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

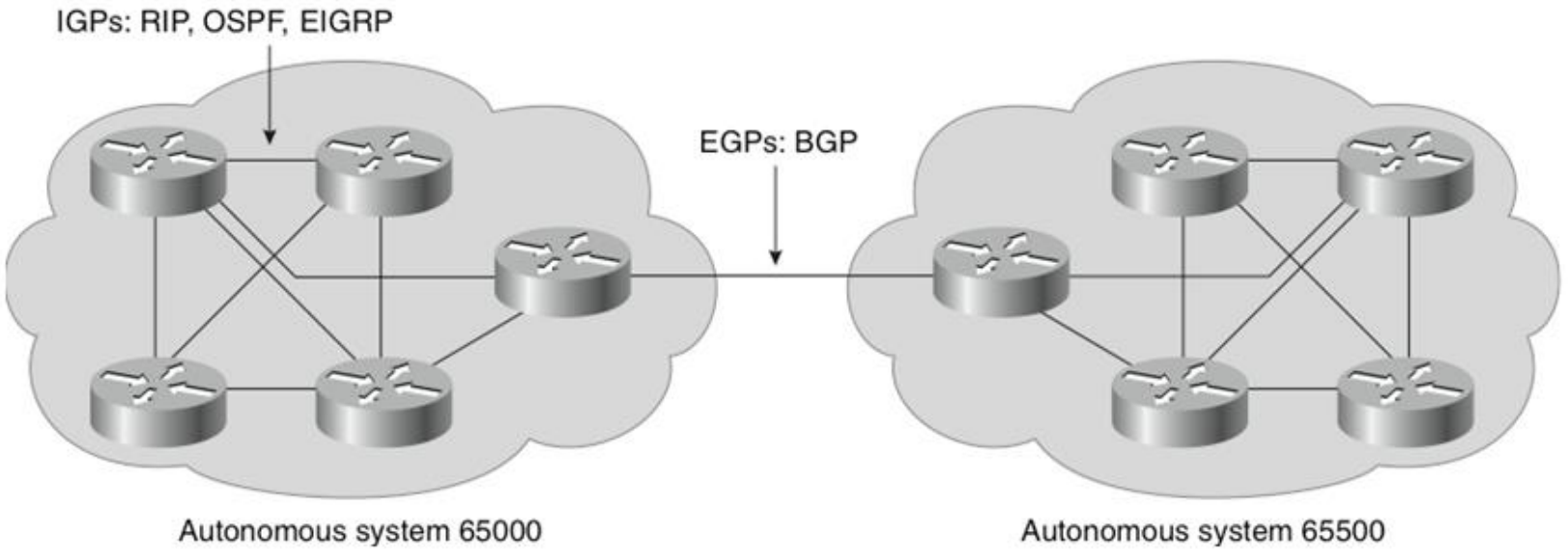
- **IGP** (Interior Gateway Protocol) - RIP, IGRP, EIGRP, OSPF = Routing protocol used to exchange routing information within an autonomous system.
- **EGP** (Exterior Gateway Protocol) - BGP = Routing protocol used to exchange routing information between autonomous systems.
- **Autonomous System** = (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.

Withdrawn



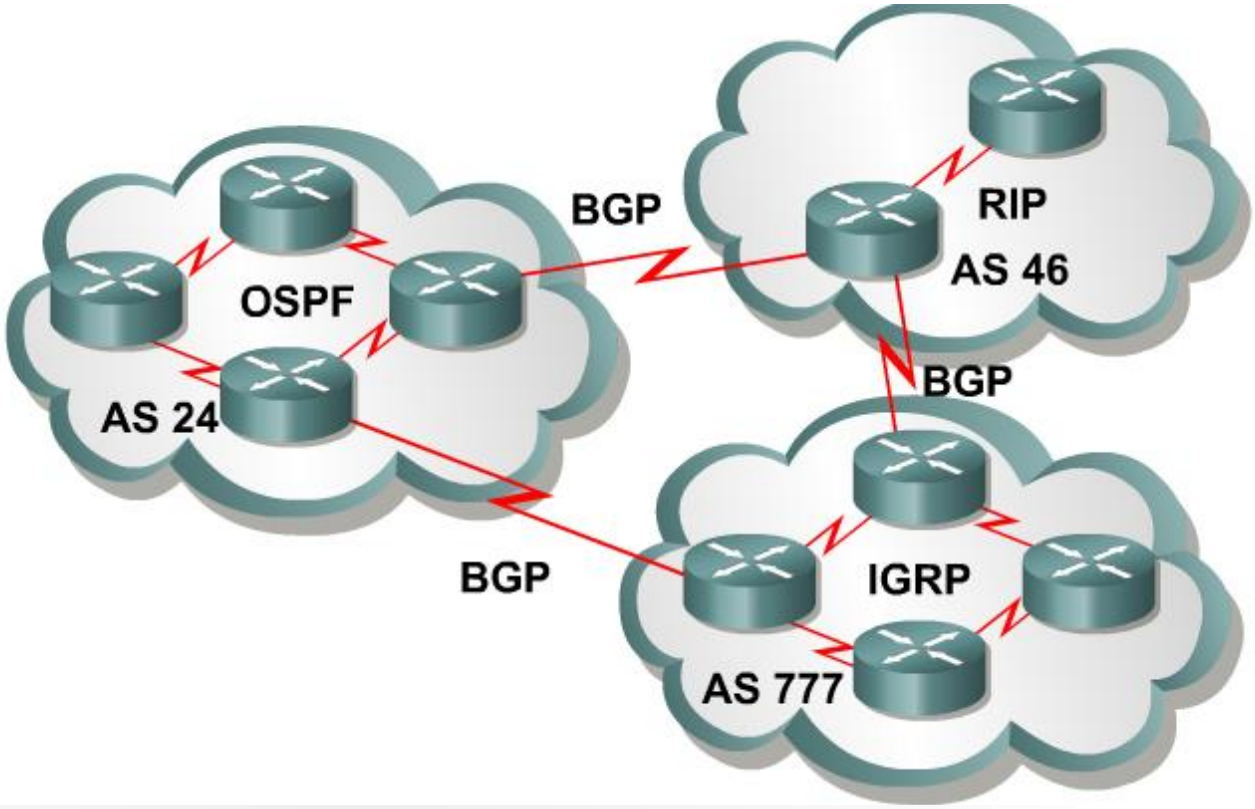
IGP versus EGP

- 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).

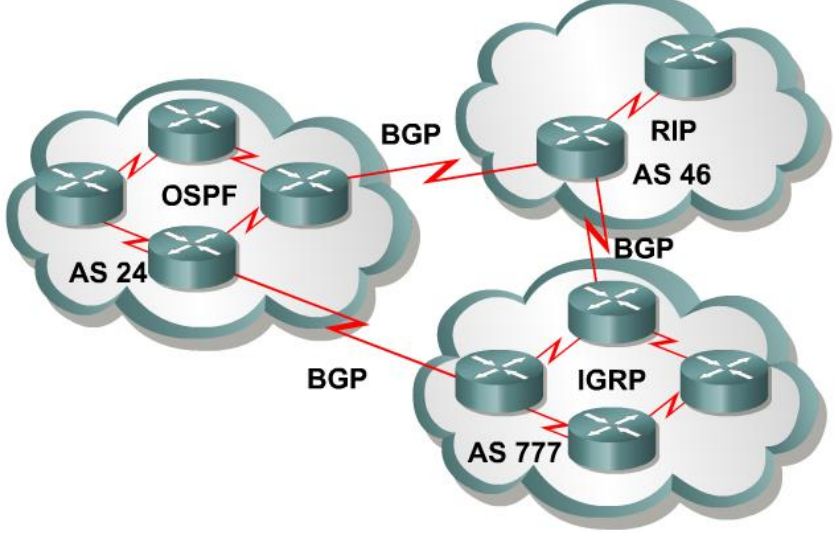
IANA

- 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)

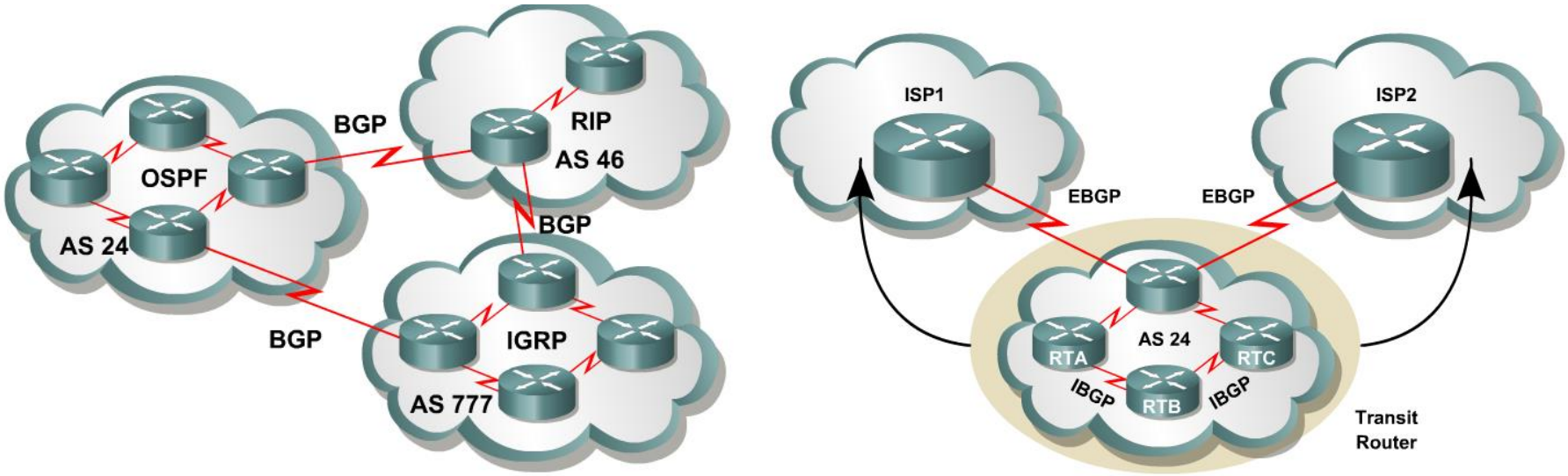
RIR Name	Geographic Coverage	Link
AfriNIC	Continent of Africa	www.afrinic.net
APNIC (Asia Pacific Network Information Centre)	Asia Pacific region	www.apnic.org
ARIN (American Registry for Internet Numbers)	Canada, the United States, and several islands in the Caribbean Sea and North Atlantic Ocean	www.arin.net
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.

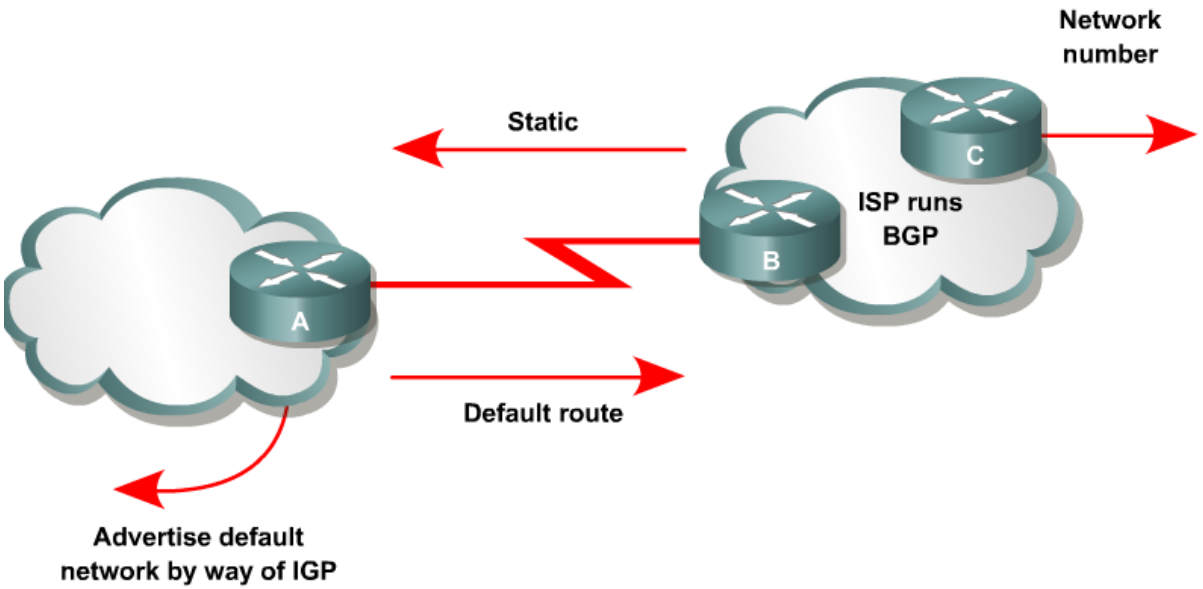
When to use BGP and when not to use BGP



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.

