all your routers in the AS running BGP and you are not running any IGP, the router has no way of knowing that, and your router will be waiting forever for an IGP update about a certain route before sending it to external peers. You have to disable synchronization manually in this case for routing to work correctly.

```
router bgp 100
no synchronization.
```

(Make sure you do a **clear ip bgp address** to reset the session)

Example:

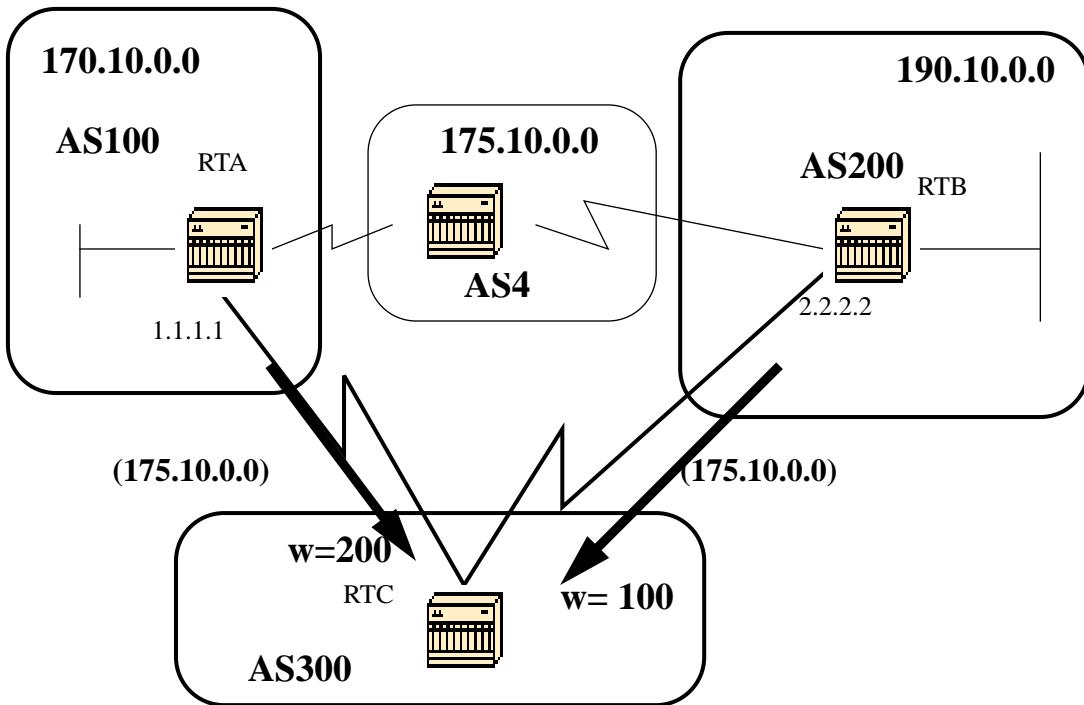


```
RTB#
router bgp 100
network 150.10.0.0
neighbor 1.1.1.2 remote-as 400
neighbor 3.3.3.3 remote-as 100
no synchronization (RTB will put 170.10.0.0 in its ip routing table and
will advertise it to RTD even if it does not have an IGP path to
170.10.0.0)

RTD#
router bgp 400
neighbor 1.1.1.1 remote-as 100
network 175.10.0.0

RTA#
router bgp 100
network 150.10.0.0
neighbor 3.3.3.4 remote-as 100
```

## 15.0 Weight Attribute



The weight attribute is a Cisco defined attribute. The weight is used for a best path selection process. The weight is assigned locally to the router. It is a value that only makes sense to the specific router and which is not propagated or carried through any of the route updates. A weight can be a number from 0 to 65535. Paths that the router originates have a weight of 32768 by default and other paths have a weight of zero.

Routes with a higher weight are preferred when multiple routes exist to the same destination. Let us study the above example. RTA has learned about network 175.10.0.0 from AS4 and will propagate the update to RTC. RTB has also learned about network 175.10.0.0 from AS4 and will propagate it to RTC. RTC has now two ways for reaching 175.10.0.0 and has to decide which way to go. If on RTC we can set the weight of the updates coming from RTA to be higher than the weight of updates coming from RTB, then we will force RTC to use RTA as a next hop to reach 175.10.0.0. This is achieved by using multiple methods:

- 1- Using the neighbor command

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

- 2- Using AS path access-lists

```
ip as-path access-list access-list-number {permit|deny} as-regular-expression
```

```
neighbor ip-address filter-list access-list-number weight weight
```

### 3-Using route-maps

example:

```
RTC#
router bgp 300
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 weight 200 (route to 175.10.0.0 from RTA will have 200
weight)
neighbor 2.2.2.2 remote-as 200
neighbor 2.2.2.2 weight 100 (route to 175.10.0.0 from RTB will have 100
weight)
```

\*Routes with higher weight are preferred when multiple routes exist to the same destination. RTA will be preferred as the next hop.

The same outcome can be achieved via ip as-path and filter lists.

```
RTC#
router bgp 300
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 filter-list 5 weight 200
neighbor 2.2.2.2 remote-as 200
neighbor 2.2.2.2 filter-list 6 weight 100

ip as-path access-list 5 permit ^100$(this will only permit path 100)
ip as-path access-list 6 permit ^200$
```

The same outcome as above can be achieved by using routemaps.

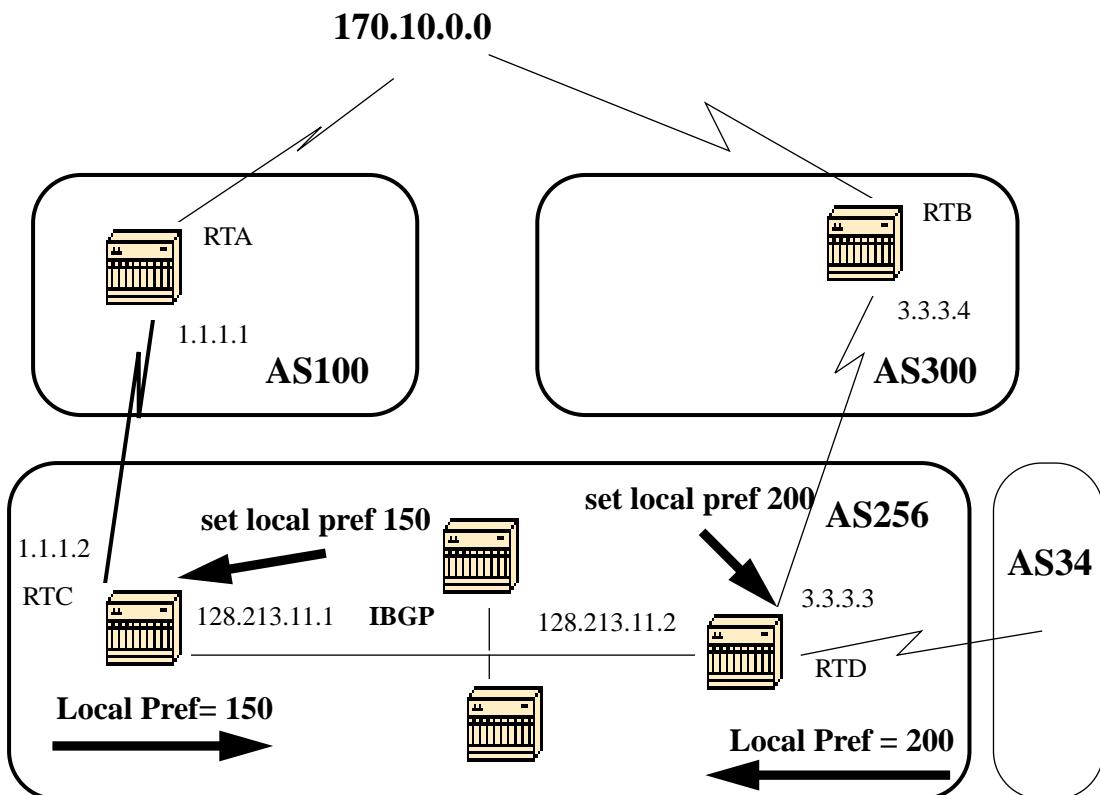
```
RTC#
router bgp 300
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 route-map setweightin in
neighbor 2.2.2.2 remote-as 200
neighbor 2.2.2.2 route-map setweightin in

ip as-path access-list 5 permit ^100$

route-map setweightin permit 10
match as-path 5
set weight 200
(anything that applies to access-list 5, i.e. packets from AS100, would
have weight 200)

route-map setweightin permit 20
set weight 100
(anything else would have weight 100)
```

## 16.0 Local Preference Attribute



Local preference is an indication to the AS about which path is preferred to exit the AS in order to reach a certain network. A path with a higher local preference is more preferred. The default value for local preference is 100.

Unlike the weight attribute which is only relevant to the local router, local preference is an attribute that is **exchanged among routers in the same AS**.

Local preference is set via the "**bgp default local-preference <value>**" command or with route-maps as will be demonstrated in the following example:

The **bgp default local-preference <value>** command will set the local preference on the updates out of the router going to peers in the same AS. In the above diagram, AS256 is receiving updates about 170.10.0.0 from two different sides of the organization. Local preference will help us determine which way to exit AS256 in order to reach that network. Let us assume that RTD is the preferred exit point. The following configuration will set the local preference for updates coming from AS300 to 200 and those coming from AS100 to 150.

```
RTC#
router bgp 256
neighbor 1.1.1.1 remote-as 100
neighbor 128.213.11.2 remote-as 256
bgp default local-preference 150
```

```
RTD#
router bgp 256
neighbor 3.3.3.4 remote-as 300
neighbor 128.213.11.1 remote-as 256
bgp default local-preference 200
```

In the above configuration RTC will set the local preference of all updates to 150. The same RTD will set the local preference of all updates to 200. Since local preference is exchanged within AS256, both RTC and RTD will realize that network 170.10.0.0 has a higher local preference when coming from AS300 rather than when coming from AS100. All traffic in AS256 addressed to that network will be sent to RTD as an exit point.

More flexibility is provided by using route maps. In the above example, all updates received by RTD will be tagged with local preference 200 when they reach RTD. This means that updates coming from AS34 will also be tagged with the local preference of 200. This might not be needed. This is why we can use route maps to specify what specific updates need to be tagged with a specific local preference as shown below:

```
RTD#
router bgp 256
neighbor 3.3.3.4 remote-as 300
neighbor 3.3.3.4 setlocalin in
neighbor 128.213.11.1 remote-as 256
```

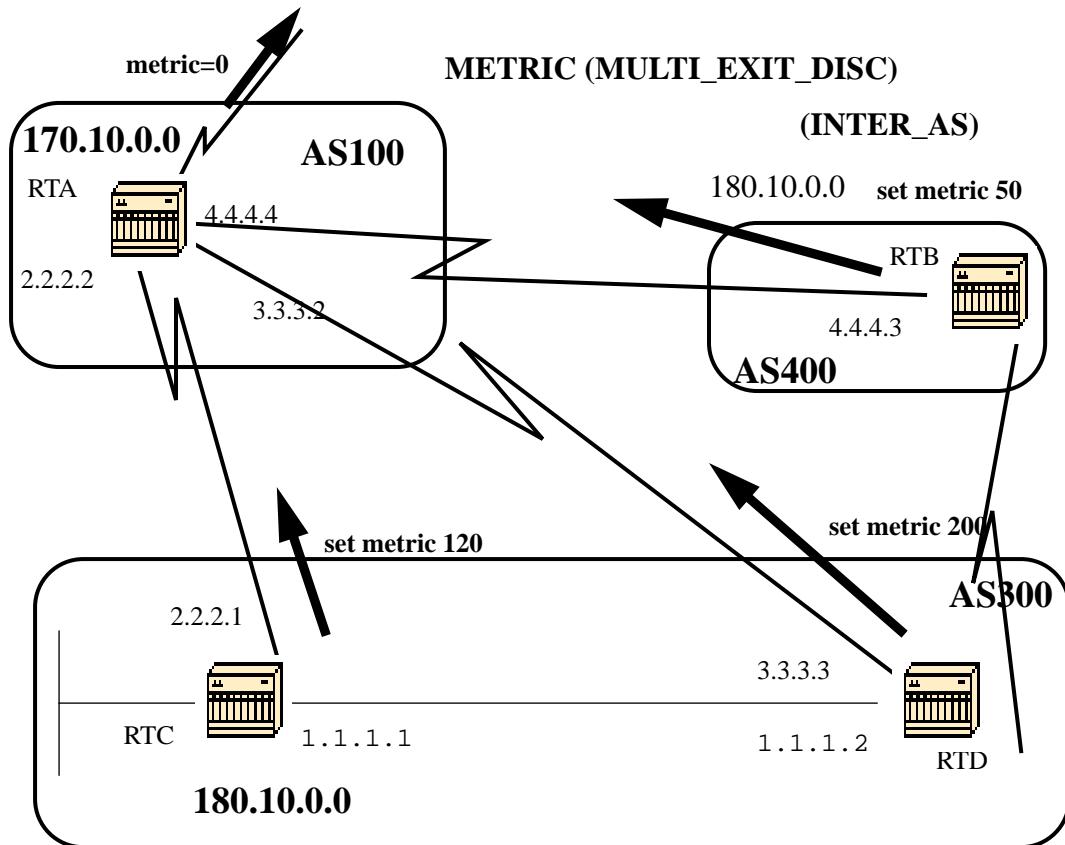
```
ip as-path 7 permit ^300$

route-map setlocalin permit 10
match as-path 7
set local-preference 400

route-map setlocalin permit 20
set local-preference 150
```

With this configuration, any update coming from AS300 will be set with a local preference of 200. Any other updates such as those coming from AS34 will be set with a value of 150.

## 17.0 Metric Attribute



The metric attribute which is also called Multi\_exit\_discriminator (MED, BGP4) or Inter-As (BGP3) is a hint to external neighbors about the preferred path into an AS. This is a dynamic way to influence another AS on which way to choose in order to reach a certain route given that we have multiple entry points into that AS. **A lower value of a metric is more preferred.**

Unlike local preference, metric is exchanged between ASs. A metric is carried into an AS but does not leave the AS. When an update enters the AS with a certain metric, that metric is used for decision making inside the AS. When the same update is passed on to a third AS, that metric will be set back to 0 as shown in the above diagram. The Metric default value is 0.

Unless otherwise specified, a router will compare metrics for paths from neighbors in the same AS. **In order for the router to compare metrics from neighbors coming from different ASs the special configuration command "bgp always-compare-med" should be configured on the router.**

In the above diagram, AS100 is getting information about network 180.10.0.0 via three different routers: RTC, RTD and RTB. RTC and

RTD are in AS300 and RTB is in AS400.

Assume that we have set the metric coming from RTC to 120, the metric coming from RTD to 200 and the metric coming from RTB to 50. Given that by default a router compares metrics coming from neighbors in the same AS, RTA can only compare the metric coming from RTC to the metric coming from RTD and will pick RTC as the best next hop because 120 is less than 200. When RTA gets an update from RTB with metric 50, he can not compare it to 120 because RTC and RTB are in different ASs (RTA has to choose based on some other attributes).

In order to force RTA to compare the metrics we have to add **bgp always-compare-med** to RTA. This is illustrated in the configs below:

```
RTA#
router bgp 100
neighbor 2.2.2.1 remote-as 300
neighbor 3.3.3.3 remote-as 300
neighbor 4.4.4.3 remote-as 400
```

```
RTC#
router bgp 300
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 route-map setmetricout out
neighbor 1.1.1.2 remote-as 300
```

```
route-map setmetricout permit 10
set metric 120
```

```
RTD#
router bgp 300
neighbor 3.3.3.2 remote-as 100
neighbor 3.3.3.2 route-map setmetricout out
neighbor 1.1.1.1 remote-as 300
```

```
route-map setmetricout permit 10
set metric 200
```

```
RTB#
router bgp 400
neighbor 4.4.4.4 remote-as 100
neighbor 4.4.4.4 route-map setmetricout out

route-map setmetricout permit 10
set metric 50
```

With the above configs, RTA will pick RTC as next hop, considering all other attributes are the same.