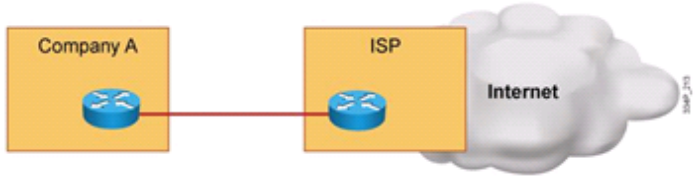
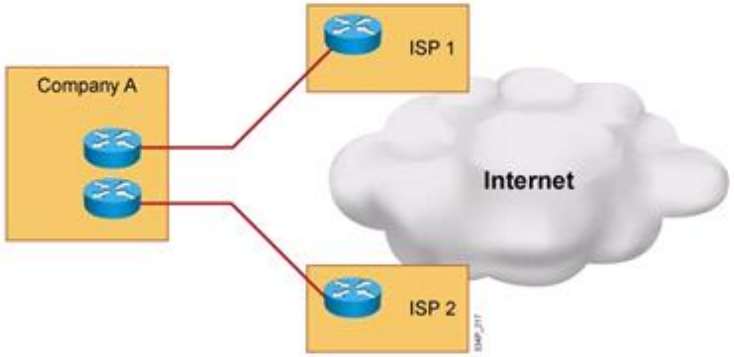
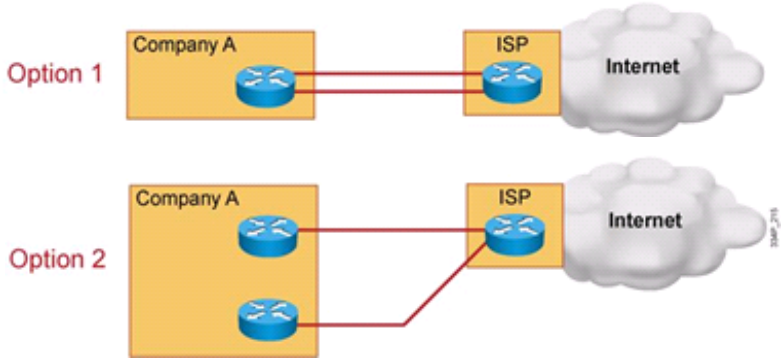
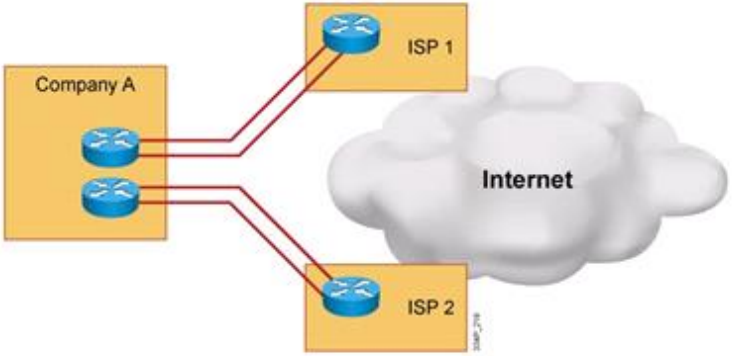
When to use BGP and when not to use BGP



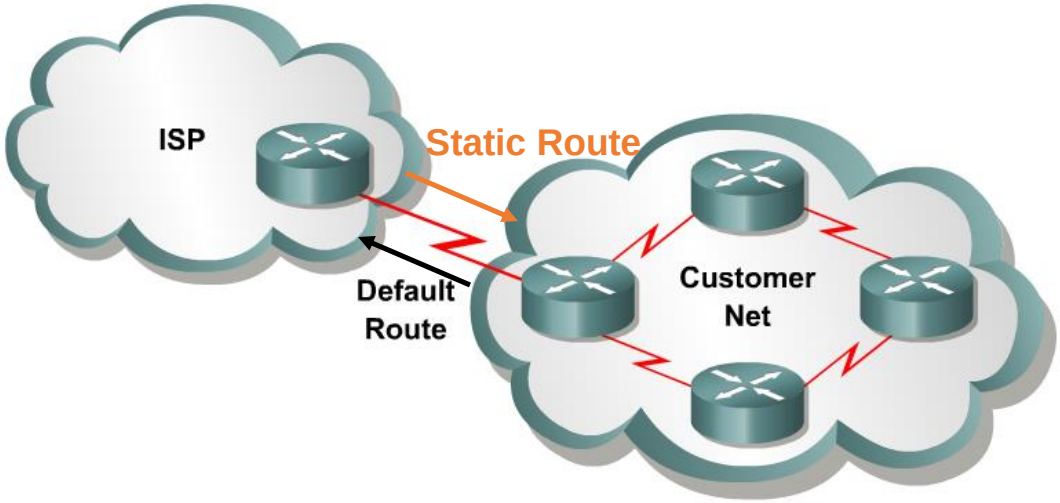
Do not use BGP if you have one or more of the following conditions:

- 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
Single-homed 	**Multihomed** 
Dual-homed 	**Dual-multihomed** 

Single-homed autonomous systems



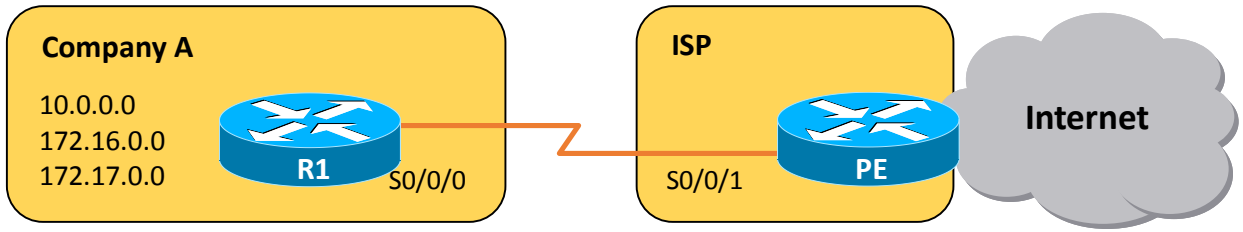
A single-homed AS can be configured with a default route to reach outside networks.

- 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

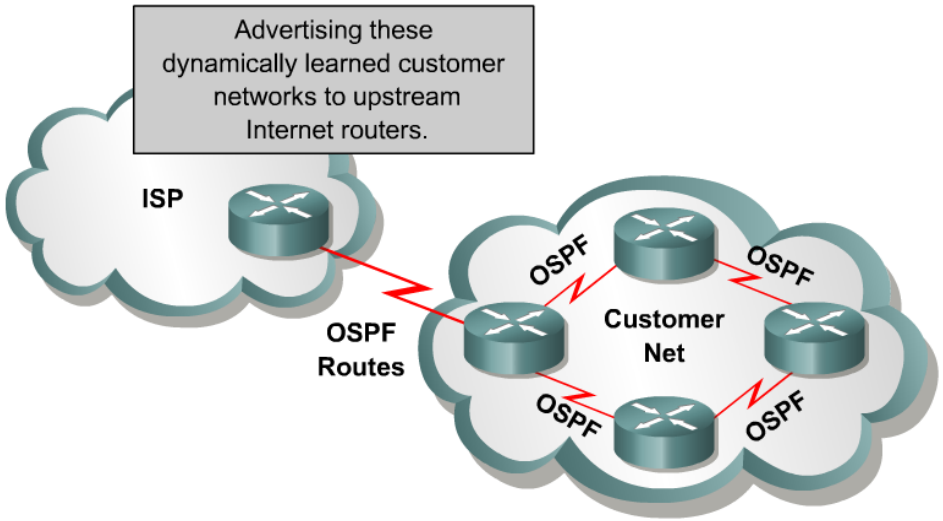
- Static routes are the simplest way to implement routing with an ISP.
  - Typically a customer has a single connection to an ISP and the customer uses a default route toward the ISP while the ISP deploys static routes toward the customer.

```
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
```

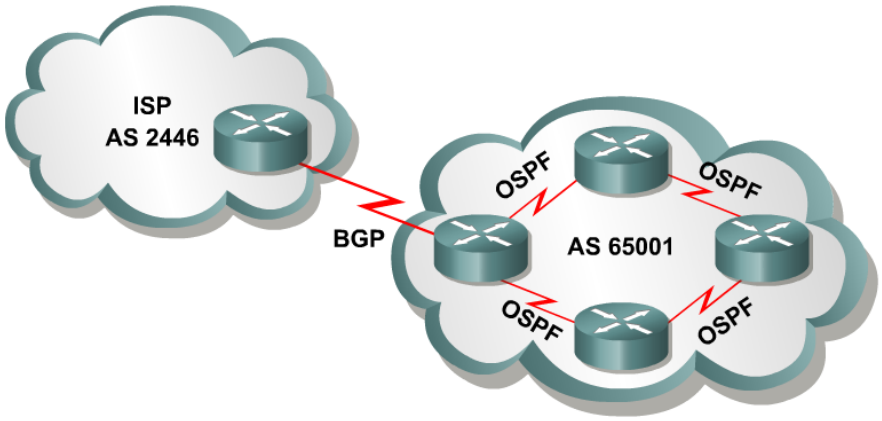
Single-homed autonomous systems



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.

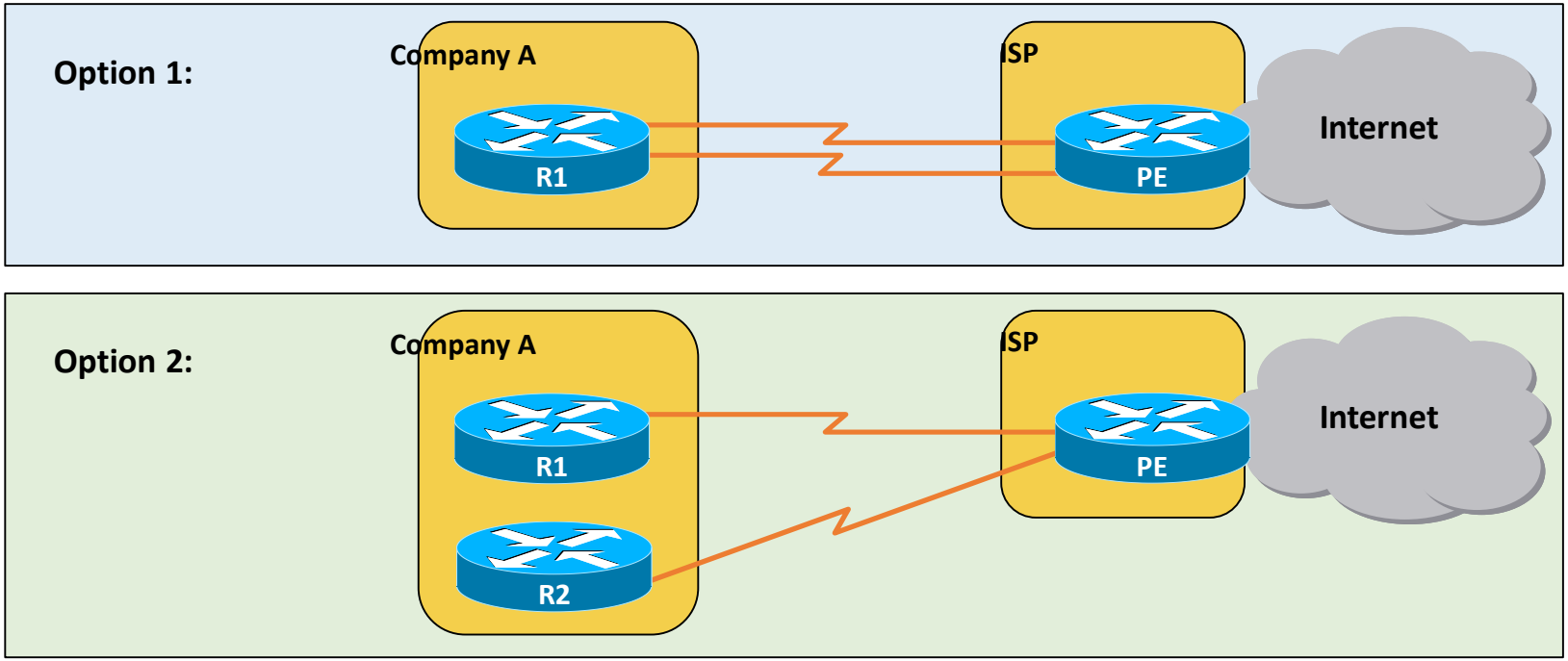
Single-homed autonomous systems

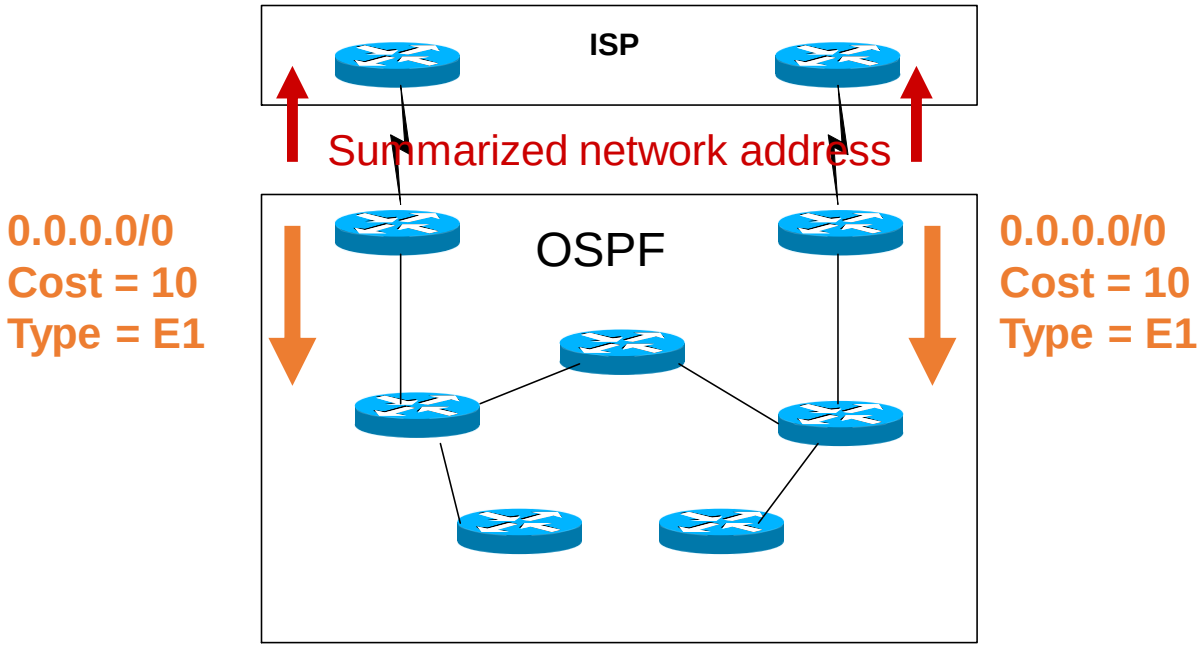


A provider may also choose to dynamically learn a customer's routes using BGP, which typically runs between the ISP router and the customer's boundary router.

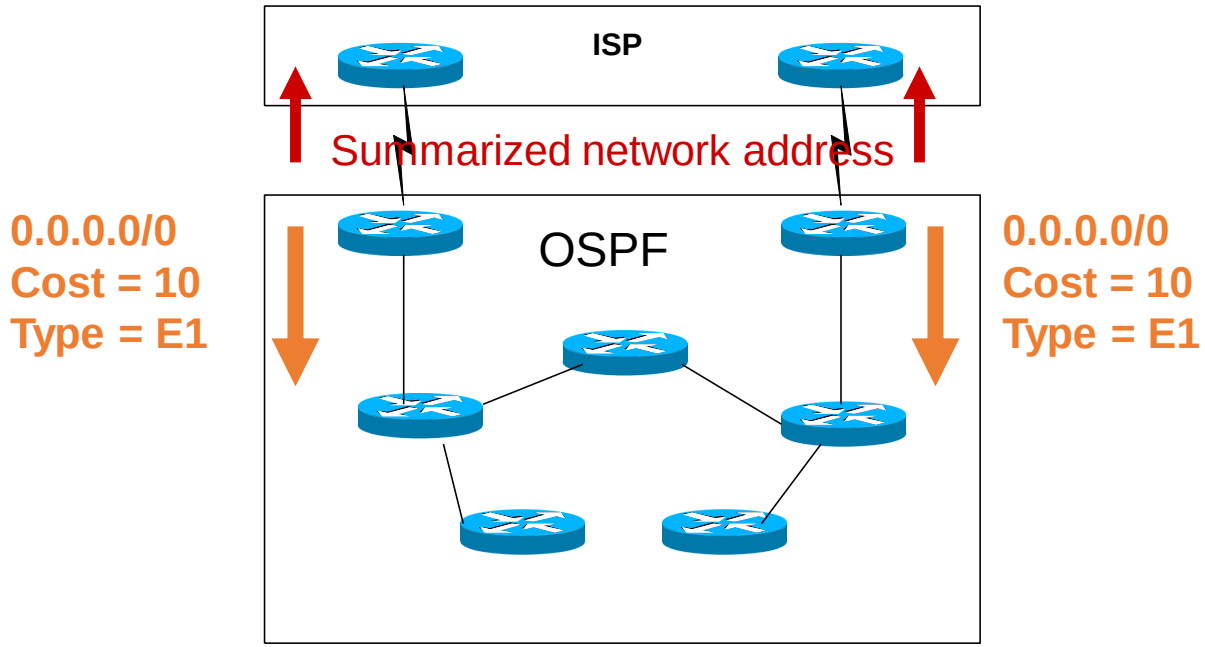
- **Use an EGP** – The third method by which the ISP can learn and advertise the customer's routes is to use an EGP such as BGP.
- In a **single-homed autonomous system** the *customer's routing policies are an extension of the policies of the provider*.
  - For this reason the Internet number registries 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:
Company A
ISP
R1
PE
Internet
Option 2:
Company A
ISP
R1
R2
PE
Internet

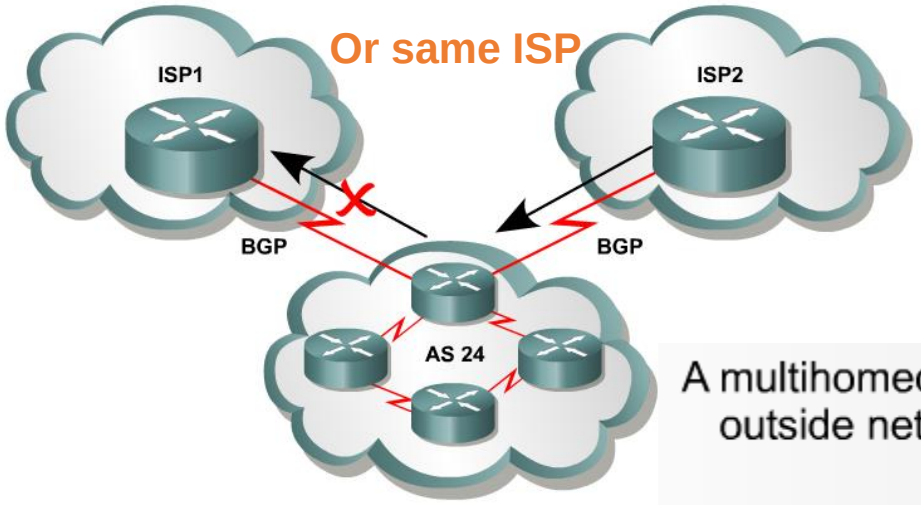

ISP
Summarized network address
0.0.0.0/0
Cost = 10
Type = E1
OSPF
0.0.0.0/0
Cost = 10
Type = E1

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.

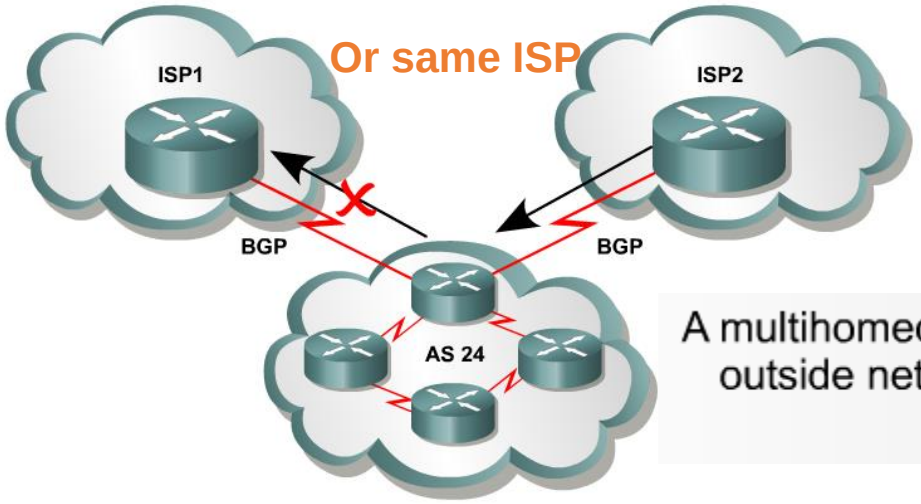
Multihomed **nontransit** autonomous systems



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.

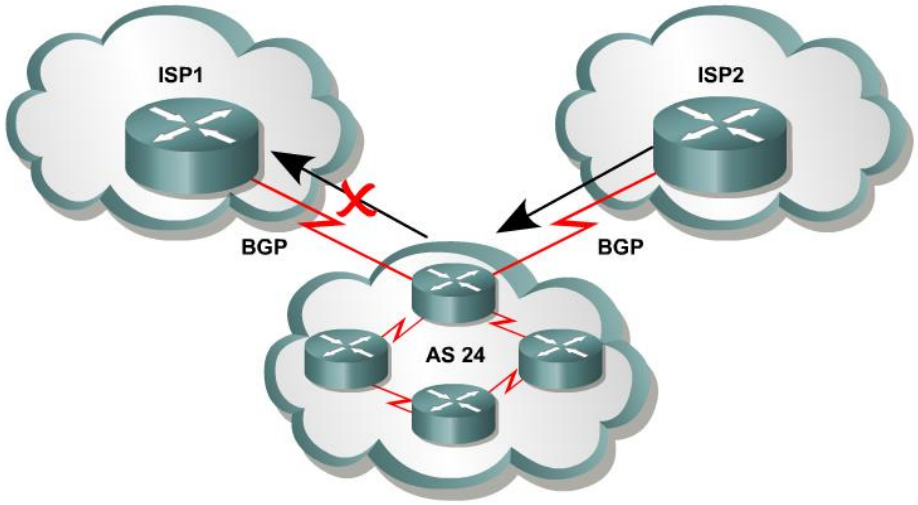
Multihomed **nontransit** autonomous systems



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.

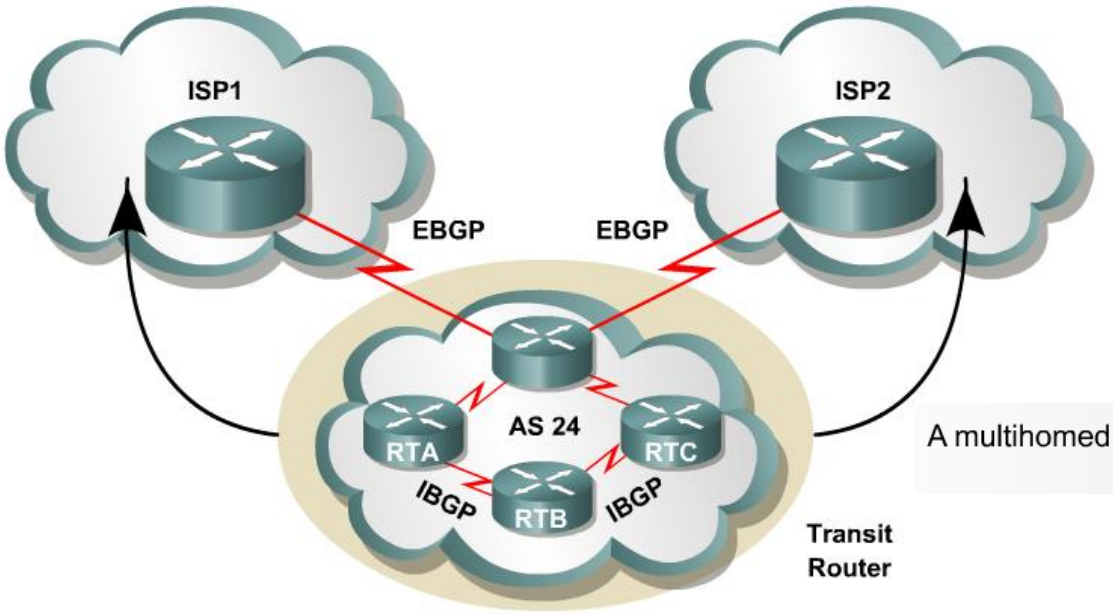
- Multihomed nontransit autonomous systems do not really need to run BGP4 with their providers.
- It is usually **recommended** and **often required** by ISPs.
- As it will be seen later in this module, BGP4 offers numerous advantages, including increased control of route propagation and filtering.

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.

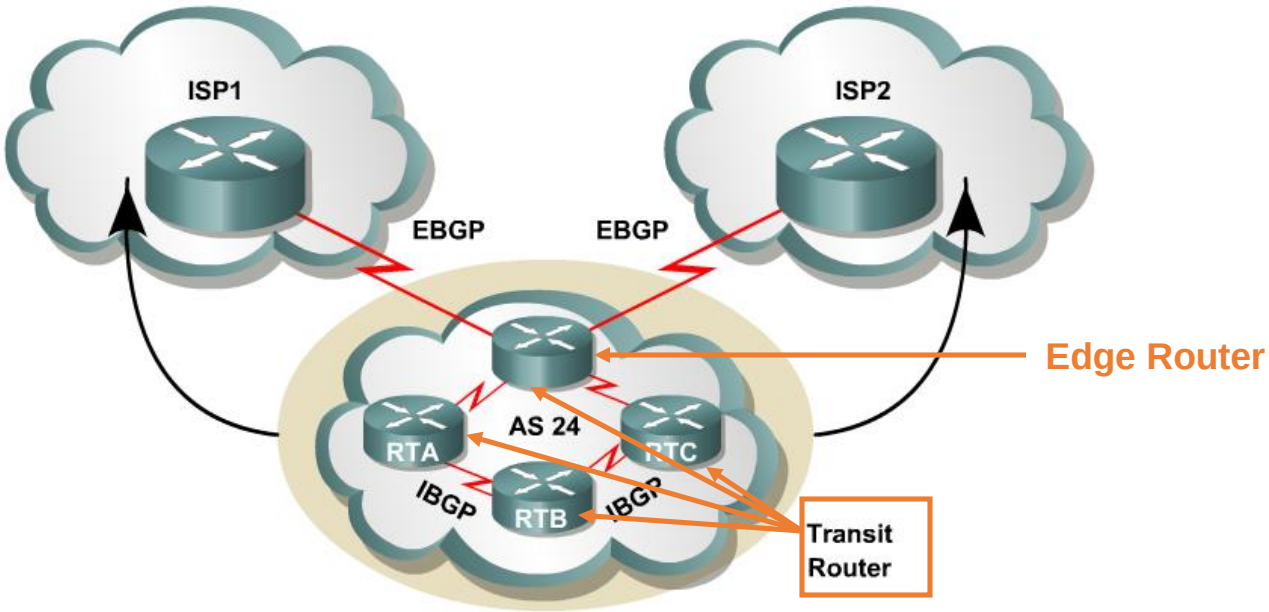
Multi-homed **Transit** Autonomous Systems



A multihomed transit system can be used for transit traffic by other 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

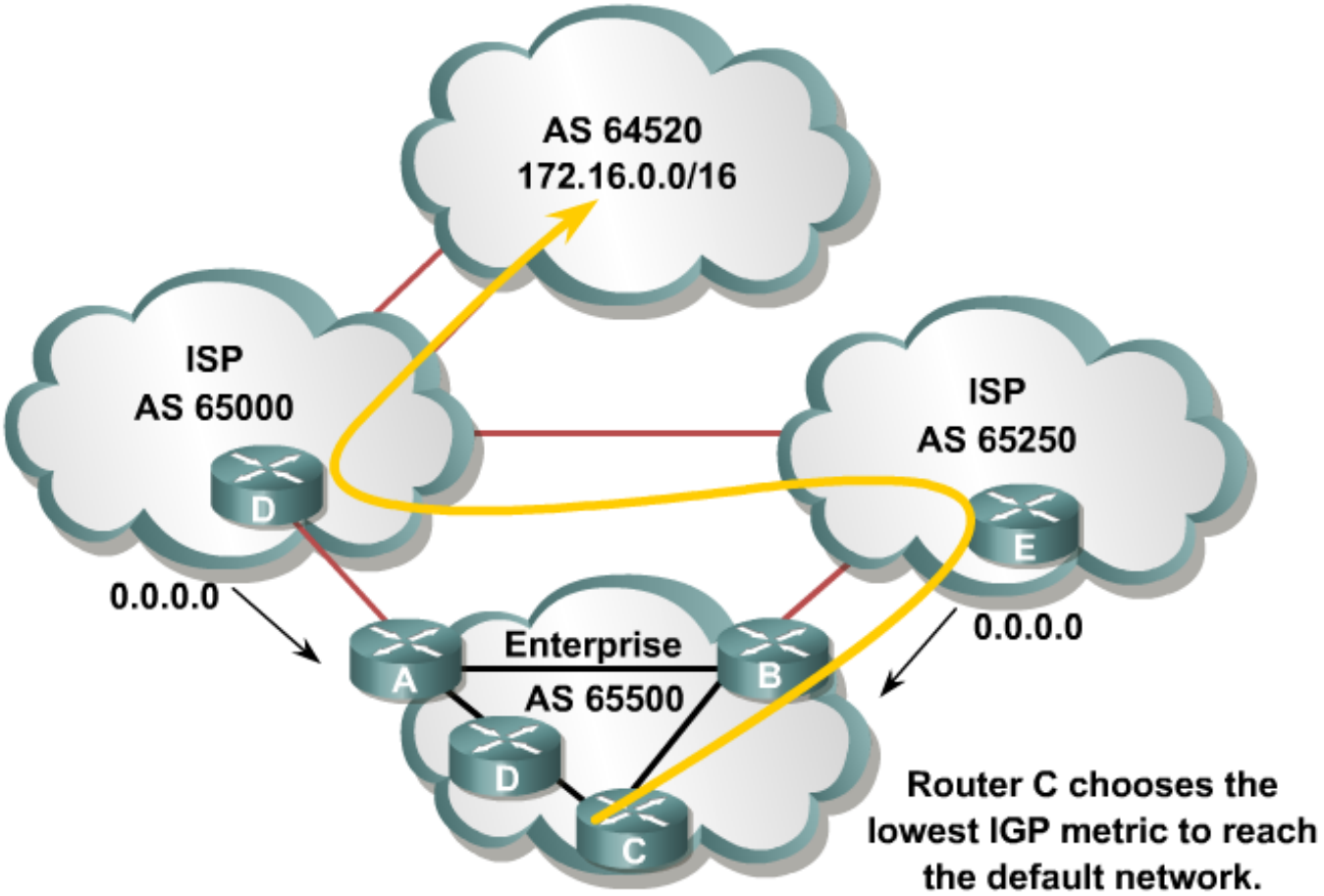


- When BGP is running inside an AS, it is referred to as **Internal BGP (IBGP).**
- When BGP runs between autonomous systems, it is called **External BGP (EBGP).**
- If the role of a BGP router is to route IBGP traffic, it is called a **transit router.**
- Routers that sit on the boundary of an AS and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