In order to have RTB included in the metric comparison, we have to configure RTA as follows:

```
RTA#
router bgp 100
neighbor 2.2.21 remote-as 300
neighbor 3.3.3.3 remote-as 300
neighbor 4.4.4.3 remote-as 400
bgp always-compare-med
```

In this case RTA will pick RTB as the best next hop in order to reach network 180.10.0.0.

Metric can also be set while redistributing routes into BGP, the command is:

```
default-metric number
```

Assume in the above example that RTB is injecting a network via static into AS100 then the following configs:

```
RTB#
router bgp 400
redistribute static
default-metric 50

ip route 180.10.0.0 255.255.0.0 null 0

will cause RTB to send out 180.10.0.0 with a metric of 50.
```

## 18.0 Community Attribute

The community attribute is a transitive, optional attribute in the range 0 to 4,294,967,200. The community attribute is a way to group destinations in a certain community and apply routing decisions (accept, prefer, redistribute, etc.) according to those communities.

We can use route maps to set the community attributes. The route map set command has the following syntax:

```
set community community-number [additive]
```

A few predefined well known communities (community-number) are:

- no-export** (Do not advertise to EBGP peers)
- no-advertise** (Do not advertise this route to any peer)
- internet** (Advertise this route to the internet community, any router belongs to it)

An example of route maps where community is set is:

```
route-map communitymap
match ip address 1
set community no-advertise
```

or

```
route-map setcommunity
match as-path 1
set community 200 additive
```

If the additive keyword is not set, 200 will replace any old community that already exists; if we use the keyword additive then the 200 will be added to the community.

**Even if we set the community attribute, this attribute will not be sent to neighbors by default.**

In order to send the attribute to our neighbor we have to use the following:

```
neighbor {ip-address|peer-group-name} send-community
```

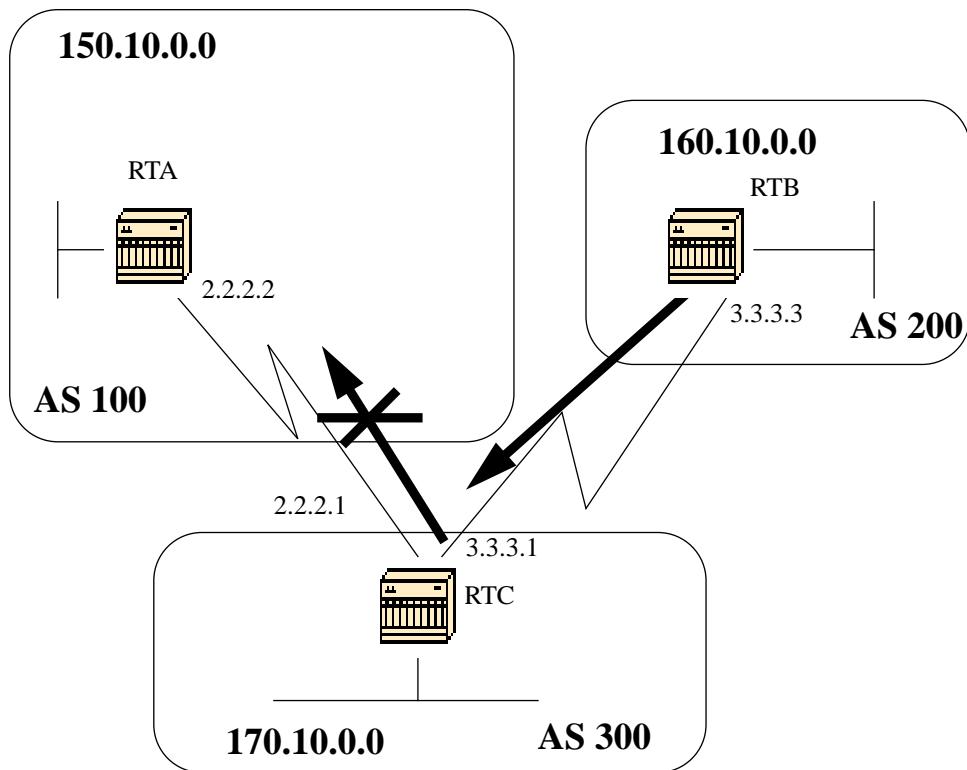
Example:

```
RTA#
router bgp 100
neighbor 3.3.3.3 remote-as 300
neighbor 3.3.3.3 send-community
neighbor 3.3.3.3 route-map setcommunity out
```

## 19.0 BGP Filtering

Sending and receiving BGP updates can be controlled by using a number of different filtering methods. BGP updates can be filtered based on route information, on path information or on communities. All methods will achieve the same results, choosing one over the other depends on the specific network configuration.

### 19.1 Route Filtering



In order to restrict the routing information that the router learns or advertises, you can filter BGP based on routing updates to or from a particular neighbor. In order to achieve this, an access-list is defined and applied to the updates to or from a neighbor. Use the following command in the router configuration mode:

```
Neighbor {ip-address|peer-group-name} distribute-list access-list-number
{in | out}
```

In the following example, RTB is originating network 160.10.0.0 and sending it to RTC. If RTC wanted to stop those updates from propagating to AS100, we would have to apply an access-list to filter those updates and apply it when talking to RTA:

```
RTC#
router bgp 300
network 170.10.0.0
neighbor 3.3.3.3 remote-as 200
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 distribute-list 1 out

access-list 1 deny 160.10.0.0 0.0.255.255
access-list 1 permit 0.0.0.0 255.255.255.255
(filter out all routing updates about 160.10.x.x)
```

Using access-lists is a bit tricky when we are dealing with supernets that might cause some conflicts.

Assume in the above example that RTB has different subnets of 160.10.X.X and our goal is to filter updates and advertise only 160.0.0.0/8 (this notation means that we are using 8 bits of subnet mask starting from the far left of the IP address; this is equivalent to 160.0.0.0 255.0.0.0.)

The following access list:

```
access-list 1 permit 160.0.0.0 0.255.255.255
```

will permit 160.0.0.0/8, 160.0.0.0/9 and so on. In order to restrict the update to only 160.0.0.0/8 we have to use an extended access list of the following format:

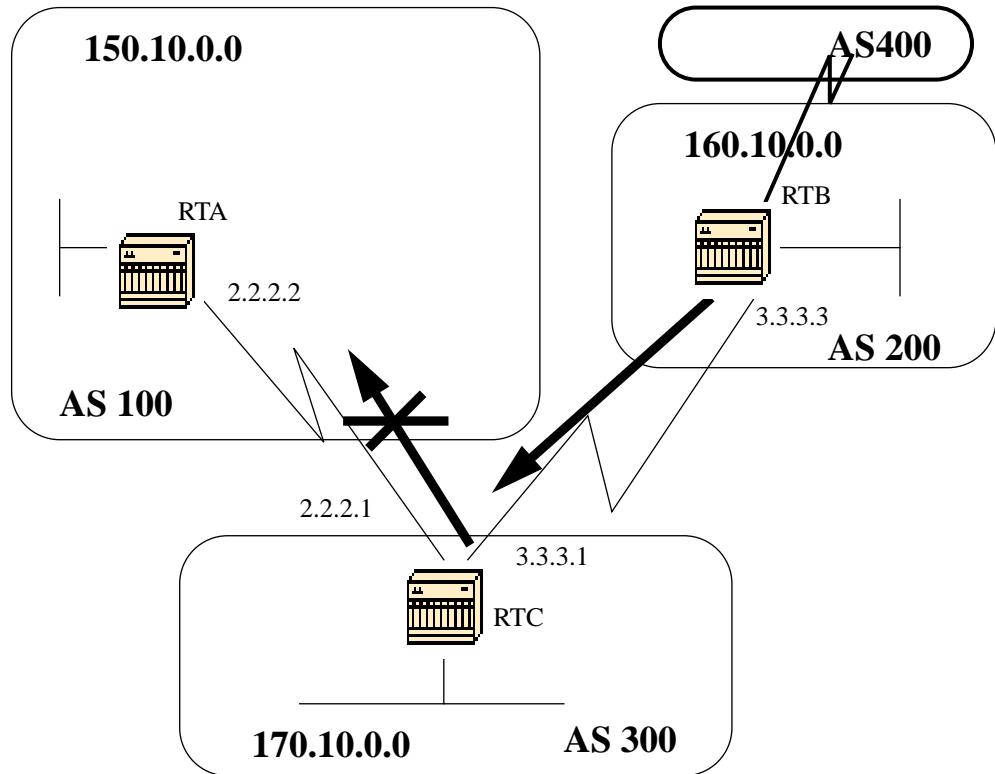
```
access-list <number> permit ip <ip address> <ip address don't care bits>
<mask> <mask don't care bits>
```

```
ex: access-list 101 permit ip 160.0.0.0 0.255.255.255 255.0.0.0 0.0.0.0
```

This list will permit 160.0.0.0/8 only.

Another type of filtering, is path filtering which is described in the next section.

## 19.2 Path Filtering



You can specify an access list on both incoming and outgoing updates based on the BGP autonomous system paths information. In the above figure we can block updates about 160.10.0.0 from going to AS100 by defining an access list on RTC that prevents any updates that have originated from AS 200 from being sent to AS100. To do this use the following statements.

```
ip as-path access-list access-list-number {permit|deny} as-regular-expression1
```

```
neighbor {ip-address|peer-group-name} filter-list access-list-number {in|out}
```

The following example will stop RTC from sending RTA updates about 160.10.0.0

```
RTC#
router bgp 300
neighbor 3.3.3.3 remote-as 200
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 filter-list 1 out (the 1 is the access list number below)
```

---

1. This term will be discussed shortly

```
ip as-path access-list 1 deny ^200$
ip as-path access-list 1 permit .*
```

In the above example, access-list 1 states: deny any updates with path information that start with 200 (^) and end with 200 (\$). The ^200\$ is called a regular expression, with ^ meaning starts with and \$ meaning ends with. Since RTB sends updates about 160.10.0.0 with path information starting with 200 and ending with 200, then this update will match the access list and will be denied.

The .\* is another regular expression with the dot meaning any character and the \* meaning the repetition of that character. So .\* is actually any path information, which is needed to permit all other updates to be sent.

What would happen if instead of using ^200\$ we have used ^200? If you have an AS400 (see figure above), updates originated by AS400 will have path information of the form (200, 400) with 200 being first and 400 being last. Those updates will match the access list ^200 because they start with 200 and will be prevented from being sent to RTA which is not the required behavior.

A good way to check whether we have implemented the correct regular expression is to do:

```
sh ip bgp regexp <regular expression>.
```

This will show us all the path that has matched the configured regular expression.

Regular expressions sound a bit complicated but actually they are not. The next section will explain what is involved in creating a regular expression.

### 19.2.1 AS-Regular Expression

A regular expression is a pattern to match against an input string. By building a regular expression we specify a string that input must match. In case of BGP we are specifying a string consisting of path information that an input should match.

In the previous example we specified the string ^200\$ and wanted path information coming inside updates to match it in order to perform a decision.

The regular expression is composed of the following:

#### A- Ranges:

A range is a sequence of characters contained within left and right square brackets. ex: [abcd]

#### B- Atoms

An atom is a single character

- . (Matches any single character)
- ^ (Matches the beginning of the input string)
- \$ (Matches the end of the input string)
- \character (Matches the character)
- (Matches a comma (,), left brace ({}), right brace (}), the beginning of the input string, the end of the input string, or a space.)

#### C-Pieces

A piece is an atom followed by one of the symbols:

- \* (Matches 0 or more sequences of the atom)
- + (Matches 1 or more sequences of the atom)
- ? (Matches the atom or the null string)

#### D- Branch

A branch is a 0 or more concatenated pieces.

Examples of regular expressions follow:

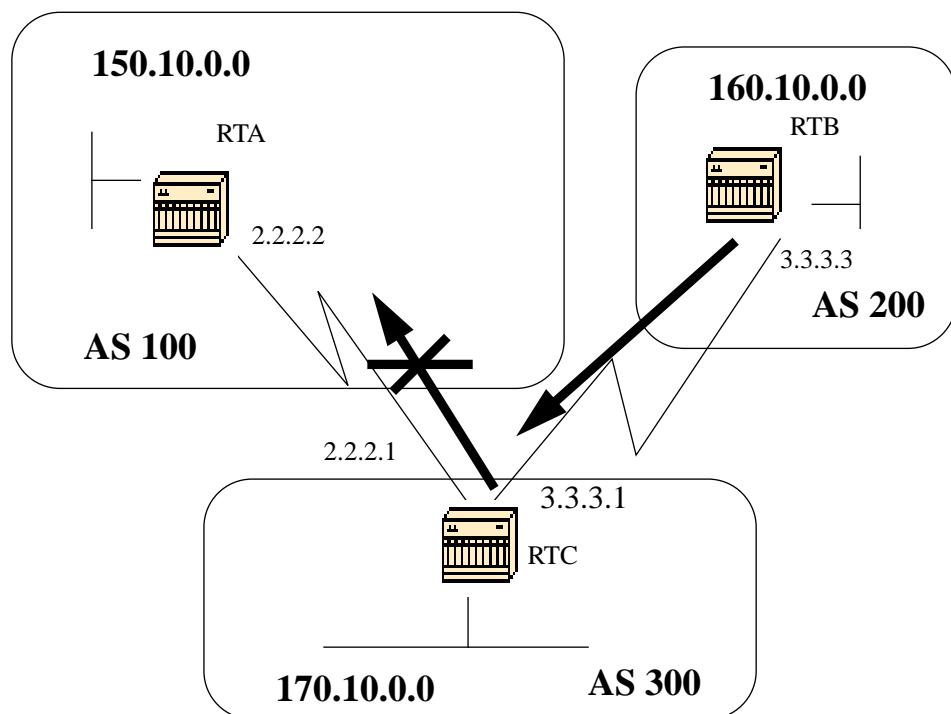
a\* any occurrence of the letter a, including none  
a+ at least one occurrence of a should be present  
ab?a this will match aa or aba

ex:

\_100\_(via AS100)  
^100\$ (origin AS100)  
^100 .\* (coming from AS100)  
^\$ (originated from this AS)

### 19.3 BGP Community Filtering

We have already seen route filtering and as-path filtering. Another method is community filtering. Community has been discussed in section 19.0 and here are few examples of how we can use it.



We would like RTB above to set the community attribute to the bgp routes it is advertising such that RTC would not propagate these routes to its external peers. The no-export community attribute is used:

```
RTB#
router bgp 200
network 160.10.0.0
neighbor 3.3.3.1 remote-as 300
neighbor 3.3.3.1 send-community
neighbor 3.3.3.1 route-map setcommunity out

route-map setcommunity
match ip address 1
set community no-export

access-list 1 permit 0.0.0.0 255.255.255.255
```

Note that we have used the route-map setcommunity in order to set the community to no-export. Note also that we had to use the "neighbor send-community" command in order to send this attribute to RTC.

When RTC gets the updates with the attribute no-export, it will not propagate them to its external peer RTA.

Example 2:

```
RTB#
router bgp 200
network 160.10.0.0
neighbor 3.3.3.1 remote-as 300
neighbor 3.3.3.1 send-community
neighbor 3.3.3.1 route-map setcommunity out

route-map setcommunity
match ip address 2
set community 100 200 additive

access-list 2 permit 0.0.0.0 255.255.255.255
```

In the above example, RTB has set the community attribute to 100 200 additive. The value 100 200 will be added to any existing community value before being sent to RTC.