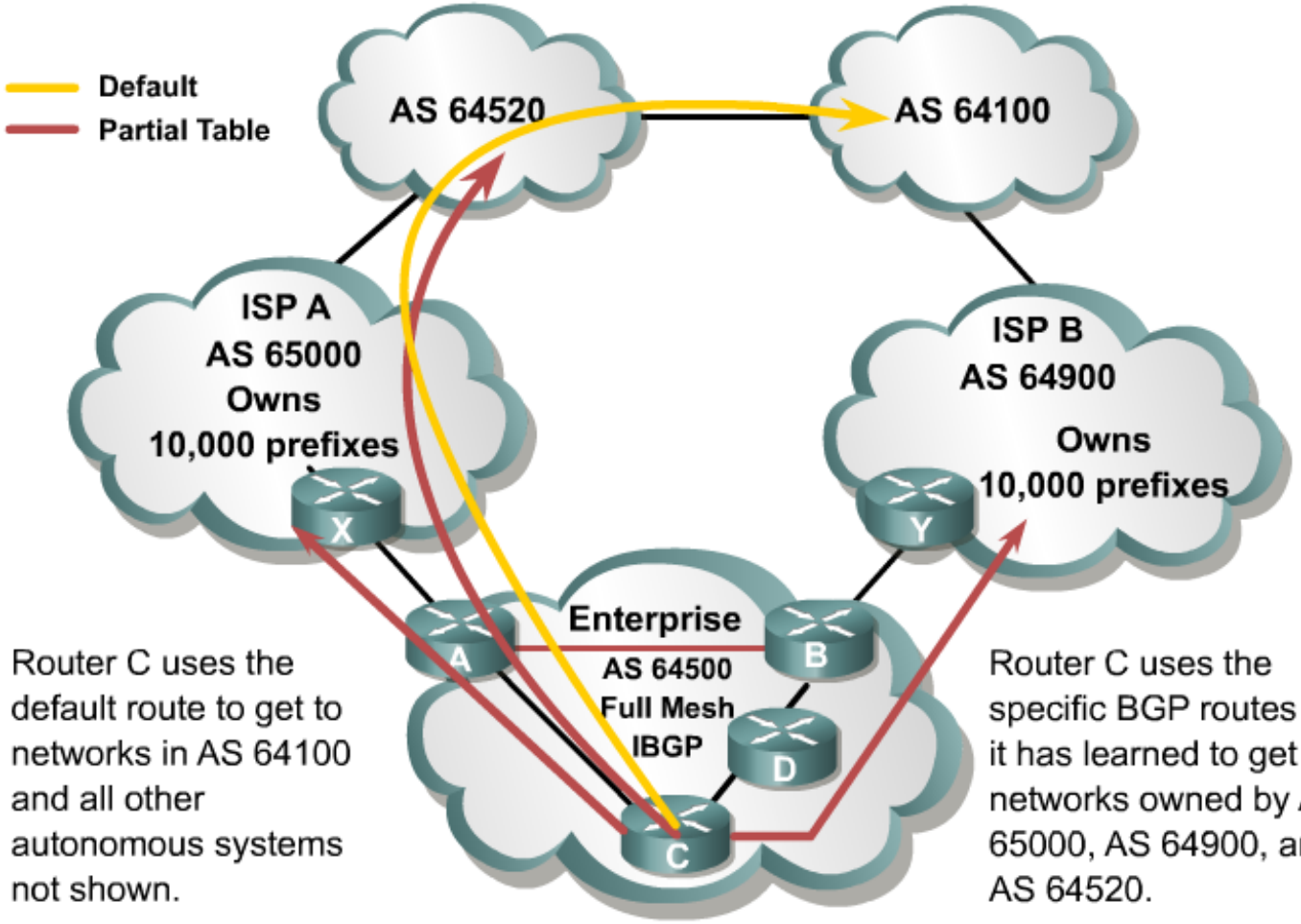
Three Multihoming Connection Options

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.

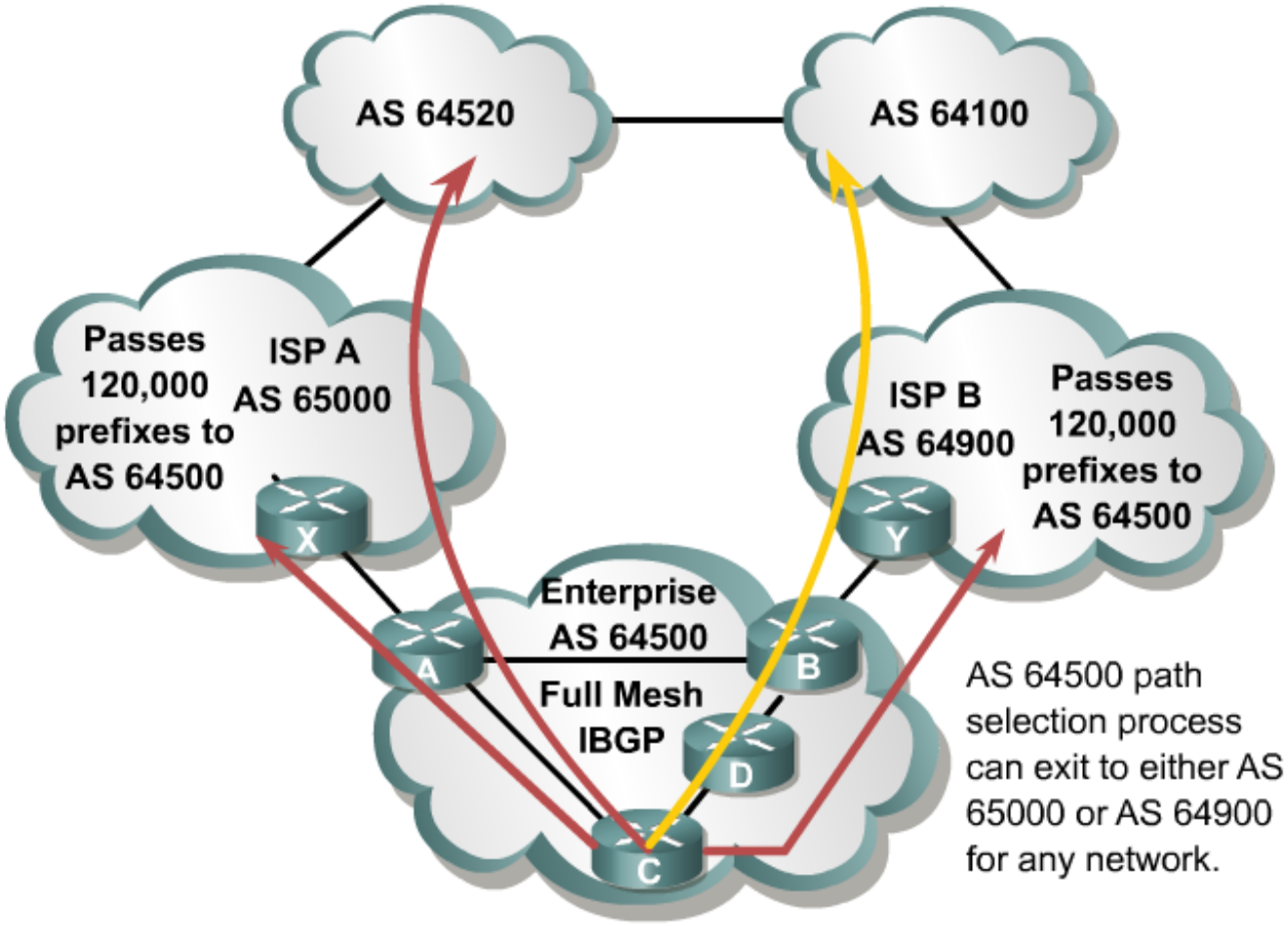
Default Routes from All Providers



Default Routes and Partial Updates



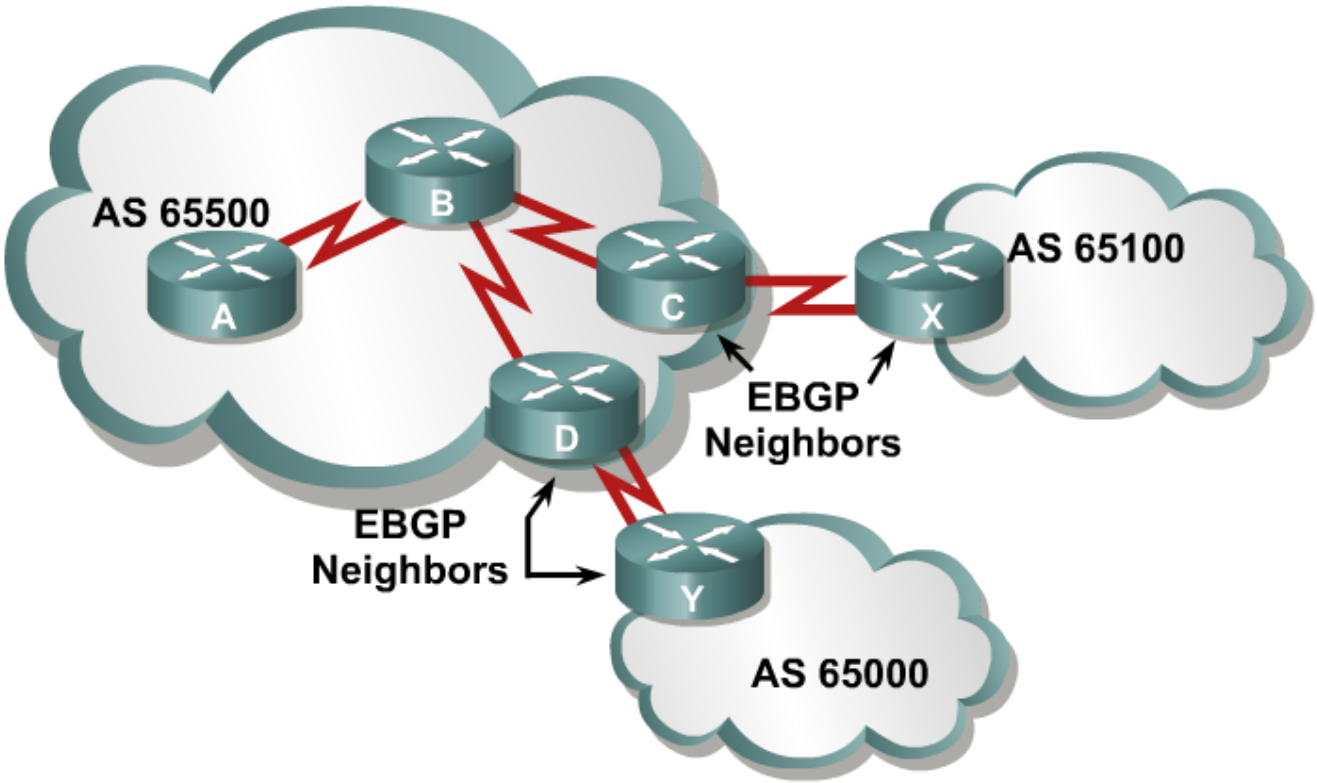
Full Routes from All Providers



EBGP vs IBGP

External BGP

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

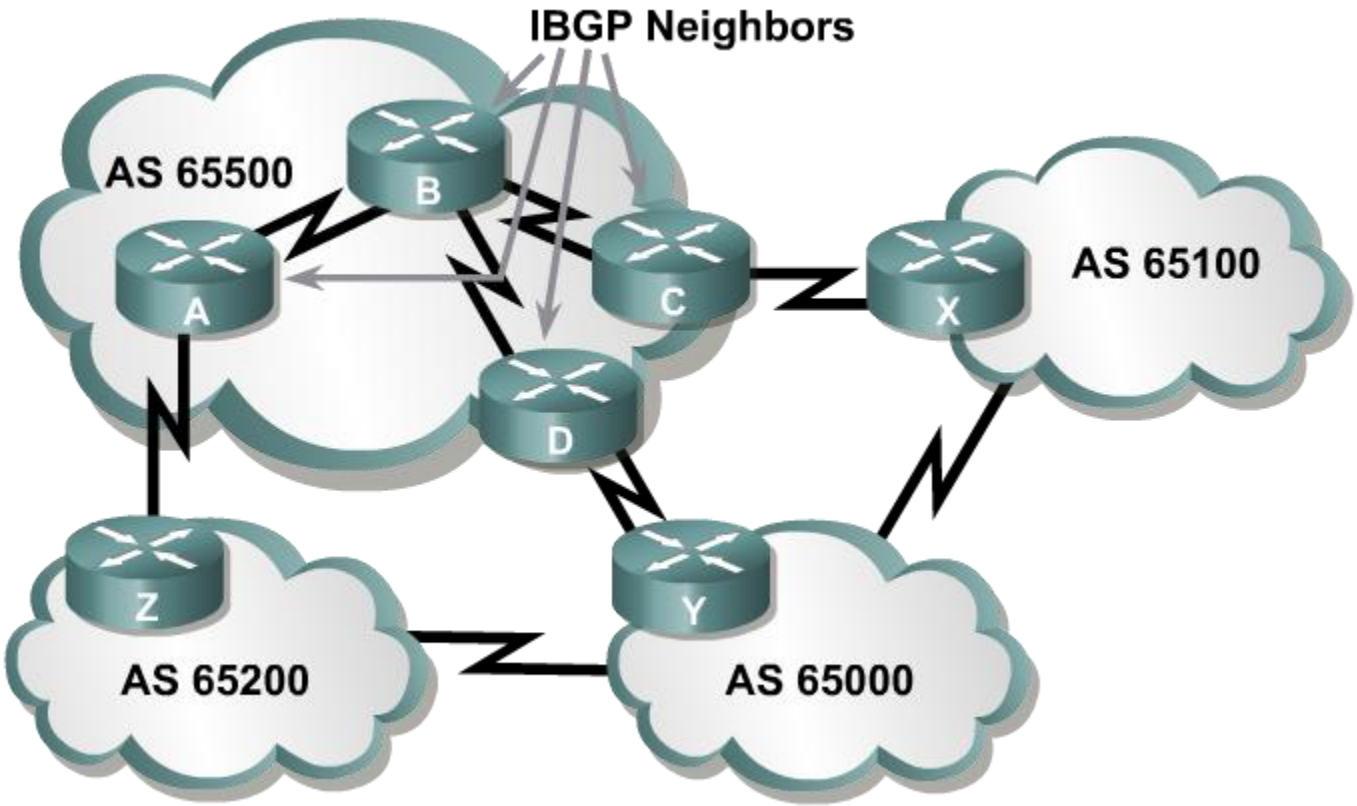


EBGP Neighbors Relationship Requirements

- 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.

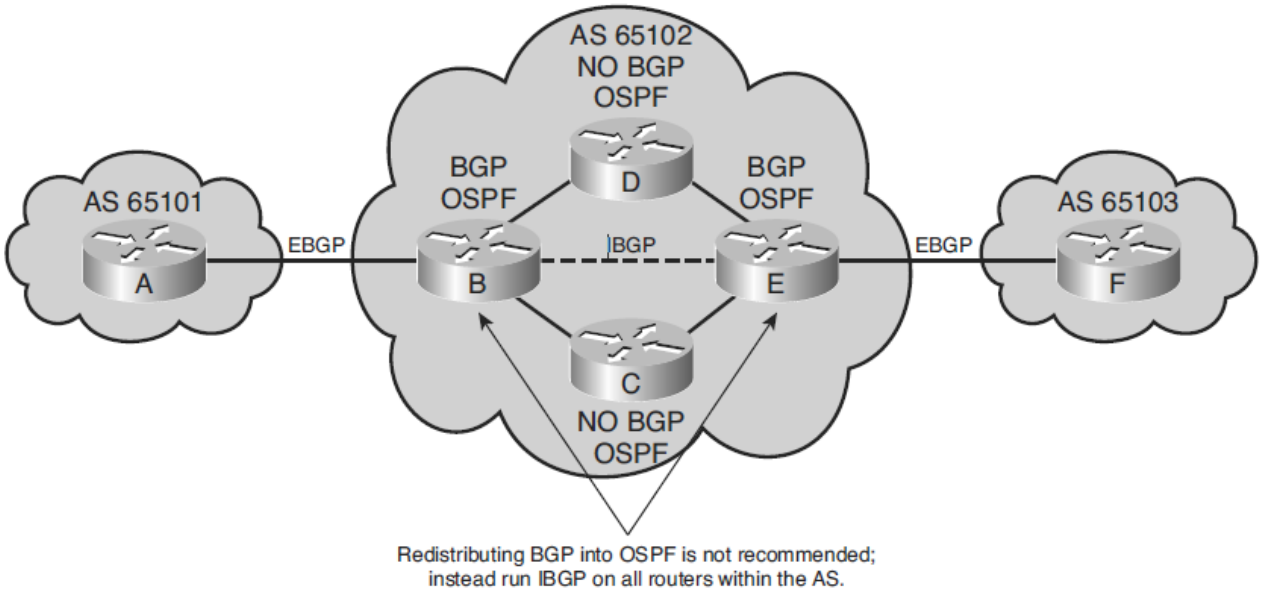


IBGP Neighbors Relationship Requirements

- **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.

IBGP 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.

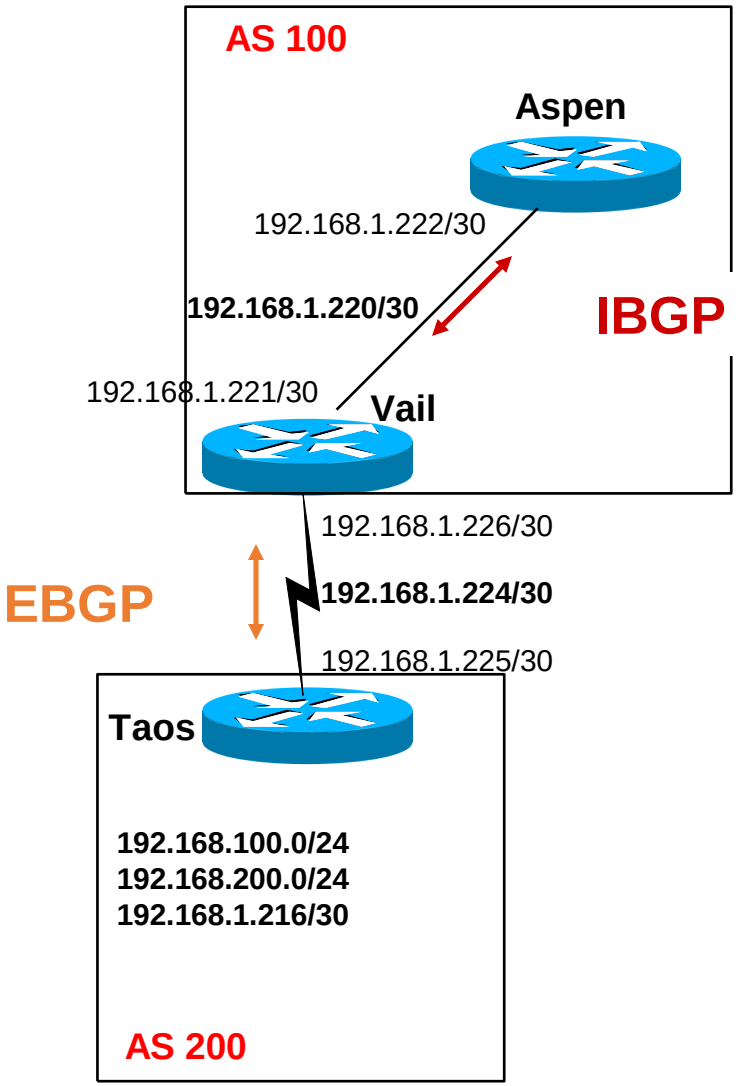


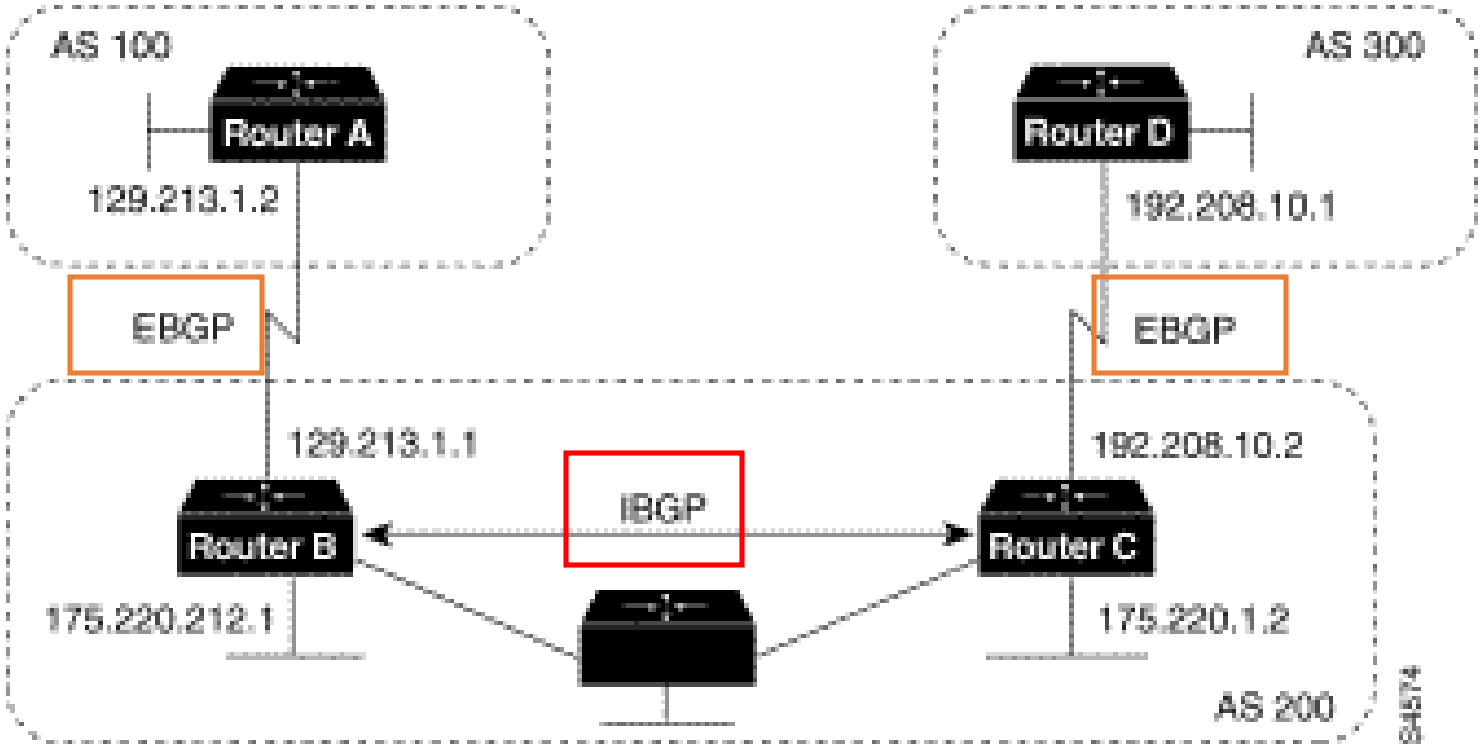
IBGP in a Nontransit AS

- 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.

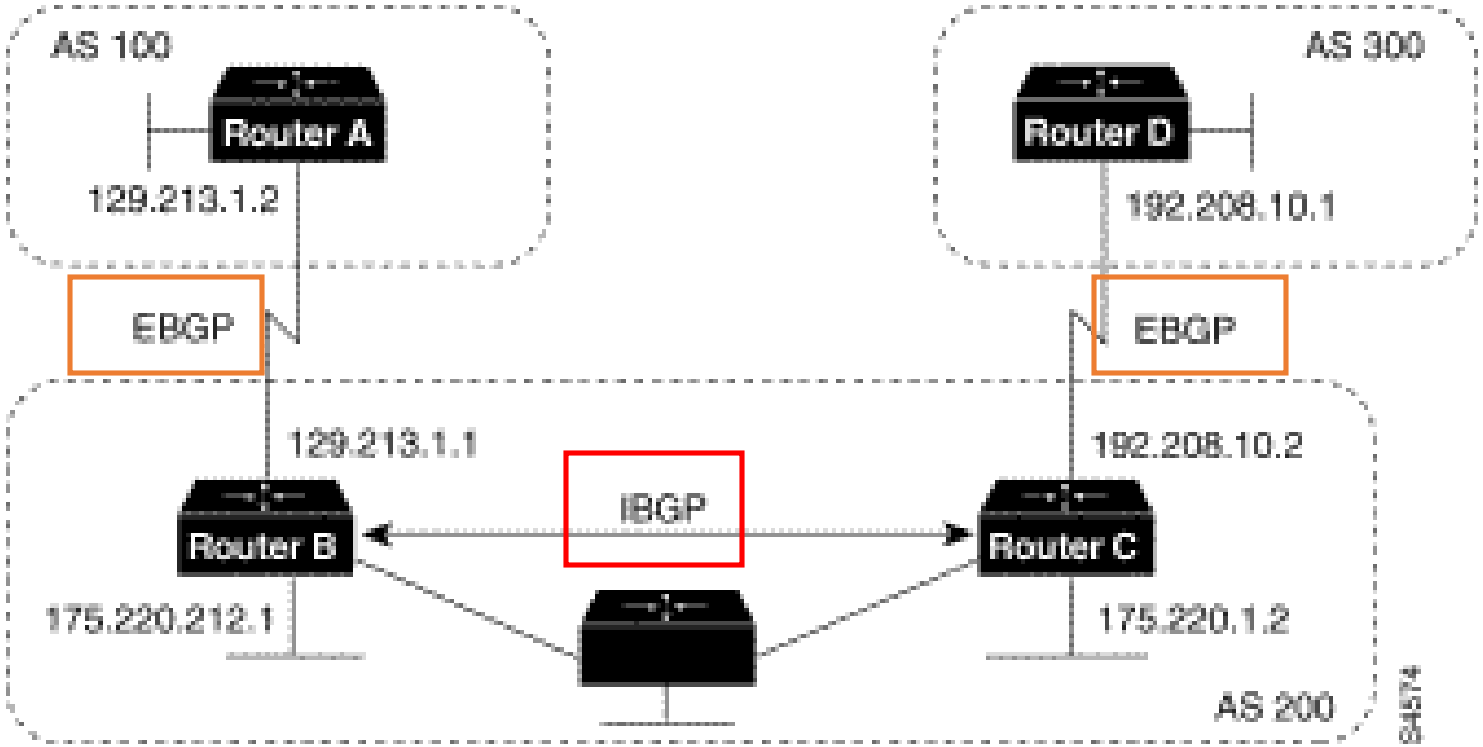
IBGP vs EBGP

- 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





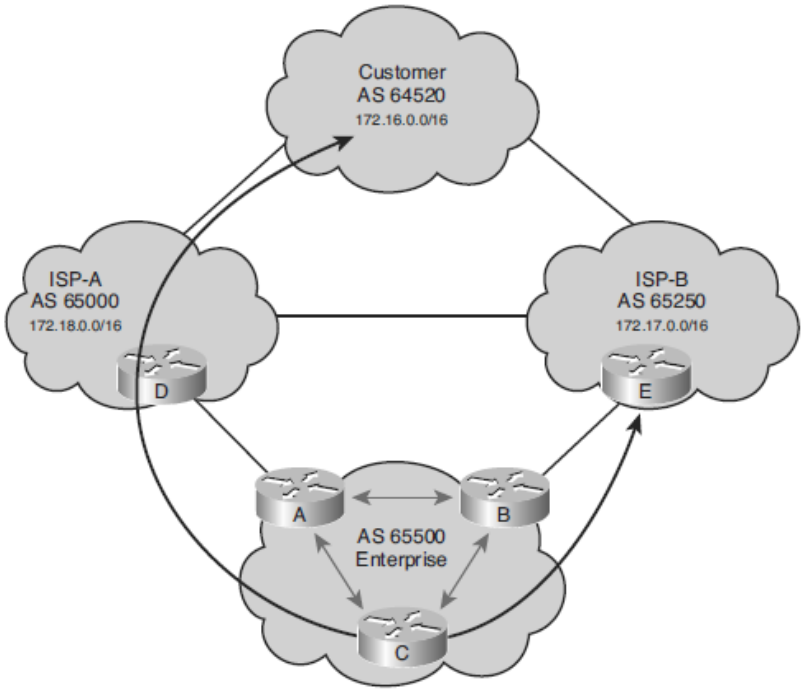
- Routers A and B are running **EBGP (BGP)**, and Routers B and C are running **IBGP**.
- Note that the **EBGP (BGP)** peers are directly connected and that the **IBGP** peers are not. (They can be.)
- As long as there is an **IGP** running that allows the two neighbors to reach one another, IBGP peers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