A community list is a group of communities that we use in a **match** clause of a route map which allows us to do filtering or setting attributes based on different lists of community numbers.

```
ip community-list community-list-number {permit|deny} community-number
```

For example we can define the following route map, match-on-community:

```
route-map match-on-community
match community 10 (10 is the community-list number)
set weight 20

ip community-list 10 permit 200 300 (200 300 is the community number)
```

We can use the above in order to filter or set certain parameters like weight and metric based on the community value in certain updates. In example two above, RTB was sending updates to RTC with a community of 100 200. If RTC wants to set the weight based on those values we could do the following:

```

RTC#
router bgp 300
neighbor 3.3.3.3 remote-as 200
neighbor 3.3.3.3 route-map check-community in

route-map check-community permit 10
match community 1
set weight 20

route-map check-community permit 20
match community 2 exact
set weight 10

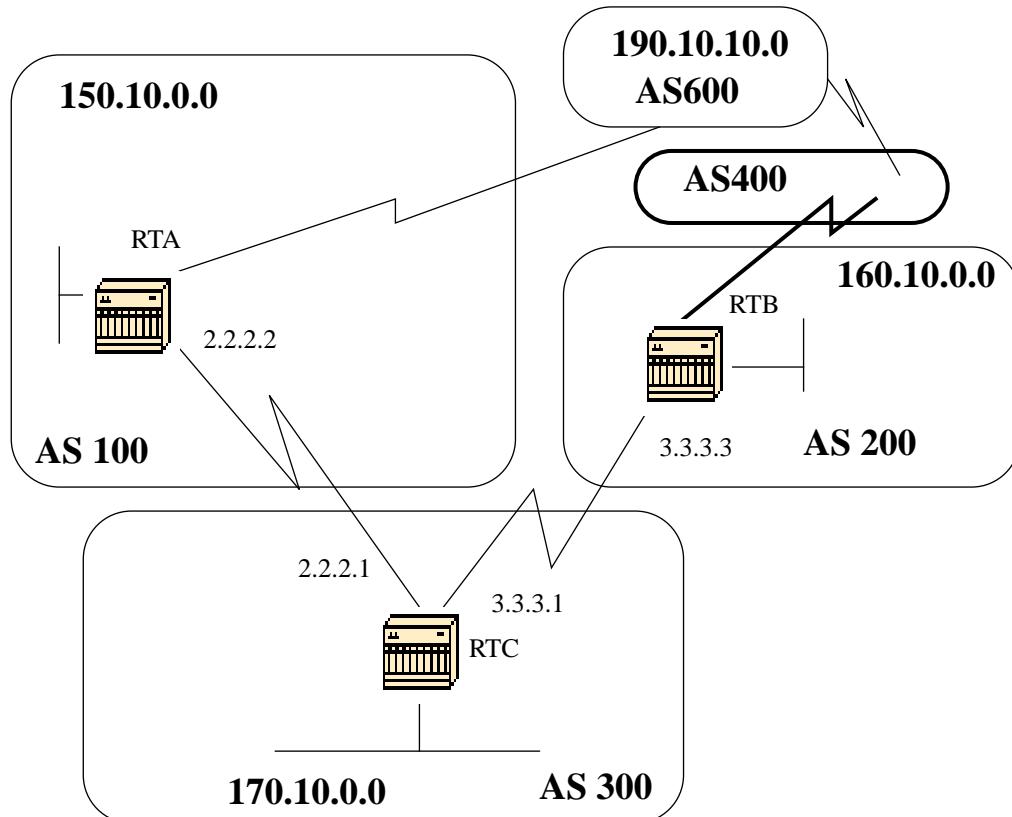
route-map check-community permit 30
match community 3

ip community-list 1 permit 100
ip community-list 2 permit 200
ip community-list 3 permit internet

```

In the above example, any route that has 100 in its community attribute will match list 1 and will have the weight set to 20. Any route that has only 200 as community will match list 2 and will have weight 20. The keyword **exact** states that community should consist of 200 only and nothing else. The last community list is here to make sure that other updates are not dropped. Remember that anything that does not match, will be dropped by default. The keyword **internet** means all routes because all routes are members of the internet community.

## 20.0 BGP Neighbors and Route maps



The neighbor command can be used in conjunction with route maps to perform either filtering or parameter setting on incoming and outgoing updates.

**Route maps associated with the neighbor statement have no affect on incoming updates when matching based on the IP address:**

```
neighbor ip-address route-map route-map-name
```

Assume in the above diagram we want RTC to learn from AS200 about networks that are local to AS200 and nothing else. Also, we want to set the weight on the accepted routes to 20. We can achieve this with a combination of neighbor and as-path access lists.

Example 1:

```
RTC#
router bgp 300
network 170.10.0.0
neighbor 3.3.3.3 remote-as 200
neighbor 3.3.3.3 route-map stamp in
```

```
route-map stamp
match as-path 1
set weight 20

ip as-path access-list 1 permit ^200$
```

Any updates that originate from AS200 have a path information that starts with 200 and ends with 200 and will be permitted. Any other updates will be dropped.

Example 2:

Assume that we want the following:

- 1- Updates originating from AS200 to be accepted with weight 20.
- 2- Updates originating from AS400 to be dropped.
- 3- Other updates to have a weight of 10.

```
RTC#
router bgp 300
network 170.10.0.0
neighbor 3.3.3.3 remote-as 200
neighbor 3.3.3.3 route-map stamp in

route-map stamp permit 10
match as-path 1
set weight 20

route-map stamp permit 20
match as-path 2
set weight 10

ip as-path access-list 1 permit ^200$
ip as-path access-list 2 permit ^200 600 .*
```

The above statement will set a weight of 20 for updates that are local to AS200, and will set a weight of 10 for updates that are behind AS400 and will drop updates coming from AS400.

## 20.1 Use of set as-path prepend

In some situations we are forced to manipulate the path information in order to manipulate the BGP decision process. The command that is used with a route map is:

```
set as-path prepend <as-path#> <as-path#> ...
```

Suppose in the above diagram that RTC is advertising its own network 170.10.0.0 to two different ASs: AS100 and AS200. When the information is propagated to AS600, the routers in AS600 will have network reachability information about 150.10.0.0 via two different routes, the first route is via AS100 with path (100, 300) and the second one is via AS400 with path (400, 200, 300). Assuming that all other attributes are the same, AS600 will pick the shortest path and will choose the route via AS100.

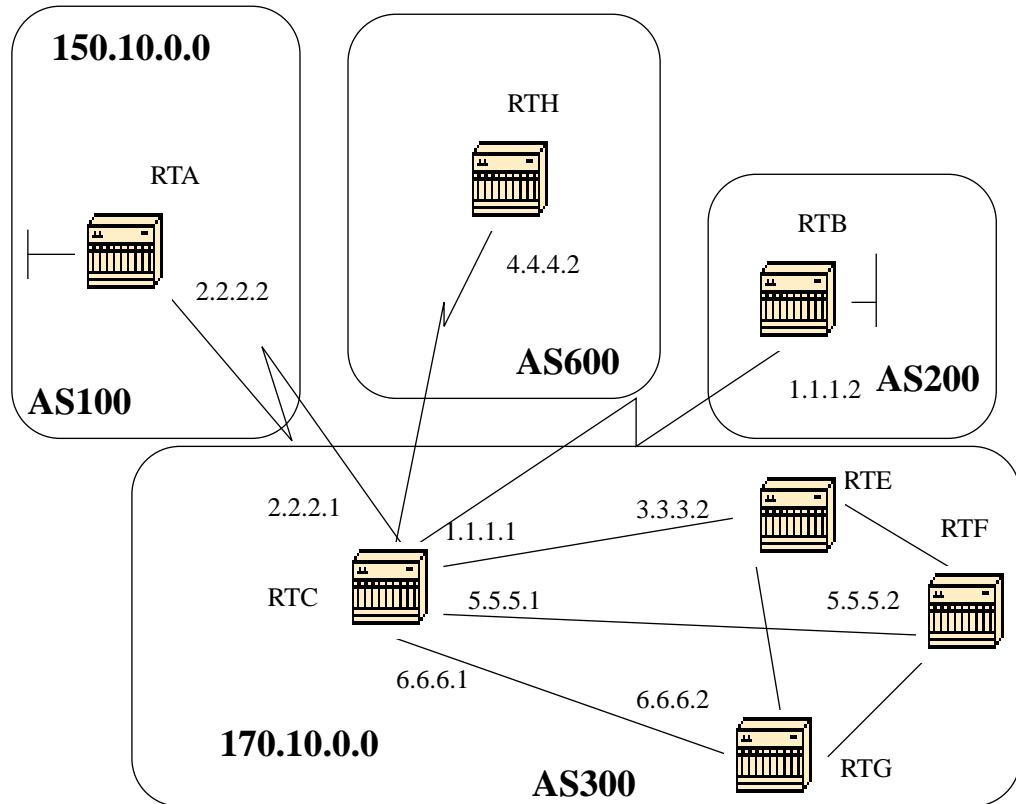
AS300 will be getting all its traffic via AS100. If we want to influence this decision from the AS300 end we can make the path through AS100 look like it is longer than the path going through AS400. We can do this by prepending autonomous system numbers to the existing path info advertised to AS100. A common practice is to repeat our own AS number using the following:

```
RTC#
router bgp 300
network 170.10.0.0
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 route-map SETPATH out

route-map SETPATH
set as-path prepend 300 300
```

Because of the above configuration, AS600 will receive updates about 170.10.0.0 via AS100 with a path information of: (100, 300, 300, 300) which is longer than (400, 200, 300) received from AS100.

## 20.2 BGP Peer Groups



A BGP peer group, is a group of BGP neighbors with the same update policies. Update policies are usually set by route maps, distribute-lists and filter-lists, etc. Instead of defining the same policies for each separate neighbor, we define a peer group name and we assign these policies to the peer group.

**Members of the peer group inherit all of the configuration options of the peer group. Members can also be configured to override these options if these options do not affect outbound updates; you can only override options set on the inbound.**

To define a peer group use the following:

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

In the following example we will see how peer groups are applied to internal and external BGP neighbors.

Example 1:

```
RTC#
router bgp 300
neighbor internalmap peer-group
neighbor internalmap remote-as 300
neighbor internalmap route-map SETMETRIC out
neighbor internalmap filter-list 1 out
neighbor internalmap filter-list 2 in
neighbor 5.5.5.2 peer-group internalmap
neighbor 6.6.6.2 peer-group internalmap
neighbor 3.3.3.2 peer-group internalmap
neighbor 3.3.3.2 filter-list 3 in
```

In the above configuration, we have defined a peer group named internalmap and we have defined some policies for that group, such as a route map SETMETRIC to set the metric to 5 and two different filter lists 1 and 2. We have applied the peer group to all internal neighbors RTE, RTF and RTG. We have defined a separate filter-list 3 for neighbor RTE, and this will override filter-list 2 inside the peer group. **Note that we could only override options that affect inbound updates.**

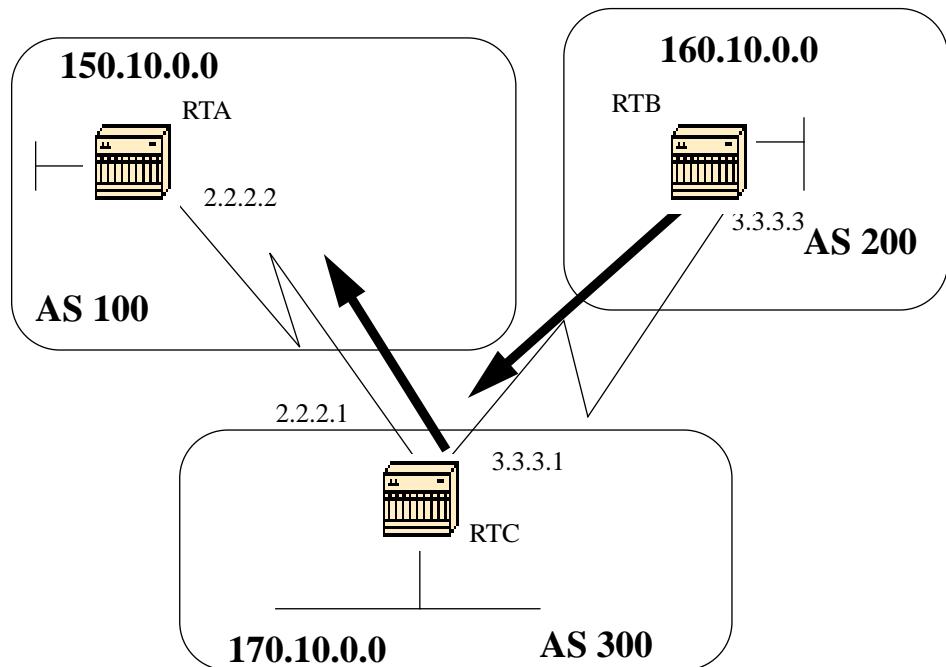
Now, let us look at how we can use peer groups with external neighbors. In the same diagram we will configure RTC with a peer-group externalmap and we will apply it to external neighbors.

Example 2:

```
RTC#
router bgp 300
neighbor externalmap peer-group
neighbor externalmap route-map SETMETRIC
neighbor externalmap filter-list 1 out
neighbor externalmap filter-list 2 in
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 peer-group externalmap
neighbor 4.4.4.2 remote-as 600
neighbor 4.4.4.2 peer-group externalmap
neighbor 1.1.1.2 remote-as 200
neighbor 1.1.1.2 peer-group externalmap
neighbor 1.1.1.2 filter-list 3 in
```

Note that in the above configs we have defined the remote-as statements outside of the peer group because we have to define different external ASs. Also we did an override for the inbound updates of neighbor 1.1.1.2 by assigning filter-list 3.

## 21.0 CIDR and Aggregate Addresses



One of the main enhancements of BGP4 over BGP3 is CIDR (Classless Interdomain Routing). CIDR or supernetting is a new way of looking at IP addresses. There is no notion of classes anymore (class A, B or C). For example, network 192.213.0.0 which used to be an illegal class C network is now a legal supernet represented by 192.213.0.0/16 where the 16 is the number of bits in the subnet mask counting from the far left of the IP address. This is similar to 192.213.0.0 255.255.0.0.

Aggregates are used to minimize the size of routing tables. Aggregation is the process of combining the characteristics of several different routes in such a way that a single route can be advertised. In the example below, RTB is generating network 160.10.0.0. We will configure RTC to propagate a supernet of that route 160.0.0.0 to RTA.

```
RTB#
router bgp 200
neighbor 3.3.3.1 remote-as 300
network 160.10.0.0
```

```
RTC#
router bgp 300
neighbor 3.3.3.3 remote-as 200
neighbor 2.2.2.2 remote-as 100
network 170.10.0.0
aggregate-address 160.0.0.0 255.0.0.0
```

RTC will propagate the aggregate address 160.0.0.0 to RTA.

## 21.1 Aggregate Commands

There is a wide range of aggregate commands. It is important to understand how each one works in order to have the desired aggregation behavior.

The first command is the one used in the previous example:

```
aggregate-address address mask
```

This will advertise the prefix route, and all of the more specific routes. The command **aggregate-address** 160.0.0.0 will propagate an additional network 160.0.0.0 but will not prevent 160.10.0.0 from being also propagated to RTA. The outcome of this is that both networks 160.0.0.0 and 160.10.0.0 have been propagated to RTA. This is what we mean by advertising the prefix and the more specific route.

**Please note that you can not aggregate an address if you do not have a more specific route of that address in the BGP routing table.**

For example, RTB can not generate an aggregate for 160.0.0.0 if it does not have a more specific entry of 160.0.0.0 in its BGP table. The more specific route could have been injected into the BGP table via incoming updates from other ASs, from redistributing an IGP or static into BGP or via the network command (network 160.10.0.0).

In case we would like RTC to propagate network 160.0.0.0 only and **NOT** the more specific route then we would have to use the following:

```
aggregate-address address mask summary-only
```

This will a advertise the prefix only; all the more specific routes are suppressed.

The command **aggregate** 160.0.0.0 255.0.0.0 **summary-only** will propagate network 160.0.0.0 and will **suppress the more specific route 160.10.0.0**.

Please note that if we are aggregating a network that is injected into our BGP via the network statement (ex: network 160.10.0.0 on RTB) then the network entry is always injected into BGP updates even though we are using the "aggregate summary-only" command. The upcoming CIDR example discusses this situation.

```
aggregate-address address mask as-set
```

This advertises the prefix and the more specific routes but it includes as-set information in the path information of the routing updates.  
ex: **aggregate 129.0.0.0 255.0.0.0 as-set.**

This will be discussed in an example by itself in the following sections.

In case we would like to suppress more specific routes when doing the aggregation we can define a route map and apply it to the aggregates. This will allow us to be selective about which more specific routes to suppress.

```
aggregate-address address-mask suppress-map map-name
```

This advertises the prefix and the more specific routes but it suppresses advertisement according to a route-map. In the previous diagram, if we would like to aggregate 160.0.0.0 and suppress the more specific route 160.20.0.0 and allow 160.10.0.0 to be propagated, we can use the following route map:

```
route-map CHECK permit 10
match ip address 1

access-list 1 deny 160.20.0.0 0.0.255.255
access-list 1 permit 0.0.0.0 255.255.255.255
```

Then we apply the route-map to the aggregate statement.

```
RTC#
router bgp 300
neighbor 3.3.3.3 remote-as 200
neighbor 2.2.2.2 remote-as 100
network 170.10.0.0
aggregate-address 160.0.0.0 255.0.0.0 suppress-map CHECK
```

Another variation is the:

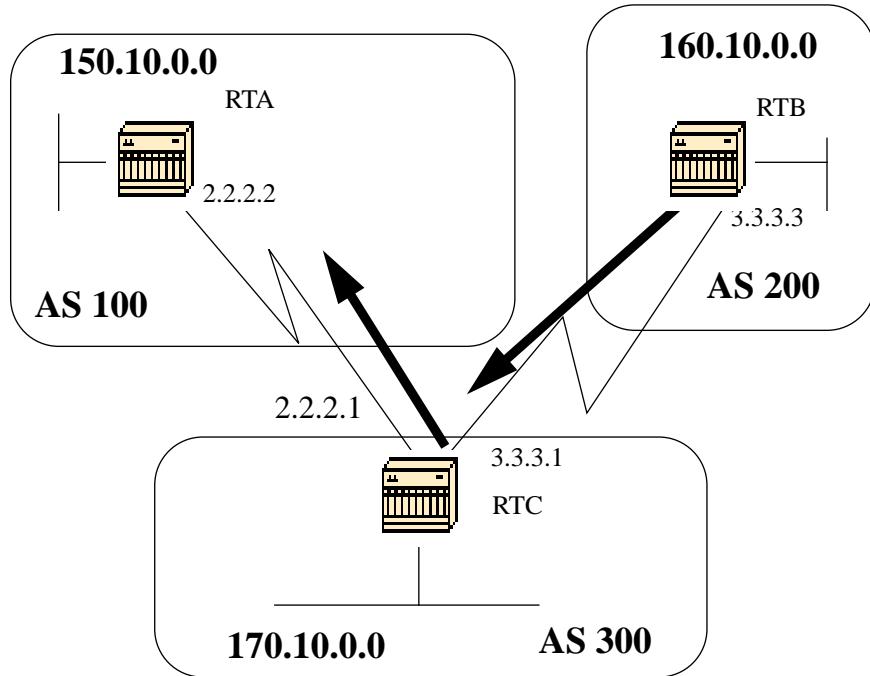
```
aggregate-address address mask attribute-map map-name
```

This allows us to set the attributes (metric, etc.) when aggregates are sent out. The following route map when applied to the aggregate attribute-map command will set the origin of the aggregates to IGP.

```
route-map SETMETRIC
set origin igrp

aggregate-address 160.0.0.0 255.0.0.0 attribute-map SETORIGIN
```

## 21.2 CIDR example 1



Request: Allow RTB to advertise the prefix 160.0.0.0 and suppress all the more specific routes. The problem here is that network 160.10.0.0 is local to AS200 i.e. AS200 is the originator of 160.10.0.0. You cannot have RTB generate a prefix for 160.0.0.0 without generating an entry for 160.10.0.0 even if you use the "aggregate summary-only" command because RTB is the originator of 160.10.0.0.

Solution 1:

The first solution is to use a static route and redistribute it into BGP. The outcome is that RTB will advertise the aggregate with an origin of incomplete (?).

```
RTB#
router bgp 200
neighbor 3.3.3.1 remote-as 300
redistribute static (This will generate an update for 160.0.0.0 with the
origin path as *incomplete*)

ip route 160.0.0.0 255.0.0.0 null0
```

Solution 2:

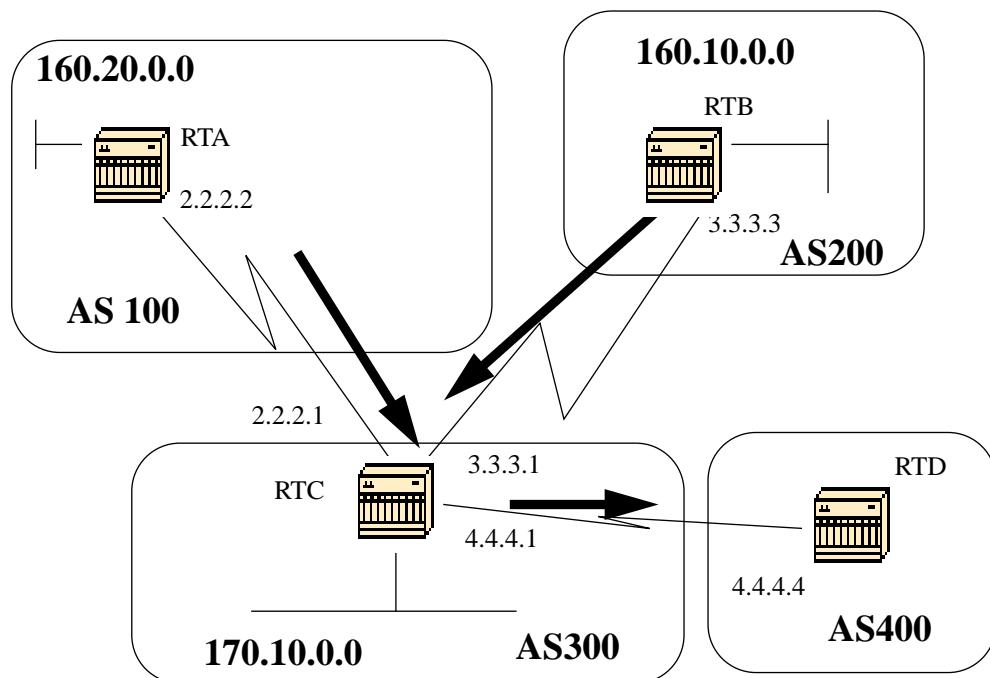
In addition to the static route we add an entry for the network command, this will have the same effect except that the origin of the update will be set to IGP.

```
RTB#
router bgp 200
network 160.0.0.0 mask 255.0.0.0 (this will mark the update with origin
IGP)
neighbor 3.3.3.1 remote-as 300
redistribute static

ip route 160.0.0.0 255.0.0.0 null0
```

### 21.3 CIDR example 2 (as-set)

AS-SETS are used in aggregation to reduce the size of the path information by listing the AS number only once, regardless of how many times it may have appeared in multiple paths that were aggregated. The **as-set** aggregate command is used in situations where aggregation of information causes loss of information regarding the path attribute. In the following example RTC is getting updates about 160.20.0.0 from RTA and updates about 160.10.0.0 from RTB. Suppose RTC wants to aggregate network 160.0.0.0/8 and send it to RTD. RTD would not know what the origin of that route is. By adding the aggregate **as-set** statement we force RTC to generate path information in the form of a set `{}`. All the path information is included in that set irrespective of which path came first.



```
RTB#
router bgp 200
network 160.10.0.0
neighbor 3.3.3.1 remote-as 300
```

```
RTA#
router bgp 100
network 160.20.0.0
neighbor 2.2.2.1 remote-as 300
```

Case 1:

RTC does not have an as-set statement. RTC will send an update 160.0.0.0/8 to RTD with path information (300) as if the route has originated from AS300.

```
RTC#
router bgp 300
neighbor 3.3.3.3 remote-as 200
neighbor 2.2.2.2 remote-as 100
neighbor 4.4.4.4 remote-as 400
aggregate 160.0.0.0 255.0.0.0 summary-only
(this causes RTC to send RTD updates about 160.0.0.0/8 with no indication
that 160.0.0.0 is actually coming from two different autonomous systems,
this may create loops if RT4 has an entry back into AS100)
```

Case 2:

```
RTC#
router bgp 300
neighbor 3.3.3.3 remote-as 200
neighbor 2.2.2.2 remote-as 100
neighbor 4.4.4.4 remote-as 400
aggregate 160.0.0.0 255.0.0.0 summary-only
aggregate 160.0.0.0 255.0.0.0 as-set
(causes RTC to send RTD updates about 160.0.0.0/8 with an indication that
160.0.0.0 belongs to a set {100 200})
```

The next two subjects, "confederation" and "route reflectors" are designed for ISPs who would like to further control the explosion of IBGP peering inside their autonomous systems.

## 22.0 BGP Confederation

BGP confederation is implemented in order to reduce the IBGP mesh inside an AS. The trick is to divide an AS into multiple ASs and assign the whole group to a single confederation. Each AS by itself will have IBGP fully meshed and has connections to other ASs inside the confederation. Even though these ASs will have EBGP peers to ASs within the confederation, they exchange routing as if they were using IBGP; next hop, metric and local preference information are preserved. To the outside world, the confederation (the group of ASs) will look as a single AS.

To configure a BGP confederation use the following:

```
bgp confederation identifier autonomous-system
```

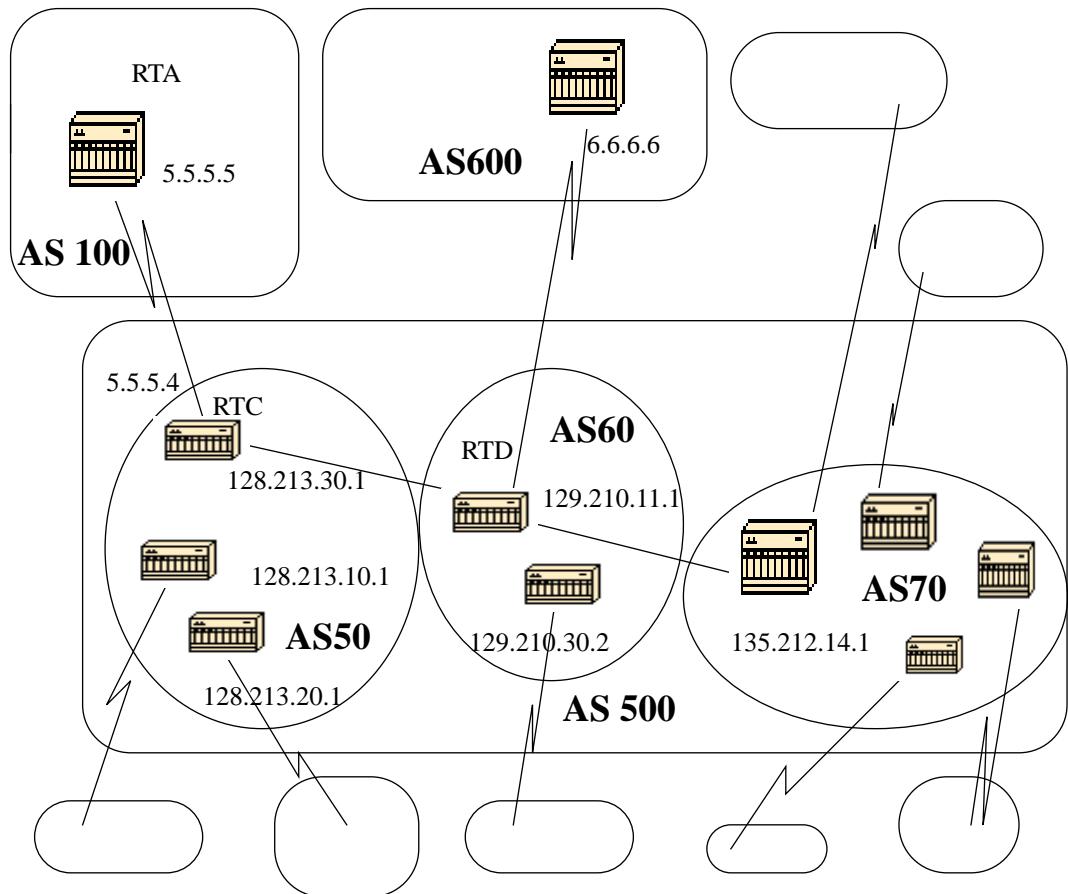
The confederation identifier will be the AS number of the confederation group. The group of ASs will look to the outside world as one AS with the AS number being the confederation identifier.

Peering within the confederation between multiple ASs is done via the following command:

```
bgp confederation peers autonomous-system [autonomous-system.]
```

The following is an example of confederation:

Example:



Let us assume that you have an autonomous system 500 consisting of nine BGP speakers (other non BGP speakers exist also, but we are only interested in the BGP speakers that have EBGP connections to other ASs). If you want to make a full IBGP mesh inside AS500 then you would need nine peer connections for each router, 8 IBGP peers and one EBGP peer to external ASs.

By using confederation we can divide AS500 into multiple ASs: AS50, AS60 and AS70. We give the AS a confederation identifier of 500. The outside world will see only one AS500. For each AS50, AS60 and AS70 we define a full mesh of IBGP peers and we define the list of confederation peers using the **bgp confederation peers** command.