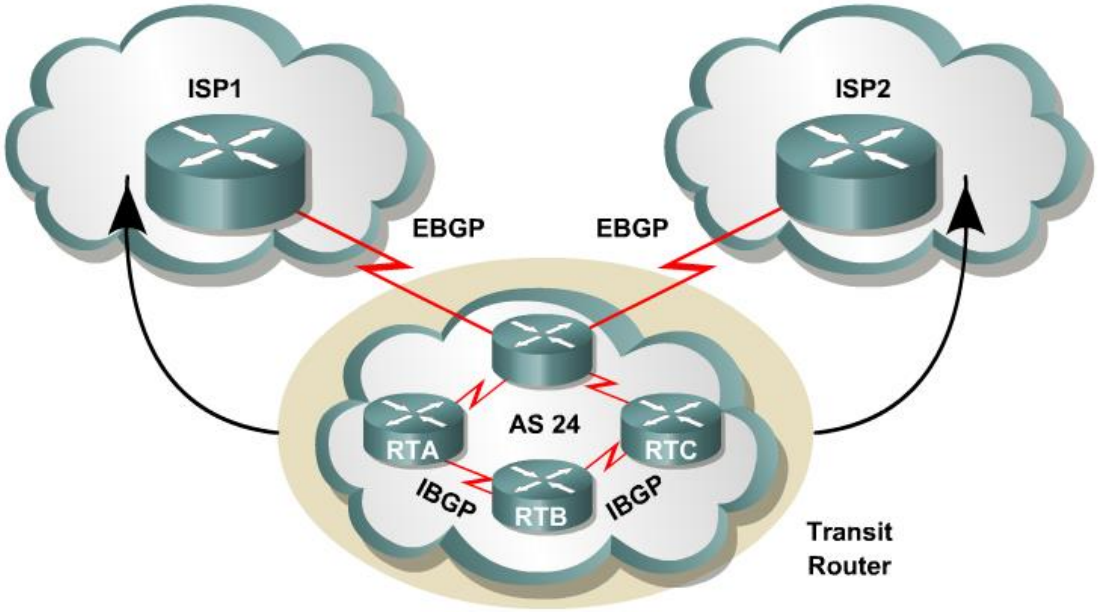
- All **BGP** speakers within an AS must establish a peer relationship with each other, that is, the **BGP** speakers within an AS must be fully meshed logically. (later)
- BGP4 provides two techniques that alleviate the requirement for a logical full mesh: confederations and route reflectors. (later)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



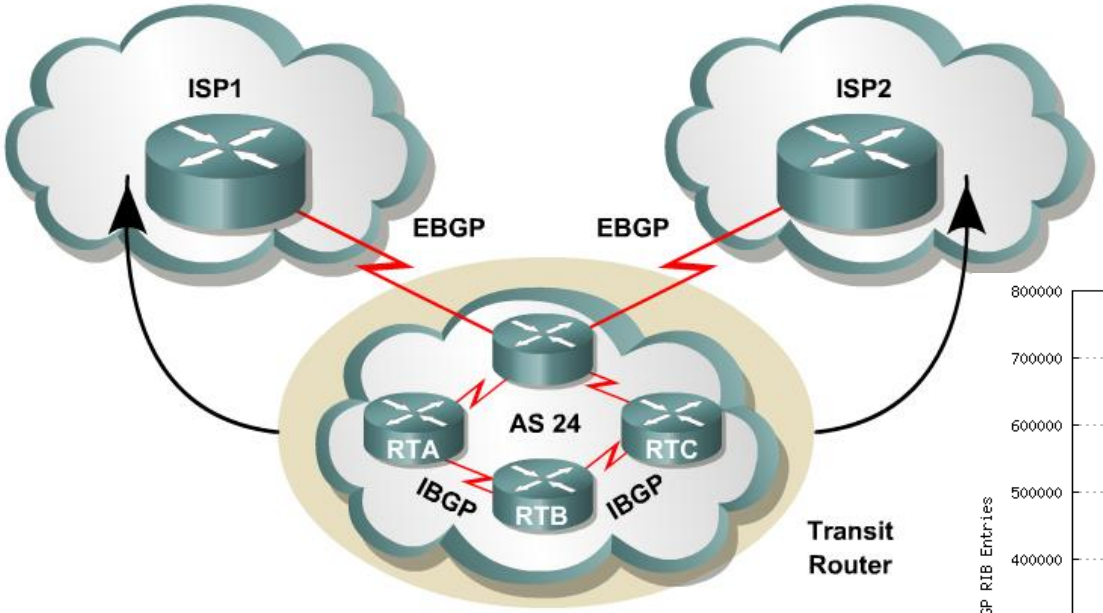
- Creating a BGP "peering" relationship involves an interesting combination of trust and mistrust.
- You must trust the network administrator on that end to know what they are doing.
- At the same time, if you are smart, you will take every practical measure to protect yourself in the event that a mistake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

BGP Hazards – Doyle, Routing TCP/IP

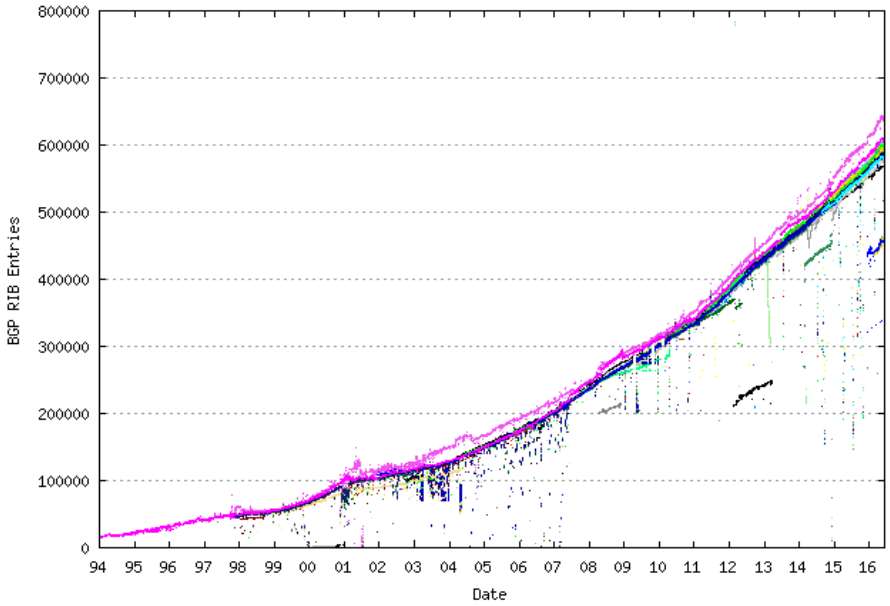
- Your ISP will show little patience with you if you make mistakes in your BGP configuration.
- Suppose, for example, that through some misconfiguration you advertise 207.46.0.0/16 to your ISP.
- On the receiving side, the ISP does not filter out this incorrect route, allowing it to be advertised to 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

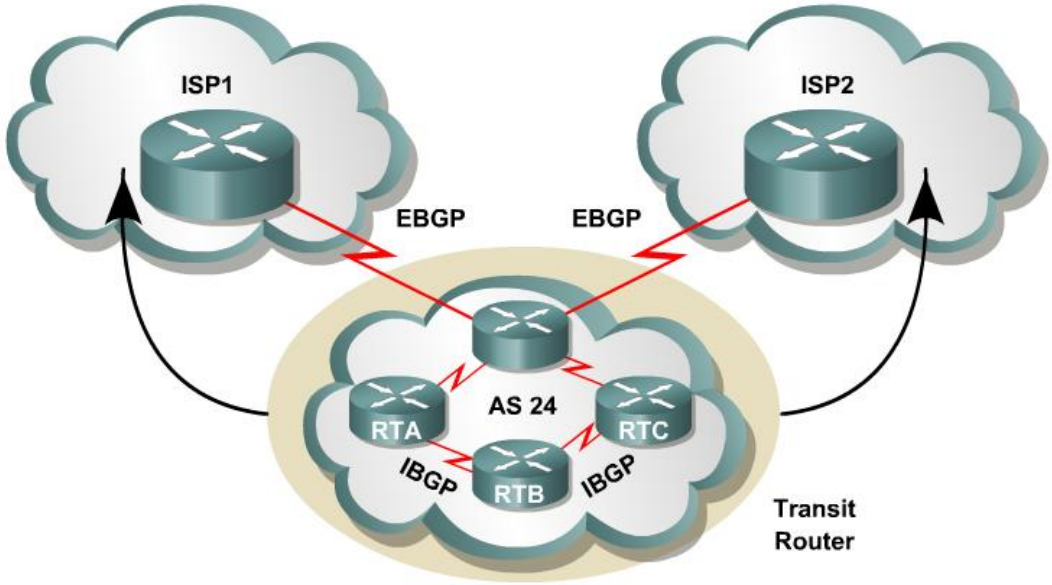


- Internal routing protocols announce a list of networks and the metrics to get to each network.
- In contrast, BGP routers exchange network reachability information, called **path vectors**, made up of **path attributes**.



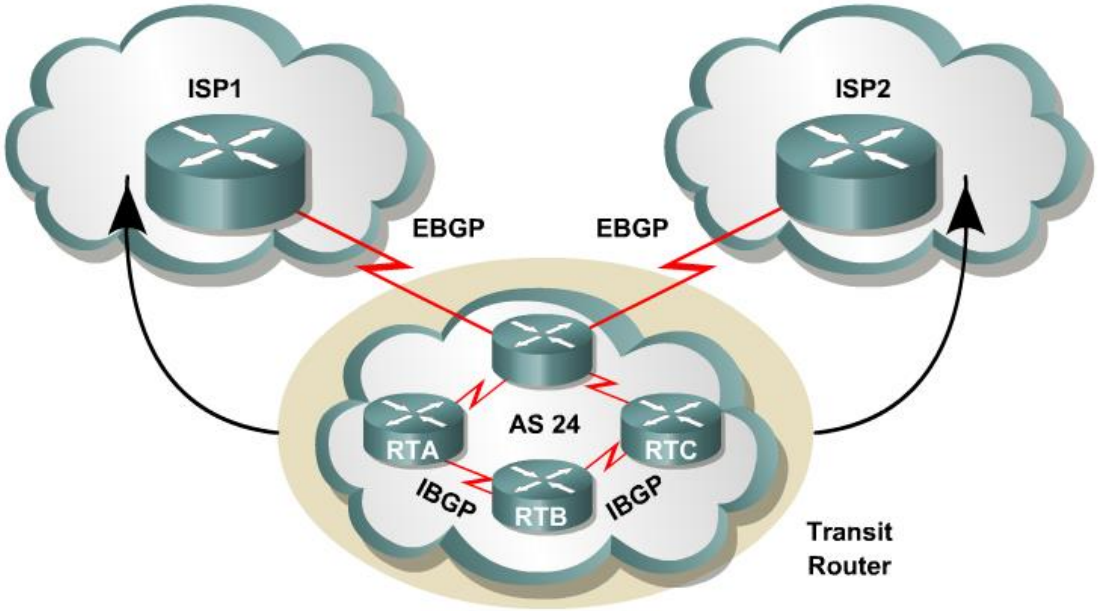
- BGP is a **path vector** routing protocol.
- BGP version 4 (BGP-4) is the latest version of BGP.
  - Defined in RFC 4271.
  - Supports supernetting (CIDR) and VLSM .
- BGP4 and CIDR prevent the Internet routing table from becoming too large.
  - Without CIDR, the Internet would have few millions entries.
  - 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.

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.

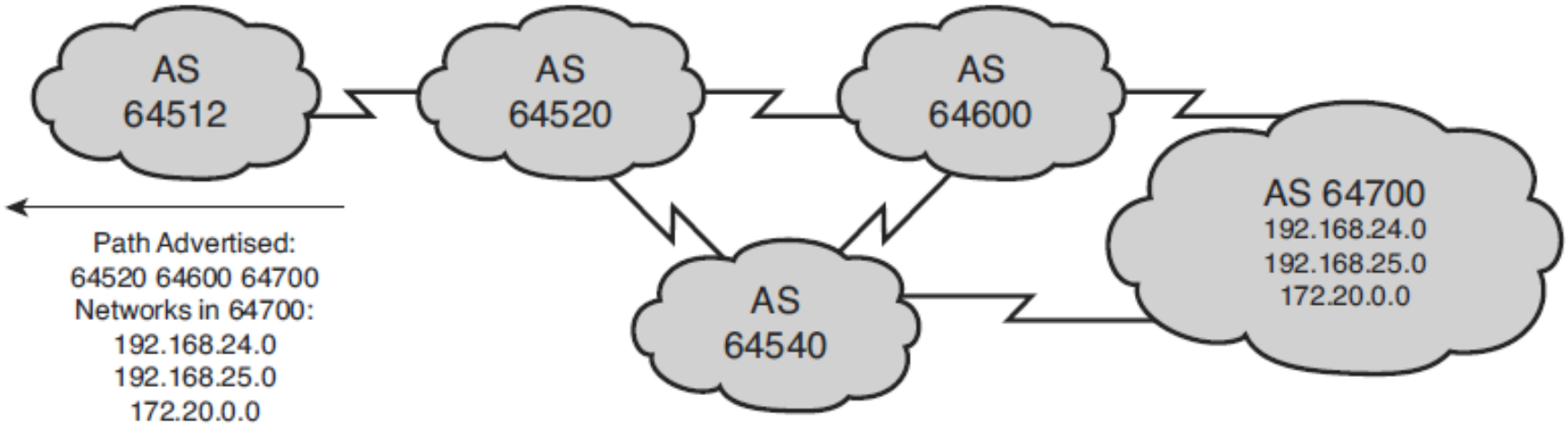
http://bgpupdates.potaroo.net/instability/bgpupd.html

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).

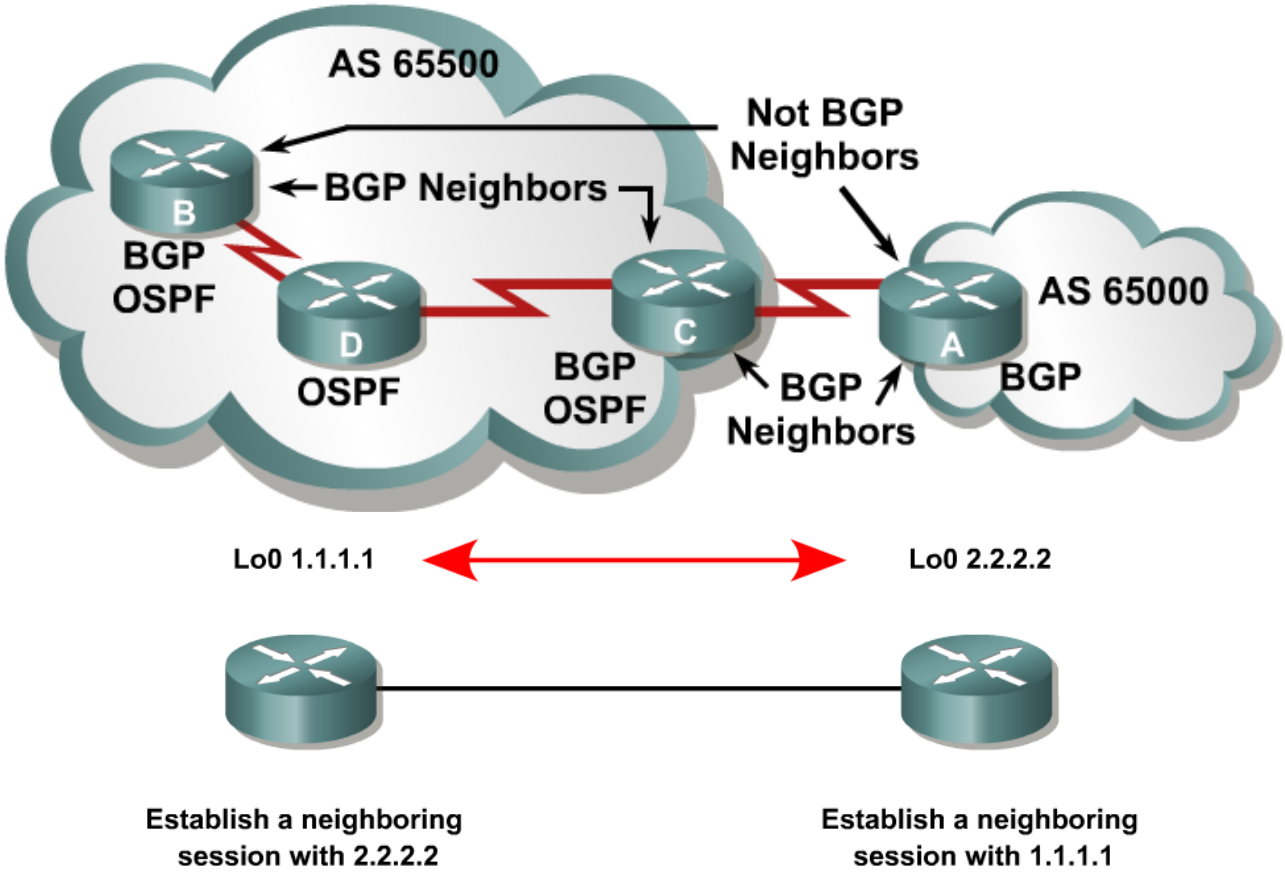


BGP Basics
Peers = Neighbors

- 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.

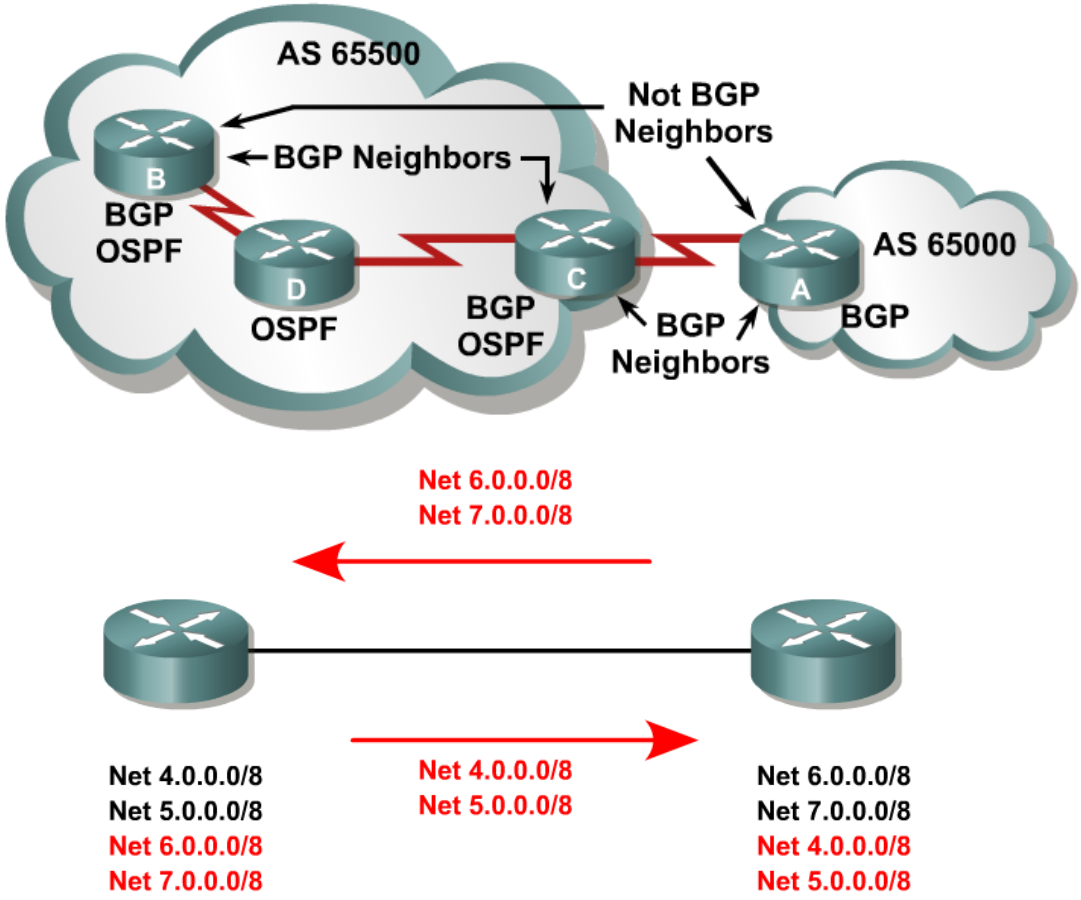
BGP Operational Overview

- 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**.

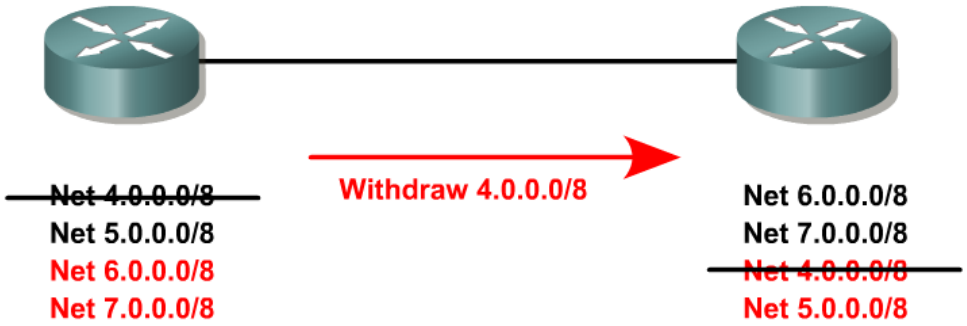


BGP Operational Overview

- 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.

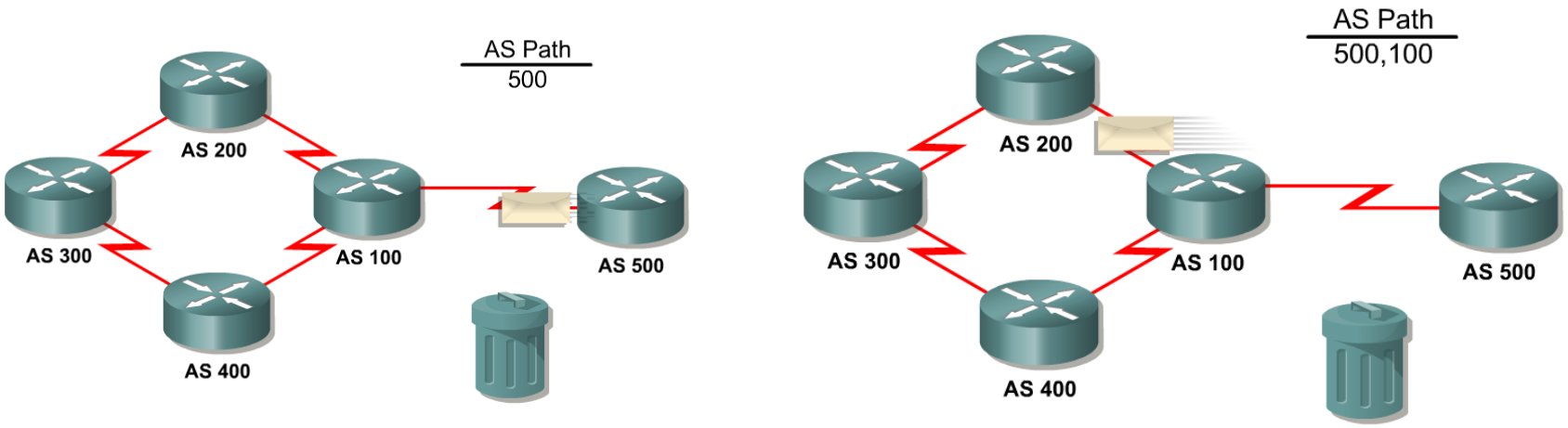


Withdrawn Routes



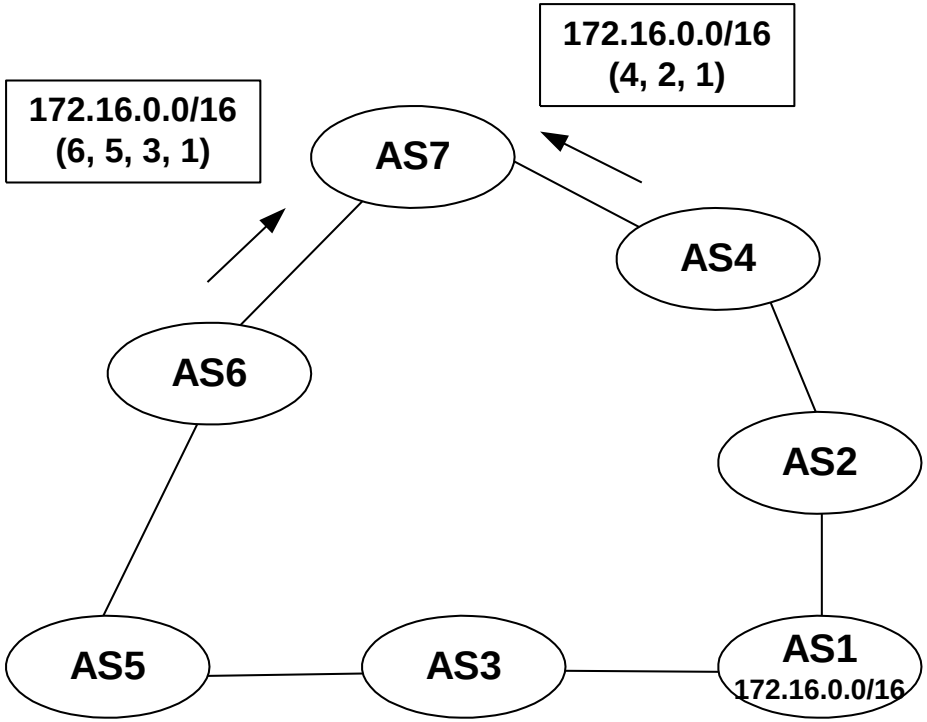
- The information for network reachability can change, such as when a route becomes unreachable or a better path becomes available.
- BGP informs its neighbors of this by withdrawing the invalid routes and injecting the new routing information.
- Withdrawn routes are part of the update message. BGP routers 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



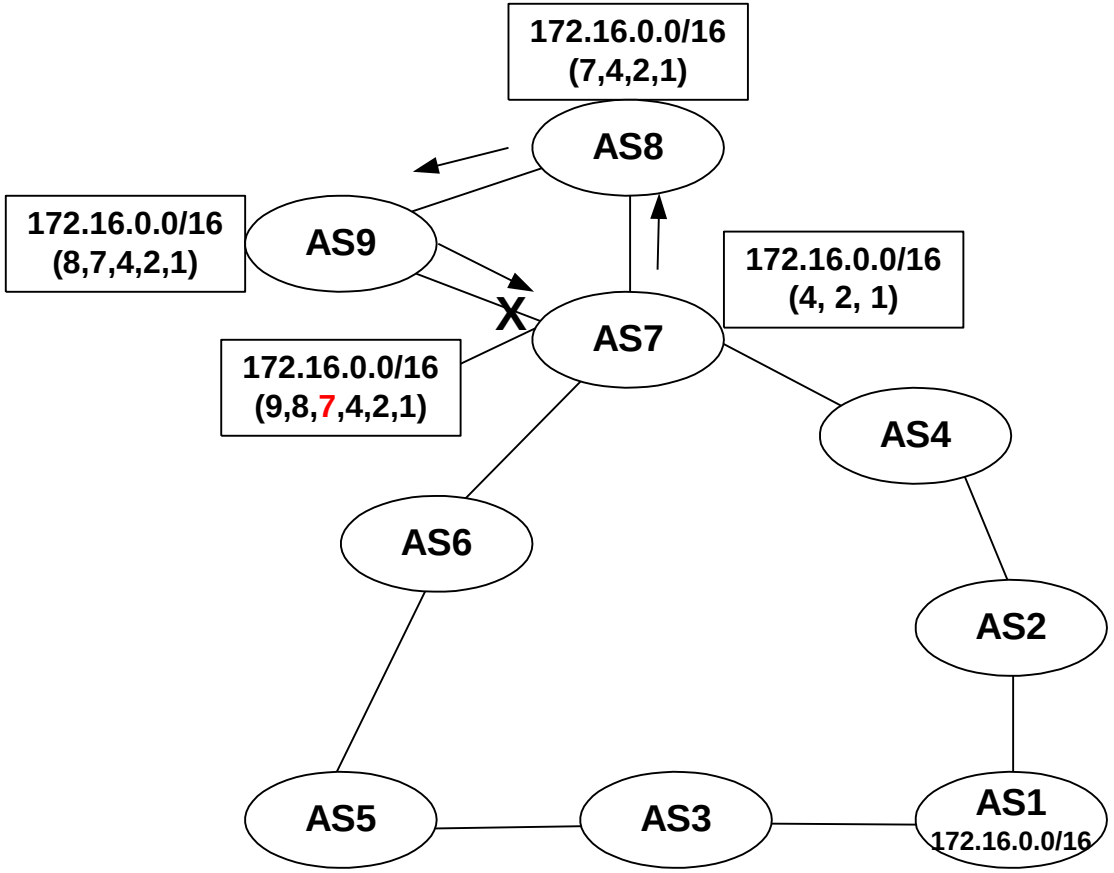
- To guarantee loop free path selection, BGP constructs a graph of autonomous systems based on the information exchanged between BGP neighbors.
- BGP views the whole internetwork as a graph, or tree, of autonomous systems.
- The connection between any two systems forms a path.
- The collection of path 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.