I will show a sample configuration of routers RTC, RTD and RTA. Note that RTA has no knowledge of ASs 50, 60 or 70. RTA has only knowledge of AS500.

```
RTC#
router bgp 50
bgp confederation identifier 500
bgp confederation peers 60 70
neighbor 128.213.10.1 remote-as 50 (IBGP connection within AS50)
neighbor 128.213.20.1 remote-as 50 (IBGP connection within AS50)
neighbor 129.210.11.1 remote-as 60 (BGP connection with confederation
peer 60)
neighbor 135.212.14.1 remote-as 70 (BGP connection with confederation
peer 70)
neighbor 5.5.5.5 remote-as 100 (EBGP connection to external AS100)
```

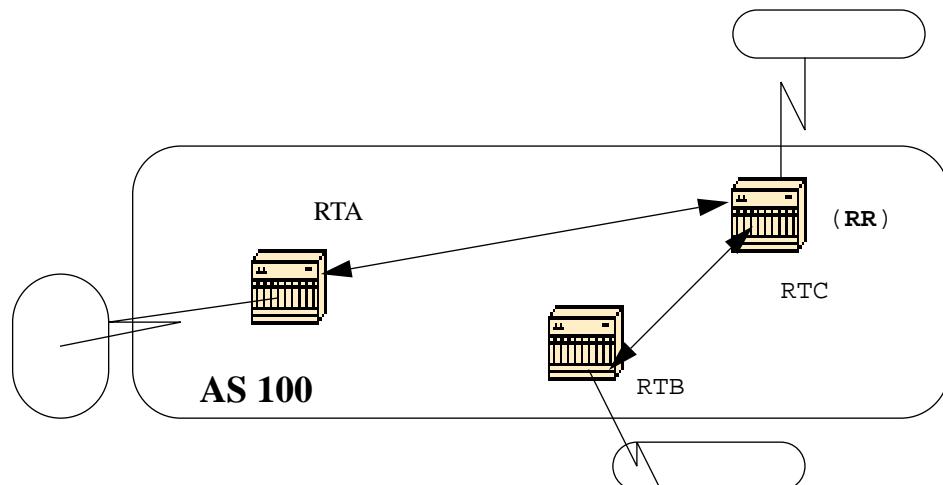
```
RTD#
router bgp 60
bgp confederation identifier 500
bgp confederation peers 50 70
neighbor 129.210.30.2 remote-as 60 (IBGP connection within AS60)
neighbor 128.213.30.1 remote-as 50(BGP connection with confederation
peer 50)
neighbor 135.212.14.1 remote-as 70 (BGP connection with confederation
peer 70)
neighbor 6.6.6.6 remote-as 600 (EBGP connection to external AS600)
```

```
RTA#
router bgp 100
neighbor 5.5.5.4 remote-as 500 (EBGP connection to confederation 500)
```

## 23.0 Route Reflectors

Another solution for the explosion of IBGP peering within an autonomous system is Route Reflectors (RR). As demonstrated in section 9.0 (Internal BGP), a BGP speaker will not advertise a route learned via another IBGP speaker to a third IBGP speaker. By relaxing this restriction a bit and by providing additional control, we can allow a router to advertise (reflect) IBGP learned routes to other IBGP speakers. This will reduce the number of IBGP peers within an AS.

Example:



In normal cases, a full IBGP mesh should be maintained between RTA, RTB and RTC within AS100. By utilizing the route reflector concept, RTC could be elected as a RR and have a partial IBGP peering with RTA and RTB. Peering between RTA and RTB is not needed because RTC will be a route reflector for the updates coming from RTA and RTB.

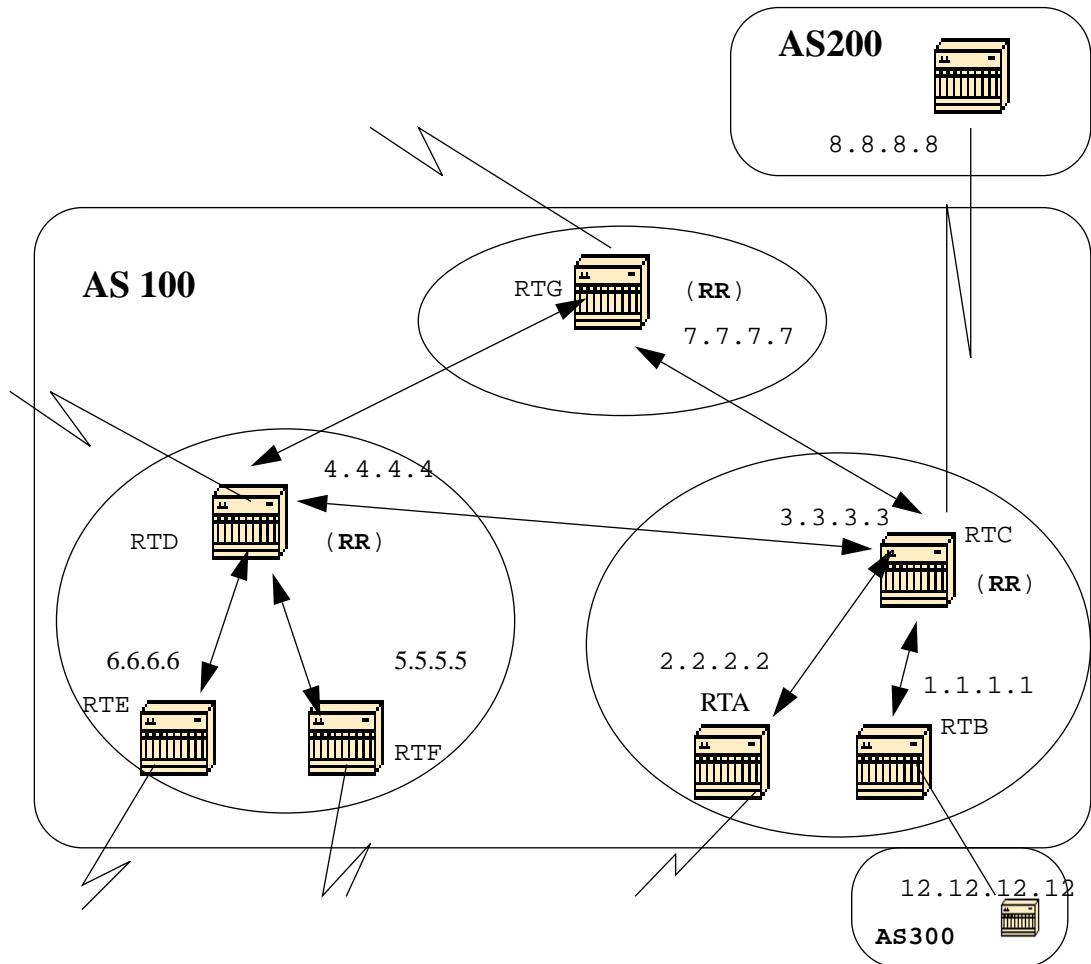
Configuring a route reflector is done using the following BGP router sub-command:

```
neighbor <ip-address> route-reflector-client
```

The router with the above command would be the RR and the neighbors pointed at would be the **clients** of that RR. In our example, RTC would be configured with the "neighbor route-reflector-client" command pointing at RTA and RTB's IP addresses. The combination of the RR and its clients is called a **cluster**. RTA, RTB and RTC above would form a cluster with a single RR within AS100.

Other IBGP peers of the RR that are not clients are called **non-clients**.

Example:



An autonomous system can have more than one route reflector; a RR would treat other RRs just like any other IBGP speaker. Other RRs could belong to the same cluster (client group) or to other clusters. In a simple configuration, the AS could be divided into multiple clusters, each RR will be configured with other RRs as non-client peers in a **fully meshed topology**. **Clients should not peer with IBGP speakers outside their cluster.**

Consider the above diagram. RTA, RTB and RTC form a single cluster with RTC being the RR. According to RTC, RTA and RTB are clients and anything else is a non-client. Remember that clients of an RR are pointed at using the "neighbor <ip-address> route-reflector-client" command. The same RTD is the RR for its clients RTE and RTF; RTG is a RR in a third cluster. Note that RTD, RTC and RTG are fully meshed but routers within a cluster are not. When a route is received by a RR, it will do the following depending on the peer type:

- 1- Route from a non-client peer: reflect to all the clients within the cluster.
- 2- Route from a client peer: reflect to all the non-client peers and also to the client peers.
- 3- Route from an EBGP peer: send the update to all client and non-client peers.

The following is the relative BGP configuration of routers RTC, RTD and RTB:

RTC#

```
router bgp 100
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 route-reflector-client
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 route-reflector-client
neighbor 7.7.7.7 remote-as 100
neighbor 4.4.4.4 remote-as 100
neighbor 8.8.8.8 remote-as 200
```

RTB#

```
router bgp 100
neighbor 3.3.3.3 remote-as 100
neighbor 12.12.12.12 remote-as 300
```

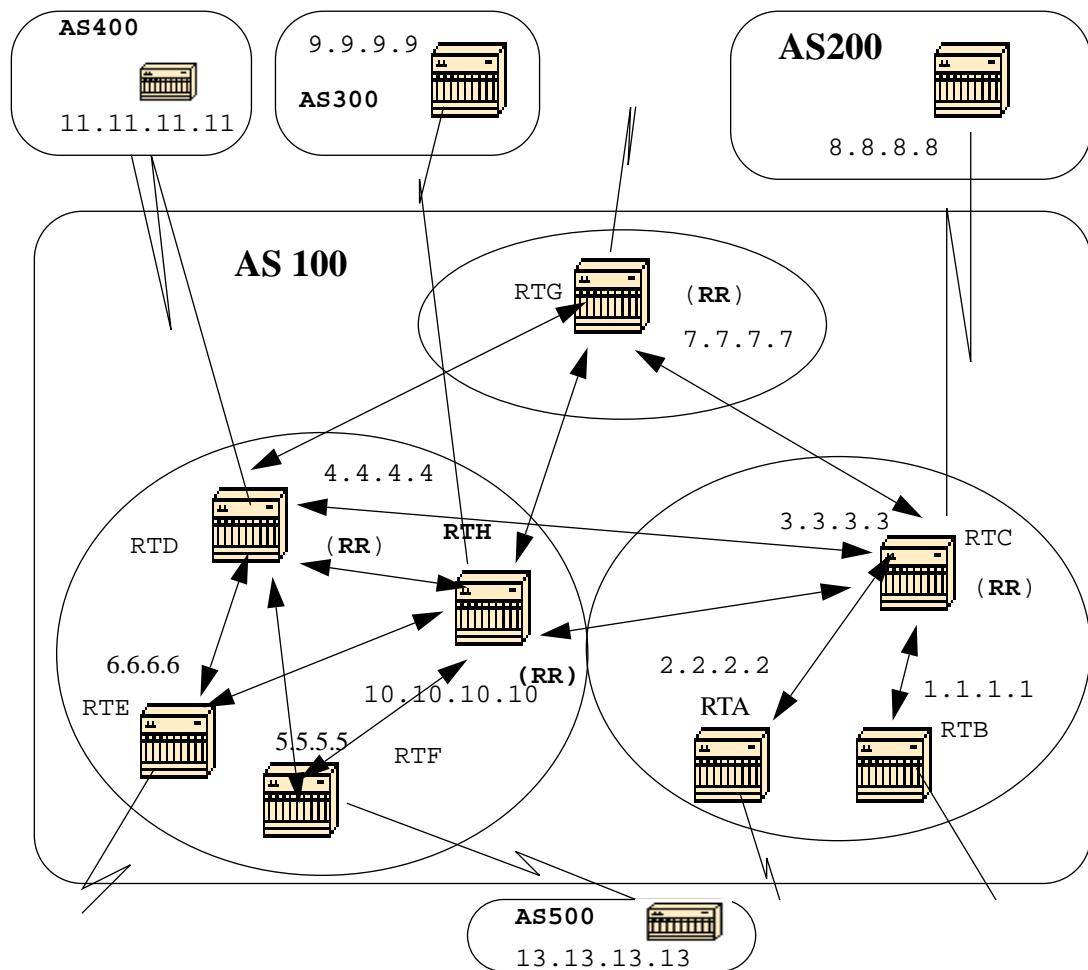
RTD#

```
router bgp 100
neighbor 6.6.6.6 remote-as 100
neighbor 6.6.6.6 route-reflector-client
neighbor 5.5.5.5 remote-as 100
neighbor 5.5.5.5 route-reflector-client
neighbor 7.7.7.7 remote-as 100
neighbor 3.3.3.3 remote-as 100
```

As the IBGP learned routes are reflected, it is possible to have the routing information loop. The Route-Reflector scheme has a few methods to avoid this:

- 1- Originator-id: this is an optional, non transitive BGP attribute that is four bytes long and is created by a RR. This attribute will carry the router-id (RID) of the originator of the route in the local AS. Thus, due to poor configuration, if the routing information comes back to the originator, it will be ignored.
- 2- Cluster-list: this will be discussed in the next section.

## 23.1 Multiple RRs within a cluster



Usually, a cluster of clients will have a single RR. In this case, the cluster will be identified by the **router-id** of the RR. In order to increase redundancy and avoid single points of failure, a cluster might have more than one RR. **All RRs in the same cluster need to be configured with a 4 byte cluster-id so that a RR can recognize updates from RRs in the same cluster.**

A **cluster-list** is a sequence of cluster-ids that the route has passed. When a RR reflects a route from its clients to non-clients outside of the cluster, it will append the local cluster-id to the cluster-list. If this update has an empty cluster-list the RR will create one. Using this attribute, a RR can identify if the routing information is looped back to the same cluster due to poor configuration. If the local cluster-id is found in the cluster-list, the advertisement will be ignored.

In the above diagram, RTD, RTE, RTF and RTH belong to one cluster with both RTD and RTH being RRs for the same cluster. Note the redundancy in that RTH has a fully meshed peering with all the RRs. In case RTD goes down, RTH will take its place. The following are the configuration of RTH, RTD, RTF and RTC:

```
RTH#
```

```
router bgp 100
neighbor 4.4.4.4 remote-as 100
neighbor 5.5.5.5 remote-as 100
neighbor 5.5.5.5 route-reflector-client
neighbor 6.6.6.6 remote-as 100
neighbor 6.6.6.6 route-reflector-client
neighbor 7.7.7.7 remote-as 100
neighbor 3.3.3.3 remote-as 100
neighbor 9.9.9.9 remote-as 300
bgp route-reflector 10 (This is the cluster-id)
```

```
RTD#
```

```
router bgp 100
neighbor 10.10.10.10 remote-as 100
neighbor 5.5.5.5 remote-as 100
neighbor 5.5.5.5 route-reflector-client
neighbor 6.6.6.6 remote-as 100
neighbor 6.6.6.6 route-reflector-client
neighbor 7.7.7.7 remote-as 100
neighbor 3.3.3.3 remote-as 100
neighbor 11.11.11.11 remote-as 400
bgp route-reflector 10 (This is the cluster-id)
```

```
RTF#
```

```
router bgp 100
neighbor 10.10.10.10 remote-as 100
neighbor 4.4.4.4 remote-as 100
neighbor 13.13.13.13 remote-as 500
```

```
RTC#
```

```
router bgp 100
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 route-reflector-client
neighbor 2.2.2.2 remote-as 100
neighbor 2.2.2.2 route-reflector-client
neighbor 4.4.4.4 remote-as 100
neighbor 7.7.7.7 remote-as 100
neighbor 10.10.10.10 remote-as 100
neighbor 8.8.8.8 remote-as 200
```

Note that we did not need the cluster command for RTC because only one RR exists in that cluster.

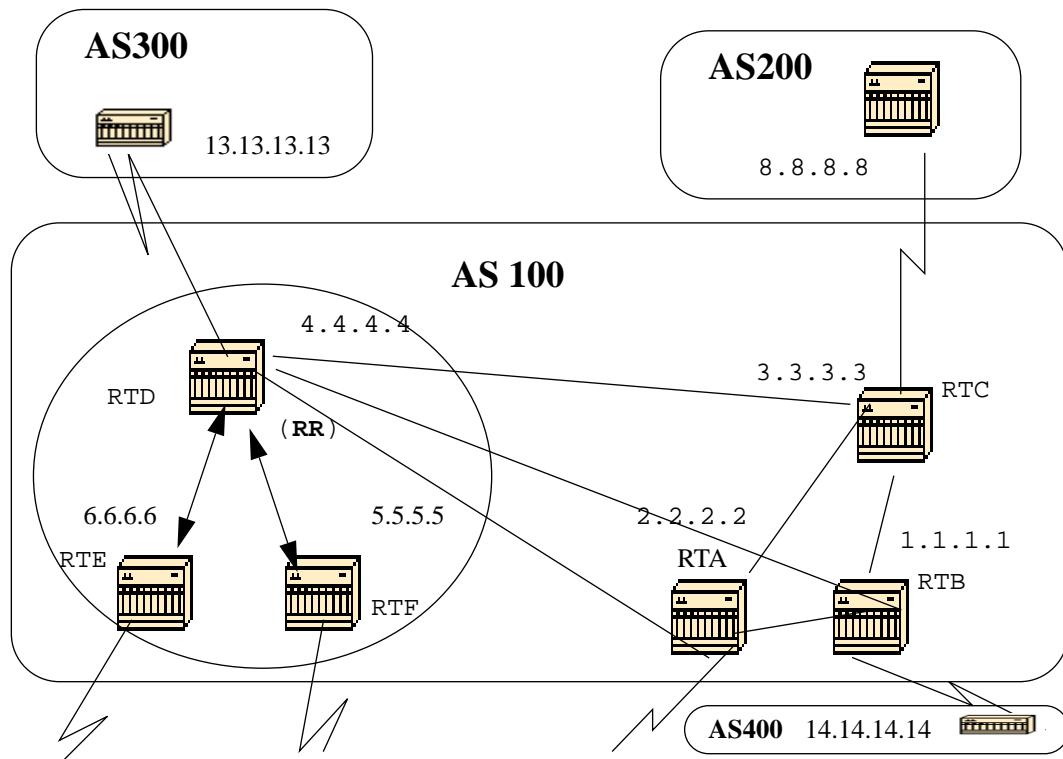
An important thing to note, is that **peer-groups were not used in the above configuration**. If the clients inside a cluster do not have direct IBGP peers among one another and they exchange updates through the RR, peer-groups should not be used. If peer groups were to be configured, then a potential withdrawal to the source of a route on the RR would be sent to all clients inside the cluster and could cause problems.

The router sub-command **`bgp client-to-client reflection`** is enabled by default on the RR. If BGP client-to-client reflection were turned off on the RR and redundant BGP peering was made between the clients, then using peer groups would be alright.

## 23.2 RR and conventional BGP speakers

It is normal in an AS to have BGP speakers that do not understand the concept of route reflectors. We will call these routers conventional BGP speakers. The route reflector scheme will allow such conventional BGP speakers to coexist. These routers could be either members of a client group or a non-client group. This would allow easy and gradual migration from the current IBGP model to the route reflector model. One could start creating clusters by configuring a single router as RR and making other RRs and their clients normal IBGP peers. Then more clusters could be created gradually.

Example:



In the above diagram, RTD, RTE and RTF have the concept of route reflection. RTC, RTA and RTB are what we call conventional routers and cannot be configured as RRs. Normal IBGP mesh could be done between these routers and RTD. Later on, when we are ready to upgrade, RTC could be made a RR with clients RTA and RTB. Clients do not have to understand the route reflection scheme; it is only the RRs that would have to be upgraded.

The following is the configuration of RTD and RTC:

RTD#

```
router bgp 100
neighbor 6.6.6.6 remote-as 100
neighbor 6.6.6.6 route-reflector-client
neighbor 5.5.5.5 remote-as 100
neighbor 5.5.5.5 route-reflector-client
neighbor 3.3.3.3 remote-as 100
neighbor 2.2.2.2 remote-as 100
neighbor 1.1.1.1 remote-as 100
neighbor 13.13.13.13 remote-as 300
```

RTC#

```
router bgp 100
neighbor 4.4.4.4 remote-as 100
neighbor 2.2.2.2 remote-as 100
neighbor 1.1.1.1 remote-as 100
neighbor 14.14.14.14 remote-as 400
```

When we are ready to upgrade RTC and make it a RR, we would remove the IBGP full mesh and have RTA and RTB become clients of RTC.

### 23.3 Avoiding looping of routing information

We have mentioned so far two attributes that are used to prevent potential information looping: the **originator-id** and the **cluster-list**.

Another means of controlling loops is to put more **restrictions on the set clause of out-bound route-maps**.

**The set clause for out-bound route-maps does not affect routes reflected to IBGP peers.**

More restrictions are also put on the **nexthop-self** which is a per neighbor configuration option. When used on RRs the **nexthop-self will only affect the nexthop of EBGP learned routes because the nexthop of reflected routes should not be changed.**

## 24.0 Route Flap Dampening

Route dampening (introduced in Cisco IOS version 11.0) is a mechanism to minimize the instability caused by route flapping and oscillation over the network. To accomplish this, criteria are defined to identify poorly behaved routes. A route which is flapping gets a penalty for each flap (1000). As soon as the cumulative penalty reaches a predefined “suppress-limit”, the advertisement of the route will be suppressed. The penalty will be exponentially decayed based on a preconfigured “half-time”. Once the penalty decreases below a predefined “reuse-limit”, the route advertisement will be un-suppressed.

Routes, external to an AS, learned via IBGP will not be dampened. This is to avoid the IBGP peers having higher penalty for routes external to the AS.

The penalty will be decayed at a granularity of 5 seconds and the routes will be un-suppressed at a granularity of 10 seconds. The dampening information is kept until the penalty becomes less than half of “reuse-limit”, at that point the information is purged from the router.

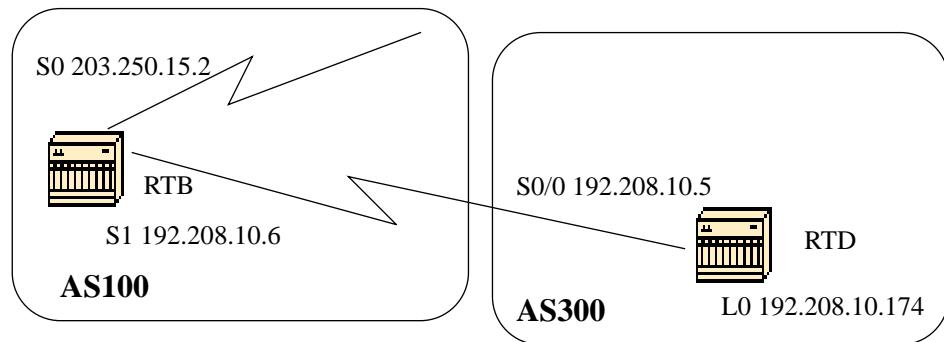
Initially, dampening will be off by default. This might change if there is a need to have this feature enabled by default. The following are the commands used to control route dampening:

|                                             |                              |
|---------------------------------------------|------------------------------|
| <b>bgp dampening</b>                        | - will turn on dampening.    |
| <b>no bgp dampening</b>                     | - will turn off dampening.   |
| <b>bgp dampening &lt;half-life-time&gt;</b> | - change the half-life time. |

A command that sets all parameters at the same time is:

|                                                                                                          |                                                                                                   |
|----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| <b>bgp dampening &lt;half-life-time&gt; &lt;reuse&gt; &lt;suppress&gt; &lt;maximum-suppress-time&gt;</b> |                                                                                                   |
| <b>&lt;half-life-time&gt;</b>                                                                            | range is 1-45 min, current default is 15 min.                                                     |
| <b>&lt;reuse-value&gt;</b>                                                                               | range is 1-20000, default is 750.                                                                 |
| <b>&lt;suppress-value&gt;</b>                                                                            | range is 1-20000, default is 2000.                                                                |
| <b>&lt;max-suppress-time&gt;</b>                                                                         | maximum duration a route can be suppressed,<br>range is 1-255, default is 4 times half-life-time. |

Example:



```
RTB#
hostname RTB
```

```
interface Serial0
ip address 203.250.15.2 255.255.255.252
```

```
interface Serial1
ip address 192.208.10.6 255.255.255.252
```

```
router bgp 100
bgp dampening
network 203.250.15.0
neighbor 192.208.10.5 remote-as 300
```

```
RTD#
hostname RTD
```

```
interface Loopback0
ip address 192.208.10.174 255.255.255.192
```

```
interface Serial0/0
ip address 192.208.10.5 255.255.255.252
```

```
router bgp 300
network 192.208.10.0
neighbor 192.208.10.6 remote-as 100
```

RTB is configured for route dampening with default parameters. Assuming the EBGP link to RTD is stable, RTB's BGP table would look like this:

```

RTB#sh ip bgp

BGP table version is 24, local router ID is 203.250.15.2 Status codes: s
suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

 Network Next Hop Metric LocPrf Weight Path
* > 192.208.10.0 192.208.10.5 0 0 300 i
* > 203.250.15.0 0.0.0.0 0 32768 i

```

In order to simulate a route flap, I will do a "clear ip bgp 192.208.10.6" on RTD. RTB's BGP table will look like this:

```

RTB#sh ip bgp

BGP table version is 24, local router ID is 203.250.15.2 Status codes: s
suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

 Network Next Hop Metric LocPrf Weight Path
h 192.208.10.0 192.208.10.5 0 0 300 i
* > 203.250.15.0 0.0.0.0 0 32768 i

```

The BGP entry for 192.208.10.0 has been put in a "history" state. Which means that we do not have a best path to the route but information about the route flapping still exists.

```

RTB#sh ip bgp 192.208.10.0

BGP routing table entry for 192.208.10.0 255.255.255.0, version 25
Paths: (1 available, no best path)
 300 (history entry)
 192.208.10.5 from 192.208.10.5 (192.208.10.174)
 Origin IGP, metric 0, external
 Dampinfo: penalty 910, flapped 1 times in 0:02:03

```

The route has been given a penalty for flapping but the penalty is still below the "suppress limit" (default is 2000). The route is not yet suppressed. If the route flaps few more times we will see the following:

```
RTB#sh ip bgp
BGP table version is 32, local router ID is 203.250.15.2 Status codes:
s suppressed, d damped, h history, * valid, > best, i - internal Origin
codes: i - IGP, e - EGP, ? - incomplete
```

| Network                   | Next Hop            | Metric   | LocPrf       | Weight     | Path     |
|---------------------------|---------------------|----------|--------------|------------|----------|
| <b>*d 192.208.10.0</b>    | <b>192.208.10.5</b> | <b>0</b> | <b>0</b>     | <b>300</b> | <b>i</b> |
| <b>*&gt; 203.250.15.0</b> | <b>0.0.0.0</b>      | <b>0</b> | <b>32768</b> | <b>i</b>   |          |

```
RTB#sh ip bgp 192.208.10.0
BGP routing table entry for 192.208.10.0 255.255.255.0, version 32
Paths: (1 available, no best path)
300, (suppressed due to dampening)
192.208.10.5 from 192.208.10.5 (192.208.10.174)
 Origin IGP, metric 0, valid, external
Dampinfo: penalty 2615, flapped 3 times in 0:05:18 , reuse in 0:27:00
```

The route has been dampened (suppressed). The route will be reused when the penalty reaches the "reuse value", in our case 750 (default). The dampening information will be purged when the penalty becomes less than half of the reuse-limit, in our case (750/2=375). The Following are the commands used to show and clear flap statistics information:

```
show ip bgp flap-statistics
(displays flap statistics for all the paths)

show ip bgp-flap-statistics regexp <regexp>
(displays flap statistics for all paths that match the regexp)

show ip bgp flap-statistics filter-list <list>
(displays flap statistics for all paths that pass the filter)

show ip bgp flap-statistics A.B.C.D m.m.m.m
(displays flap statistics for a single entry)

show ip bgp flap-statistics A.B.C.D m.m.m.m longer-prefixes
(displays flap statistics for more specific entries)

show ip bgp neighbor [dampened-routes] | [flap-statistics]
(displays flap statistics for all paths from a neighbor)

clear ip bgp flap-statistics (clears flap statistics for all routes)

clear ip bgp flap-statistics regexp <regexp>
(clears flap statistics for all the paths that match the regexp)

clear ip bgp flap-statistics filter-list <list>
(clears flap statistics for all the paths that pass the filter)

clear ip bgp flap-statistics A.B.C.D m.m.m.m
(clears flap statistics for a single entry)

clear ip bgp A.B.C.D flap-statistics
(clears flap statistics for all paths from a neighbor)
```

## 25.0 How BGP selects a Path

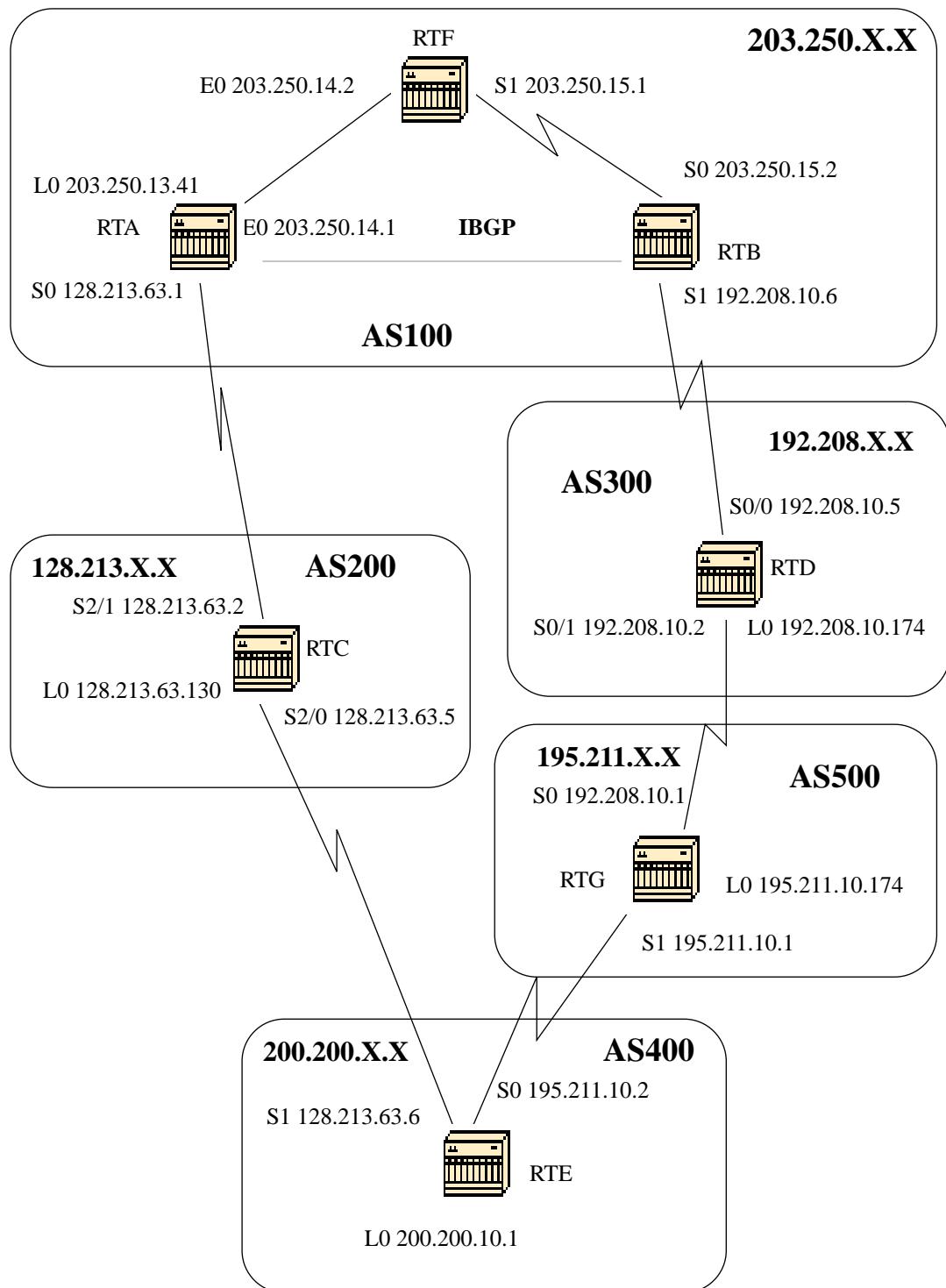
Now that we are familiar with the BGP attributes and terminology, the following list indicates how BGP selects the best path for a particular destination. Remember that we only select one path as the best path. We put that path in our routing table and we propagate it to our BGP neighbors.

Path selection is based on the following:

- 1-If **NextHop is inaccessible** do not consider it.
- 2-Prefer the largest **Weight**.
- 3-If same weight prefer largest **Local Preference**.
- 4-If same Local Preference prefer **the route** that the specified **router has originated**.
- 5-If no route was originated prefer the **shorter AS path**.
- 6-If all paths are external prefer the **lowest origin code (IGP<EGP<INCOMPLETE)**.
- 7-If origin codes are the same prefer the path with the **lowest MULTI\_EXIT\_DISC**.
- 8-If path is the same length prefer **External path over Internal**.
- 9-If IGP synchronization is disabled and only internal path remain prefer the path through the **closest IGP neighbor**.
- 10-Prefer the route with the **lowest ip address** value for BGP router ID.

The ollowing is a design example thatis intended to show the configuration and routing tables as they actually appear on the C.isco routers.

## 26.0 Practical design example:



We will build the above configuration step by step and see what can go wrong along the way. Whenever you have an AS that is connected to two ISPs via EBGP, it is always good to run IBGP within your AS in order to have a better control of your routes. In this example we will run IBGP inside AS100 between RTA and RTB and we will run OSPF as an IGP.

Assuming that AS200 and AS300 are the two ISPs we are connected to, the following are the first run of configuration for all the routers. **This is NOT the final configuration.**

```
RTA#
hostname RTA

ip subnet-zero

interface Loopback0
 ip address 203.250.13.41 255.255.255.0

interface Ethernet0
 ip address 203.250.14.1 255.255.255.0

interface Serial0
 ip address 128.213.63.1 255.255.255.252

router ospf 10
 network 203.250.0.0 0.0.255.255 area 0

router bgp 100
 network 203.250.0.0 mask 255.255.0.0
 neighbor 128.213.63.2 remote-as 200
 neighbor 203.250.15.2 remote-as 100
 neighbor 203.250.15.2 update-source Loopback0
```

```
RTF#
hostname RTF

ip subnet-zero

interface Ethernet0
 ip address 203.250.14.2 255.255.255.0

interface Serial1
 ip address 203.250.15.1 255.255.255.252

router ospf 10
 network 203.250.0.0 0.0.255.255 area 0
```

```
RTB#
hostname RTB

ip subnet-zero

interface Serial0
 ip address 203.250.15.2 255.255.255.252

interface Serial1
 ip address 192.208.10.6 255.255.255.252

router ospf 10
 network 203.250.0.0 0.0.255.255 area 0

router bgp 100
network 203.250.15.0
 neighbor 192.208.10.5 remote-as 300
 neighbor 203.250.13.41 remote-as 100
```

```
RTC#
hostname RTC

ip subnet-zero

interface Loopback0
 ip address 128.213.63.130 255.255.255.192

interface Serial2/0
 ip address 128.213.63.5 255.255.255.252
!
interface Serial2/1
 ip address 128.213.63.2 255.255.255.252

router bgp 200
network 128.213.0.0
neighbor 128.213.63.1 remote-as 100
neighbor 128.213.63.6 remote-as 400
```

```
RTD#
hostname RTD

ip subnet-zero

interface Loopback0
 ip address 192.208.10.174 255.255.255.192

interface Serial0/0
 ip address 192.208.10.5 255.255.255.252
!
interface Serial0/1
 ip address 192.208.10.2 255.255.255.252

router bgp 300
 network 192.208.10.0
 neighbor 192.208.10.1 remote-as 500
 neighbor 192.208.10.6 remote-as 100
```

```
RTE#
hostname RTE

ip subnet-zero

interface Loopback0
 ip address 200.200.10.1 255.255.255.0

interface Serial0
 ip address 195.211.10.2 255.255.255.252

interface Serial1
 ip address 128.213.63.6 255.255.255.252
 clockrate 1000000

router bgp 400
 network 200.200.10.0
 neighbor 128.213.63.5 remote-as 200
 neighbor 195.211.10.1 remote-as 500
```

```

RTG#
hostname RTG

ip subnet-zero

interface Loopback0
 ip address 195.211.10.174 255.255.255.192

interface Serial0
 ip address 192.208.10.1 255.255.255.252

interface Serial1
 ip address 195.211.10.1 255.255.255.252

router bgp 500
 network 195.211.10.0
 neighbor 192.208.10.2 remote-as 300
 neighbor 195.211.10.2 remote-as 400

```

It is always better to use the network command or redistribute static entries into BGP to advertise networks, rather than redistributing IGP into BGP.

This is why, throughout this example I will only use the network command to inject networks into BGP.

**Let us assume to start with that s1 on RTB is shutdown, as if the link between RTB and RTD does not exist. The following is RTB's BGP table.**

```

RTB#sh ip bgp BGP
table version is 4, local router ID is 203.250.15.2 Status
codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
 Network Next Hop Metric LocPrf Weight Path
*i128.213.0.0 128.213.63.2 0 100 0 200 i
*i192.208.10.0 128.213.63.2 100 0 200 400 500
300 i
*i195.211.10.0 128.213.63.2 100 0 200 400 500 i
*i200.200.10.0 128.213.63.2 100 0 200 400 i
*>i203.250.13.0 203.250.13.41 0 100 0 i
*>i203.250.14.0 203.250.13.41 0 100 0 i
*>203.250.15.0 0.0.0.0 0 32768 i

```

Let me go over the basic notations of the above table. The "i" at the beginning means that the entry was learned via an internal BGP peer. The "i" at the end indicates the ORIGIN of the path information to be IGP. The path info is intuitive. For example network 128.213.0.0 is learned via path 200 with nexthop of 128.213.63.2. Note that any locally generated entry such as 203.250.15.0 has a nexthop 0.0.0.0.

The > symbol indicates that BGP has chosen the best route based on the list of decision steps that I have gone through earlier in this document under "How BGP selects a Path". Bgp will only pick one best Path to reach a destination, will install this path in the ip routing table and will advertise it to other bgp peers. Notice the nexthop attribute. RTB knows about 128.213.0.0 via a nexthop of 128.213.63.2 which is the ebgp nexthop carried into IBGP.

Let us look at the IP routing table:

```
RTB#sh ip rou
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
 i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate
 default
```

Gateway of last resort is not set

```
 203.250.13.0 255.255.255.255 is subnetted, 1 subnets
O 203.250.13.41 [110/75] via 203.250.15.1, 02:50:45, Serial0
 203.250.15.0 255.255.252 is subnetted, 1 subnets
C 203.250.15.0 is directly connected, Serial0
O 203.250.14.0 [110/74] via 203.250.15.1, 02:50:46, Serial0
```

Well, it doesn't look like any of the BGP entries has made it to the routing table. There are two problems here:

Problem 1:

The Nexthop for these entries 128.213.63.2 is unreachable. This is true because we do not have a way to reach that nexthop via our IGP (OSPF). RTB has not learned about 128.213.63.0 via OSPF. We can run OSPF on RTA s0 and make it passive, and this way RTB would know how to reach the nexthop 128.213.63.2. **We could also change the nexthop by using the bgp nexthopself command between RTA and RTB.** RTA's configs would be:

```

RTA#
hostname RTA

ip subnet-zero

interface Loopback0
 ip address 203.250.13.41 255.255.255.0

interface Ethernet0
 ip address 203.250.14.1 255.255.255.0

interface Serial0
 ip address 128.213.63.1 255.255.255.252

router ospf 10
 passive-interface Serial0
 network 203.250.0.0 0.0.255.255 area 0
 network 128.213.0.0 0.0.255.255 area 0

router bgp 100
 network 203.250.0.0 mask 255.255.0.0
 neighbor 128.213.63.2 remote-as 200
 neighbor 203.250.15.2 remote-as 100
 neighbor 203.250.15.2 update-source Loopback0

```

The new BGP table on RTB now looks like this:

```

RTB#sh ip bgp
BGP table version is 10, local router ID is 203.250.15.2
Status codes: s suppressed, d damped, h history, * valid, > best,
i - internal Origin codes: i - IGP, e - EGP, ? - incomplete

 Network Next Hop Metric LocPrf Weight Path
*->i128.213.0.0 128.213.63.2 0 100 0 200 i
*->i192.208.10.0 128.213.63.2 100 100 0 200 400 500
300 i
*->i195.211.10.0 128.213.63.2 100 100 0 200 400 500 i
*->i200.200.10.0 128.213.63.2 100 100 0 200 400 i
*>i203.250.13.0 203.250.13.41 0 100 0 i
*>i203.250.14.0 203.250.13.41 0 100 0 i
*> 203.250.15.0 0.0.0.0 0 32768 i

```

Note that all the entries have >, which means that BGP is ok with next hop. Let us look at the routing table now:

```

RTB#sh ip rou
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
 i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * -
candidate default

Gateway of last resort is not set

 203.250.13.0 255.255.255.255 is subnetted, 1 subnets
O 203.250.13.41 [110/75] via 203.250.15.1, 00:04:46, Serial0
 203.250.15.0 255.255.255.252 is subnetted, 1 subnets
C 203.250.15.0 is directly connected, Serial0
O 203.250.14.0 [110/74] via 203.250.15.1, 00:04:46, Serial0
 128.213.0.0 255.255.255.252 is subnetted, 1 subnets
O 128.213.63.0 [110/138] via 203.250.15.1, 00:04:47, Serial0

```

Problem 2:

We still do not see the BGP entries; the only difference is that 128.213.63.0 is now reachable via OSPF. This is the synchronization issue, BGP is not putting these entries in the routing table and will not send them in BGP updates because it is not synchronized with the IGP. Note that RTF has no notion of networks 192.208.10.0 or 195.211.10.0 because we have not redistributed BGP into OSPF yet.

In this scenario, if we turn synchronization off, we will have the entries in the routing table, but connectivity would still be broken.

If you turn off synchronization on RTB this is what will happen:

```
RTB#sh ip rou
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
 i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * -
candidate default
```

Gateway of last resort is not set

```
B 200.200.10.0 [200/0] via 128.213.63.2, 00:01:07
B 195.211.10.0 [200/0] via 128.213.63.2, 00:01:07
B 192.208.10.0 [200/0] via 128.213.63.2, 00:01:07
 203.250.13.0 is variably subnetted, 2 subnets, 2 masks
O 203.250.13.41 255.255.255.255
 [110/75] via 203.250.15.1, 00:12:37, Serial0
B 203.250.13.0 255.255.255.0 [200/0] via 203.250.13.41, 00:01:08
 203.250.15.0 255.255.255.252 is subnetted, 1 subnets
C 203.250.15.0 is directly connected, Serial0
O 203.250.14.0 [110/74] via 203.250.15.1, 00:12:37, Serial0
 128.213.0.0 is variably subnetted, 2 subnets, 2 masks
B 128.213.0.0 255.255.0.0 [200/0] via 128.213.63.2, 00:01:08
O 128.213.63.0 255.255.255.252
 [110/138] via 203.250.15.1, 00:12:37, Serial0
```

The routing table looks fine, but there is no way we can reach those networks because RTF in the middle does not know how to reach them:

```
RTF#sh ip rou
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
 i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * -
candidate default
```

Gateway of last resort is not set

```
 203.250.13.0 255.255.255.255 is subnetted, 1 subnets
O 203.250.13.41 [110/11] via 203.250.14.1, 00:14:15, Ethernet0
 203.250.15.0 255.255.255.252 is subnetted, 1 subnets
C 203.250.15.0 is directly connected, Serial1
C 203.250.14.0 is directly connected, Ethernet0
 128.213.0.0 255.255.255.252 is subnetted, 1 subnets
O 128.213.63.0 [110/74] via 203.250.14.1, 00:14:15, Ethernet0
```

So, turning off synchronization in this situation did not help this particular issue, but we will need it for other issues later on.

Let's redistribute OSPF into BGP on RTA , with a metric of 2000.

```

RTA#
hostname RTA

ip subnet-zero

interface Loopback0
 ip address 203.250.13.41 255.255.255.0

interface Ethernet0
 ip address 203.250.14.1 255.255.255.0

interface Serial0
 ip address 128.213.63.1 255.255.255.252

router ospf 10
 redistribute bgp 100 metric 2000 subnets
 passive-interface Serial0
 network 203.250.0.0 0.0.255.255 area 0
 network 128.213.0.0 0.0.255.255 area 0

router bgp 100
 network 203.250.0.0 mask 255.255.0.0
 neighbor 128.213.63.2 remote-as 200
 neighbor 203.250.15.2 remote-as 100
 neighbor 203.250.15.2 update-source Loopback0

```

The routing table will look like this:

```

RTB#sh ip rou
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
 i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * -
candidate default
Gateway of last resort is not set

O E2 200.200.10.0 [110/2000] via 203.250.15.1, 00:00:14, Serial0
O E2 195.211.10.0 [110/2000] via 203.250.15.1, 00:00:14, Serial0
O E2 192.208.10.0 [110/2000] via 203.250.15.1, 00:00:14, Serial0
 203.250.13.0 is variably subnetted, 2 subnets, 2 masks
O 203.250.13.41 255.255.255.255
 [110/75] via 203.250.15.1, 00:00:15, Serial0
O E2 203.250.13.0 255.255.255.0
 [110/2000] via 203.250.15.1, 00:00:15, Serial0
 203.250.15.0 255.255.255.252 is subnetted, 2 subnets
C 203.250.15.8 is directly connected, Loopback1
C 203.250.15.0 is directly connected, Serial0
O 203.250.14.0 [110/74] via 203.250.15.1, 00:00:15, Serial0
 128.213.0.0 is variably subnetted, 2 subnets, 2 masks
O E2 128.213.0.0 255.255.0.0 [110/2000] via 203.250.15.1,
00:00:15,Serial0
O 128.213.63.0 255.255.255.252
 [110/138] via 203.250.15.1, 00:00:16, Serial0

```

**The BGP entries have disappeared because OSPF has a better distance (110) than internal bgp (200).**

I will also turn sync off on RTA in order for it to advertise 203.250.15.0, because it will not sync up with OSPF due to the difference in masks. I will also keep sync off on RTB in order for it to advertise 203.250.13.0 for the same reason.

Let us bring RTB's s1 up and see what all the routes will look like. I will also enable OSPF on serial 1 of RTB and make it passive in order for RTA to know about the nexthop 192.208.10.5 via IGP. Otherwise some looping will occur because in order to get to nexthop 192.208.10.5 we would have to go the other way via EBGP. The updated configs of RTA and RTB follow:

```
RTA#
hostname RTA

ip subnet-zero

interface Loopback0
 ip address 203.250.13.41 255.255.255.0

interface Ethernet0
 ip address 203.250.14.1 255.255.255.0

interface Serial0
 ip address 128.213.63.1 255.255.255.252

router ospf 10
 redistribute bgp 100 metric 2000 subnets
 passive-interface Serial0
 network 203.250.0.0 0.0.255.255 area 0
 network 128.213.0.0 0.0.255.255 area 0

router bgp 100
 no synchronization
 network 203.250.0.0 mask 255.255.0.0
 neighbor 128.213.63.2 remote-as 200
 neighbor 203.250.15.2 remote-as 100
 neighbor 203.250.15.2 update-source Loopback0
```

```

RTB#
hostname RTB

ip subnet-zero

interface Serial0
 ip address 203.250.15.2 255.255.255.252

interface Serial1
 ip address 192.208.10.6 255.255.255.252

router ospf 10
 redistribute bgp 100 metric 1000 subnets
 passive-interface Serial1
 network 203.250.0.0 0.0.255.255 area 0
 network 192.208.0.0 0.0.255.255 area 0

router bgp 100
 no synchronization
 network 203.250.15.0
 neighbor 192.208.10.5 remote-as 300
 neighbor 203.250.13.41 remote-as 100

```

And the BGP tables look like this:

```

RTA#sh ip bgp
BGP table version is 117, local router ID is 203.250.13.41
Status codes: s suppressed, d damped, h history, * valid, > best,
i -internal Origin codes: i - IGP, e - EGP, ? - incomplete

```

| Network         | Next Hop     | Metric | LocPrf | Weight | Path          |
|-----------------|--------------|--------|--------|--------|---------------|
| *> 128.213.0.0  | 128.213.63.2 | 0      |        | 0      | 200 i         |
| *>i192.208.10.0 | 192.208.10.5 | 0      | 100    | 0      | 300 i         |
| *>i195.211.10.0 | 192.208.10.5 | 100    |        | 0      | 300 500 i     |
| *               | 128.213.63.2 |        |        | 0      | 200 400 500 i |
| *> 200.200.10.0 | 128.213.63.2 |        |        | 0      | 200 400 i     |
| *> 203.250.13.0 | 0.0.0.0      | 0      |        | 32768  | i             |
| *> 203.250.14.0 | 0.0.0.0      | 0      |        | 32768  | i             |
| *>i203.250.15.0 | 203.250.15.2 | 0      | 100    | 0      | i             |

```

RTB#sh ip bgp
BGP table version is 12, local router ID is 203.250.15.10
Status codes: s suppressed, d damped, h history, * valid, > best,
i -internal Origin codes: i - IGP, e - EGP, ? - incomplete

 Network Next Hop Metric LocPrf Weight Path
*->i128.213.0.0 128.213.63.2 0 100 0 200 i
* 192.208.10.5 0 300 500 400
200 i
*> 192.208.10.0 192.208.10.5 0 300 i
*> 195.211.10.0 192.208.10.5 0 300 500 i
*>i200.200.10.0 128.213.63.2 100 0 200 400 i
* 192.208.10.5 0 300 500 400 i
*>i203.250.13.0 203.250.13.41 0 100 0 i
*>i203.250.14.0 203.250.13.41 0 100 0 i
*> 203.250.15.0 0.0.0.0 0 32768 i

```

There are multiple ways to design our network to talk to the two different ISPs AS200 and AS300. One way is to have a primary ISP and a backup ISP. We could learn partial routes from one of the ISPs and default routes to both ISPs. In this example, I have chosen to receive partial routes from AS200 and only local routes from AS300.

Both RTA and RTB are generating default routes into OSPF with RTB being more preferred (lower metric). This way I could balance outgoing traffic between the two ISPs.

Potential asymmetry might occur if traffic going out from RTA comes back via RTB. This might occur if you are using the same pool of IP addresses (same major net) when talking to the two ISPs. Because of aggregation your whole AS might look as one whole entity to the outside world and entry points to your network could occur via RTA or RTB. You might find out that all incoming traffic to your AS is coming via one single point even though you have multiple points to the internet. In our example, I have chosen two different major nets when talking to the two ISPs.

One other potential reason for asymmetry is the different advertised path length to reach your AS. One service provider might be closer to a certain destination than another. In our example, traffic from AS400 destined to your network will always come in via RTA because of the shorter path. You might try to affect that decision by prepending path numbers to your updates to make the path length look longer (set as-path prepend). But, if AS400 has somehow set its exit point to be via AS200 based on attributes such as local preference or metric or weight then there is nothing you can do.

This is the final configuration for all of the routers:

```

RTA#
hostname RTA

ip subnet-zero

interface Loopback0
 ip address 203.250.13.41 255.255.255.0

interface Ethernet0
 ip address 203.250.14.1 255.255.255.0

interface Serial0
 ip address 128.213.63.1 255.255.255.252

router ospf 10
 redistribute bgp 100 metric 2000 subnets
 passive-interface Serial0
 network 203.250.0.0 0.0.255.255 area 0
 network 128.213.0.0 0.0.255.255 area 0
 default-information originate metric 2000

router bgp 100
 no synchronization
 network 203.250.13.0
 network 203.250.14.0
 neighbor 128.213.63.2 remote-as 200
 neighbor 128.213.63.2 route-map setlocalpref in
 neighbor 203.250.15.2 remote-as 100
 neighbor 203.250.15.2 update-source Loopback0

ip classless
ip default-network 200.200.0.0

route-map setlocalpref permit 10
 set local-preference 200

```

On RTA, the local preference for routes coming from AS200 is set to 200. I have also picked network 200.200.0.0 to be the candidate default, using the "ip default-network" command.

The "default-information originate" command is used with OSPF to inject the default route inside the OSPF domain. This command is also used with ISIS and BGP. For RIP, 0.0.0.0 is automatically redistributed into RIP without additional configuration. For IGRP and EIGRP, the default information is injected into the IGP domain after redistributing BGP into IGRP/EIGRP. Also with IGRP/EIGRP we can redistribute a static route to 0.0.0.0 into the IGP domain.

```

RTF#
hostname RTF

ip subnet-zero

interface Ethernet0
 ip address 203.250.14.2 255.255.255.0

interface Serial1
 ip address 203.250.15.1 255.255.255.252

router ospf 10
 network 203.250.0.0 0.0.255.255 area 0

ip classless

RTB#
hostname RTB

ip subnet-zero

interface Loopback1
 ip address 203.250.15.10 255.255.255.252

interface Serial0
 ip address 203.250.15.2 255.255.255.252
!
interface Serial1
 ip address 192.208.10.6 255.255.255.252

router ospf 10
 redistribute bgp 100 metric 1000 subnets
 passive-interface Serial1
 network 203.250.0.0 0.0.255.255 area 0
 network 192.208.10.6 0.0.0.0 area 0
 default-information originate metric 1000
!
router bgp 100
 no synchronization
 network 203.250.15.0
 neighbor 192.208.10.5 remote-as 300
 neighbor 192.208.10.5 route-map localonly in
 neighbor 203.250.13.41 remote-as 100
!
ip classless
ip default-network 192.208.10.0
ip as-path access-list 1 permit ^300$

route-map localonly permit 10
 match as-path 1
 set local-preference 300

```

For RTB, the local preference for updates coming in from AS300 is set to 300 which is higher than the IBGP updates coming in from RTA. This way AS100 will pick RTB for AS300's local routes. Any other routes on RTB (if they exist) will be sent internally with a local preference of 100 which is lower than 200 coming in from RTA, and this way RTA will be preferred.

Note that I have only advertised AS300's local routes. Any path info that does not match ^300\$ will be dropped. If you wanted to advertise the local routes and the neighbor routes (customers of the ISP) you can use the following: ^300\_[0-9]\*

This is the output of the regular expression indicating AS300's local routes:

```
RTB#sh ip bgp regexp ^300$
BGP table version is 14, local router ID is 203.250.15.10
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

| Network         | Next Hop     | Metric | LocPrf | Weight | Path |
|-----------------|--------------|--------|--------|--------|------|
| *> 192.208.10.0 | 192.208.10.5 | 0      | 300    | 0      | 300  |

```
RTC#
hostname RTC

ip subnet-zero

interface Loopback0
 ip address 128.213.63.130 255.255.255.192

interface Serial2/0
 ip address 128.213.63.5 255.255.255.252
!
interface Serial2/1
 ip address 128.213.63.2 255.255.255.252

router bgp 200
 network 128.213.0.0
 aggregate-address 128.213.0.0 255.255.0.0 summary-only
 neighbor 128.213.63.1 remote-as 100
 neighbor 128.213.63.1 distribute-list 1 out
 neighbor 128.213.63.6 remote-as 400

ip classless
access-list 1 deny 195.211.0.0 0.0.255.255
access-list 1 permit any
```

On RTC, I have aggregated 128.213.0.0/16 and indicated the specific routes to be injected into AS100. If the ISP refuses to do this task then you have to filter on the incoming end of AS100.

```

RTD#
hostname RTD

ip subnet-zero

interface Loopback0
 ip address 192.208.10.174 255.255.255.192
!
interface Serial0/0
 ip address 192.208.10.5 255.255.255.252
!
interface Serial0/1
 ip address 192.208.10.2 255.255.255.252

router bgp 300
 network 192.208.10.0
 neighbor 192.208.10.1 remote-as 500
 neighbor 192.208.10.6 remote-as 100

```

```

RTG#
hostname RTG

ip subnet-zero

interface Loopback0
 ip address 195.211.10.174 255.255.255.192

interface Serial0
 ip address 192.208.10.1 255.255.255.252

interface Serial1
 ip address 195.211.10.1 255.255.255.252

router bgp 500
 network 195.211.10.0
 aggregate-address 195.211.0.0 255.255.0.0 summary-only
 neighbor 192.208.10.2 remote-as 300
 neighbor 192.208.10.2 send-community
 neighbor 192.208.10.2 route-map setcommunity out
 neighbor 195.211.10.2 remote-as 400
!
ip classless
access-list 1 permit 195.211.0.0 0.0.255.255
access-list 2 permit any
access-list 101 permit ip 195.211.0.0 0.0.255.255 host 255.255.0.0
route-map setcommunity permit 20
 match ip address 2
!
route-map setcommunity permit 10
 match ip address 1
 set community no-export

```

On RTG, I have demonstrated the use of community filtering by adding a no-export community to 195.211.0.0 updates towards RTD. This way RTD will not export that route to RTB. It doesn't matter in our case because RTB is not accepting these routes anyway.

```
RTE#
hostname RTE

ip subnet-zero

interface Loopback0
 ip address 200.200.10.1 255.255.255.0

interface Serial0
 ip address 195.211.10.2 255.255.255.252

interface Serial1
 ip address 128.213.63.6 255.255.255.252

router bgp 400
 network 200.200.10.0
 aggregate-address 200.200.0.0 255.255.0.0 summary-only
 neighbor 128.213.63.5 remote-as 200
 neighbor 195.211.10.1 remote-as 500

ip classless

RTE is aggregating 200.200.0.0/16.
```

And following are the final bgp and routing tables for RTA, RTF and RTB:

```
RTA#sh ip bgp
BGP table version is 21, local router ID is 203.250.13.41
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

| Network           | Next Hop     | Metric | LocPrf | Weight | Path      |
|-------------------|--------------|--------|--------|--------|-----------|
| *> 128.213.0.0    | 128.213.63.2 | 0      | 200    | 0      | 200 i     |
| *>i192.208.10.0   | 192.208.10.5 | 0      | 300    | 0      | 300 i     |
| *> 200.200.0.0/16 | 128.213.63.2 |        | 200    | 0      | 200 400 i |
| *> 203.250.13.0   | 0.0.0.0      | 0      |        | 32768  | i         |
| *> 203.250.14.0   | 0.0.0.0      | 0      |        | 32768  | i         |
| *>i203.250.15.0   | 203.250.15.2 | 0      | 100    | 0      | i         |

```
RTA#sh ip rou
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
 i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * -
candidate default

Gateway of last resort is 128.213.63.2 to network 200.200.0.0

 192.208.10.0 is variably subnetted, 2 subnets, 2 masks
O E2 192.208.10.0 255.255.255.0
 [110/1000] via 203.250.14.2, 00:41:25, Ethernet0
O 192.208.10.4 255.255.255.252
 [110/138] via 203.250.14.2, 00:41:25, Ethernet0
C 203.250.13.0 is directly connected, Loopback0
203.250.15.0 is variably subnetted, 3 subnets, 3 masks
O 203.250.15.10 255.255.255.255
 [110/75] via 203.250.14.2, 00:41:25, Ethernet0
O 203.250.15.0 255.255.255.252
 [110/74] via 203.250.14.2, 00:41:25, Ethernet0
B 203.250.15.0 255.255.255.0 [200/0] via 203.250.15.2, 00:41:25
C 203.250.14.0 is directly connected, Ethernet0
128.213.0.0 is variably subnetted, 2 subnets, 2 masks
B 128.213.0.0 255.255.0.0 [20/0] via 128.213.63.2, 00:41:26
C 128.213.63.0 255.255.255.252 is directly connected, Serial0
B* 200.200.0.0 255.255.0.0 [20/0] via 128.213.63.2, 00:02:38
```

```

RTF#sh ip rou
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
 i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is 203.250.15.2 to network 0.0.0.0

 192.208.10.0 is variably subnetted, 2 subnets, 2 masks
O E2 192.208.10.0 255.255.255.0
 [110/1000] via 203.250.15.2, 00:48:50, Serial1
O 192.208.10.4 255.255.255.252
 [110/128] via 203.250.15.2, 01:12:09, Serial1
 203.250.13.0 is variably subnetted, 2 subnets, 2 masks
O 203.250.13.41 255.255.255.255
 [110/11] via 203.250.14.1, 01:12:09, Ethernet0
O E2 203.250.13.0 255.255.255.0
 [110/2000] via 203.250.14.1, 01:12:09, Ethernet0
 203.250.15.0 is variably subnetted, 2 subnets, 2 masks
O 203.250.15.10 255.255.255.255
 [110/65] via 203.250.15.2, 01:12:09, Serial1
C 203.250.15.0 255.255.255.252 is directly connected, Serial1
C 203.250.14.0 is directly connected, Ethernet0
 128.213.0.0 is variably subnetted, 2 subnets, 2 masks
O E2 128.213.0.0 255.255.0.0
 [110/2000] via 203.250.14.1, 00:45:01, Ethernet0
O 128.213.63.0 255.255.255.252
 [110/74] via 203.250.14.1, 01:12:11, Ethernet0
O E2 200.200.0.0 255.255.0.0 [110/2000] via 203.250.14.1, 00:03:47,
Ethernet0
O*E2 0.0.0.0 0.0.0.0 [110/1000] via 203.250.15.2, 00:03:33, Serial1

```

Note RTF's routing table which indicates that networks local to AS300 such as 192.208.10.0 are to be reached via RTB. Other known networks such as 200.200.0.0 are to be reached via RTA. The gateway of last resort is set to RTB. In case something happens to the connection between RTB and RTD, then the default advertised by RTA will kick in with a metric of 2000.

```

RTB#sh ip bgp
BGP table version is 14, local router ID is 203.250.15.10
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete

```

| Network           | Next Hop      | Metric | LocPrf | Weight | Path      |
|-------------------|---------------|--------|--------|--------|-----------|
| *>i128.213.0.0    | 128.213.63.2  | 0      | 200    | 0      | 200 i     |
| *> 192.208.10.0   | 192.208.10.5  | 0      | 300    | 0      | 300 i     |
| *>i200.200.0.0/16 | 128.213.63.2  |        | 200    | 0      | 200 400 i |
| *>i203.250.13.0   | 203.250.13.41 | 0      | 100    | 0      | i         |
| *>i203.250.14.0   | 203.250.13.41 | 0      | 100    | 0      | i         |
| *> 203.250.15.0   | 0.0.0.0       | 0      |        | 32768  | i         |

```

RTB#sh ip rou
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
 i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * -
candidate default

```

Gateway of last resort is 192.208.10.5 to network 192.208.10.0

```

* 192.208.10.0 is variably subnetted, 2 subnets, 2 masks
B* 192.208.10.0 255.255.255.0 [20/0] via 192.208.10.5, 00:50:46
C 192.208.10.4 255.255.255.252 is directly connected, Serial1
 203.250.13.0 is variably subnetted, 2 subnets, 2 masks
O 203.250.13.41 255.255.255.255
 [110/75] via 203.250.15.1, 01:20:33, Serial0
O E2 203.250.13.0 255.255.255.0
 [110/2000] via 203.250.15.1, 01:15:40, Serial0
 203.250.15.0 255.255.255.252 is subnetted, 2 subnets
C 203.250.15.8 is directly connected, Loopback1
C 203.250.15.0 is directly connected, Serial0
O 203.250.14.0 [110/74] via 203.250.15.1, 01:20:33, Serial0
 128.213.0.0 is variably subnetted, 2 subnets, 2 masks
O E2 128.213.0.0 255.255.0.0 [110/2000] via 203.250.15.1, 00:46:55,
Serial0
O 128.213.63.0 255.255.255.252
 [110/138] via 203.250.15.1, 01:20:34, Serial0
O E2 200.200.0.0 255.255.0.0 [110/2000] via 203.250.15.1, 00:05:42,
Serial0

```