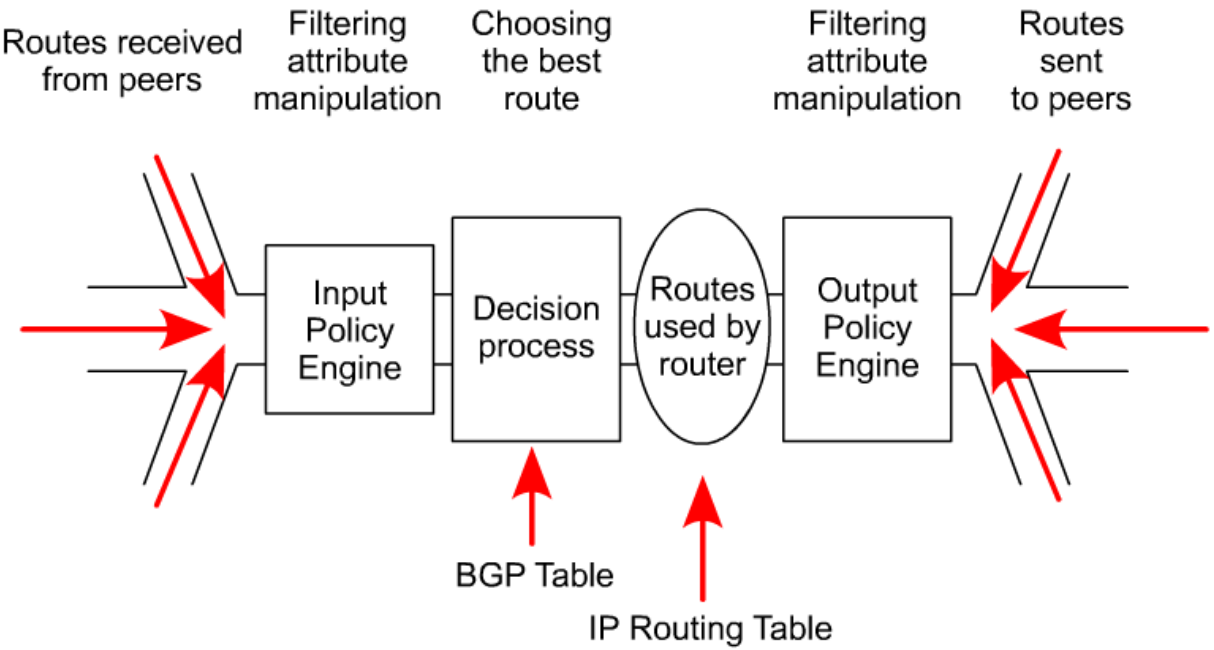
BGP Databases

- 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

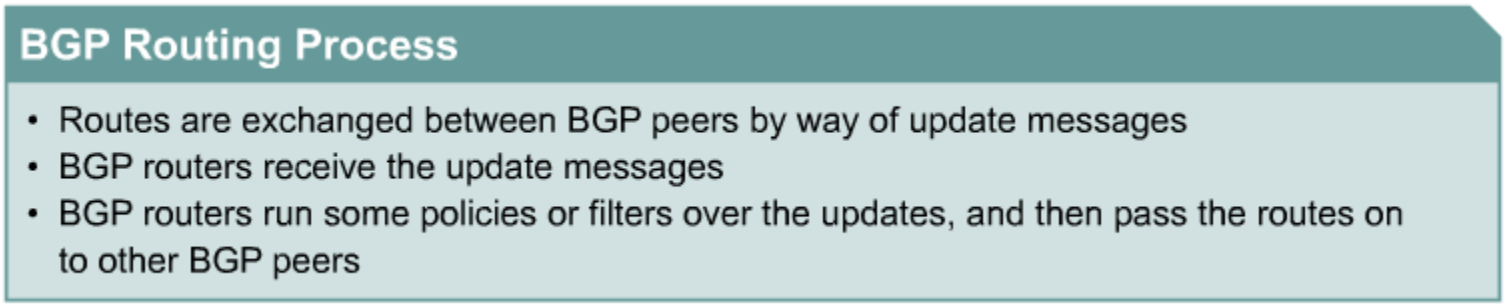
- BGP keeps its own table for storing BGP information received from and sent BGP neighbors.
  - This table is also known as the BGP table, BGP topology table, BGP topology database, BGP routing table, and the BGP forwarding database.
- The router offers the best routes from the BGP table to the IP routing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



- 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.

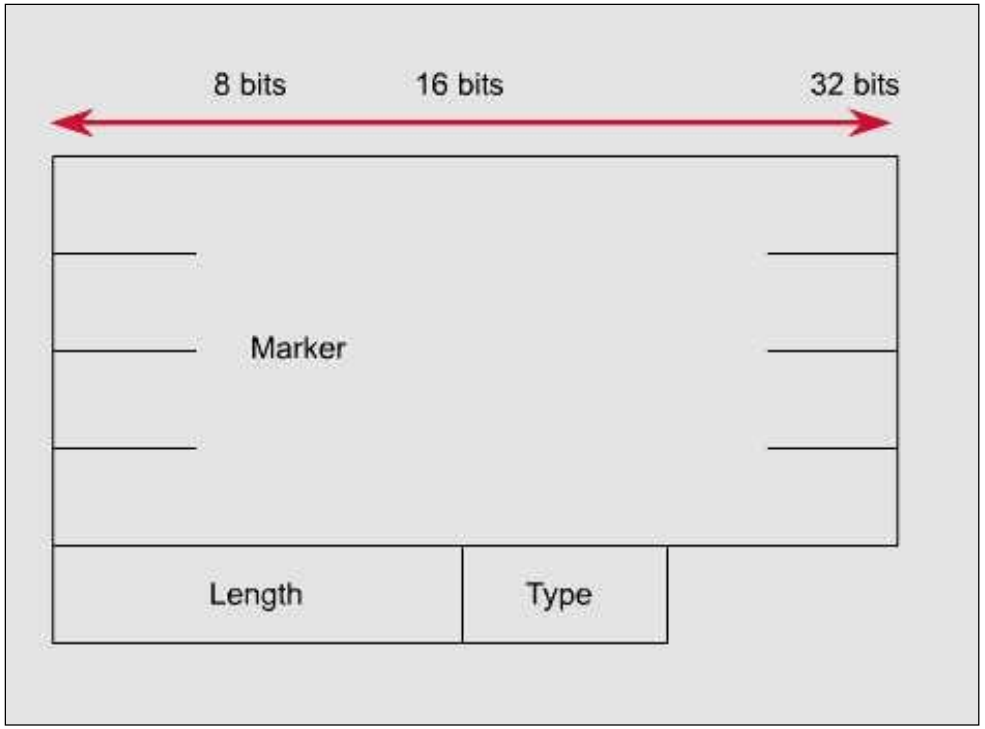
BGP Message Types

- 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 exchanges 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

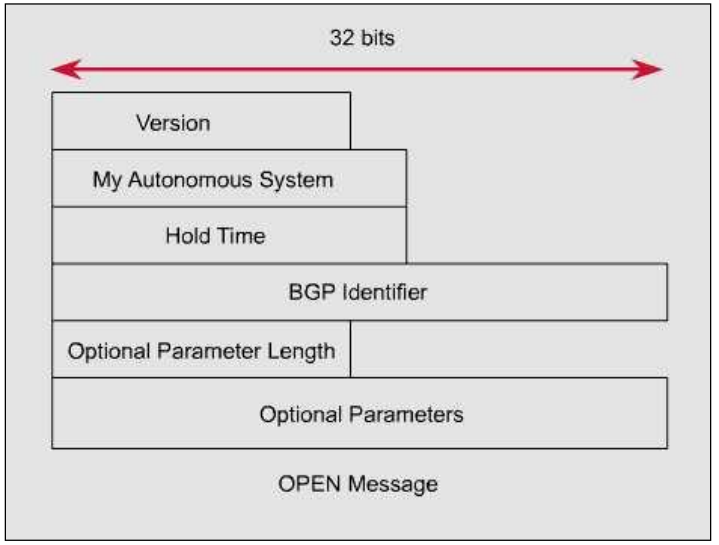
Notification Message

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

Type 2:
BGP Keepalive Message

Keep alive message transmitting

- 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

				Unreachable Routes Information		Path Attributes Information		NLRI Information
Octets	16	2	1	2	Variable	2	Variable	Variable
	Marker	Length	Type	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.

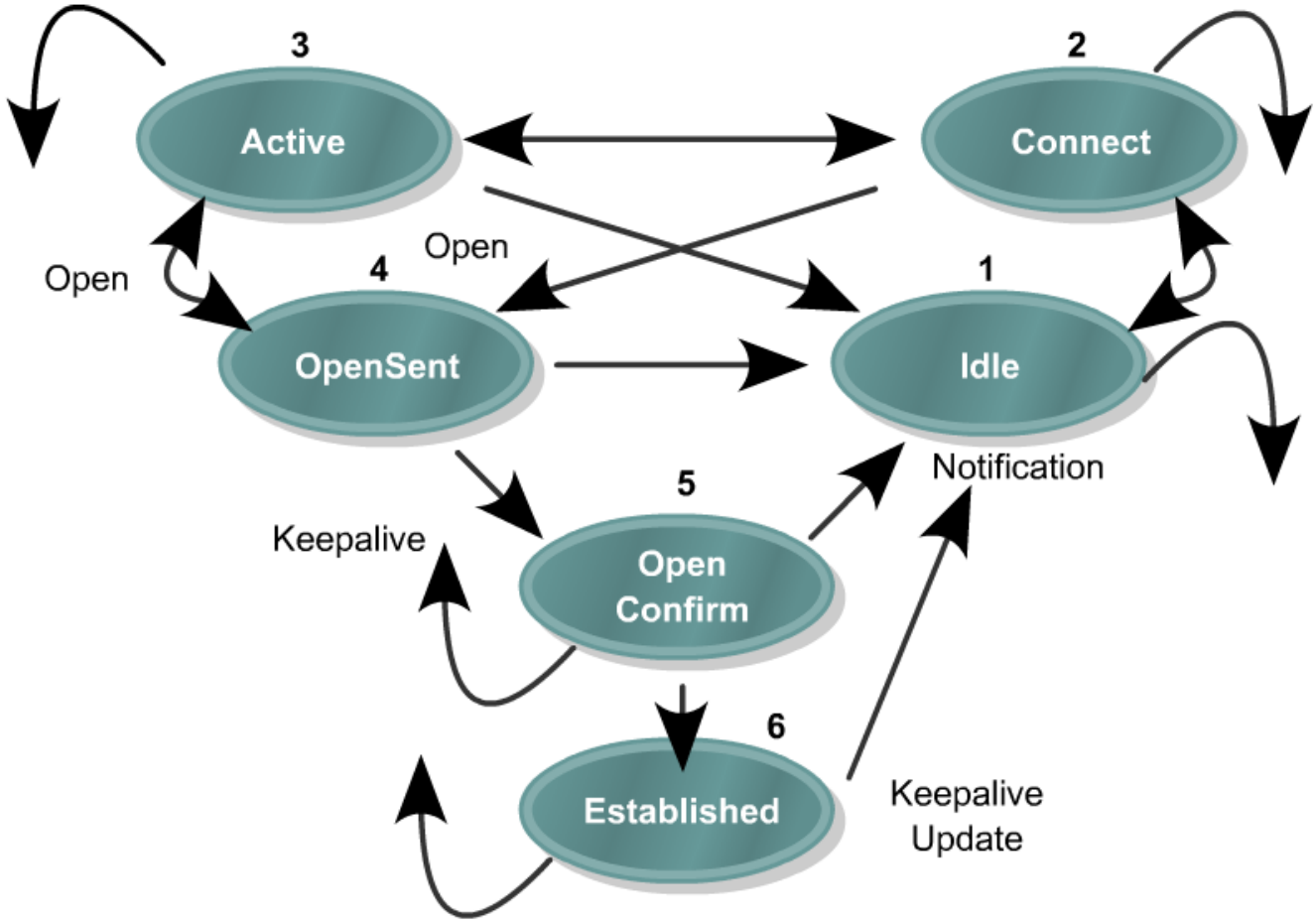
Octets	16	2	1	1	1	Variable
	Marker	Length	Type	Error Code	Error Sub-code	Diagnostic Data

Type 4:
BGP Notification Message

- Sample error codes and their associated subcodes.

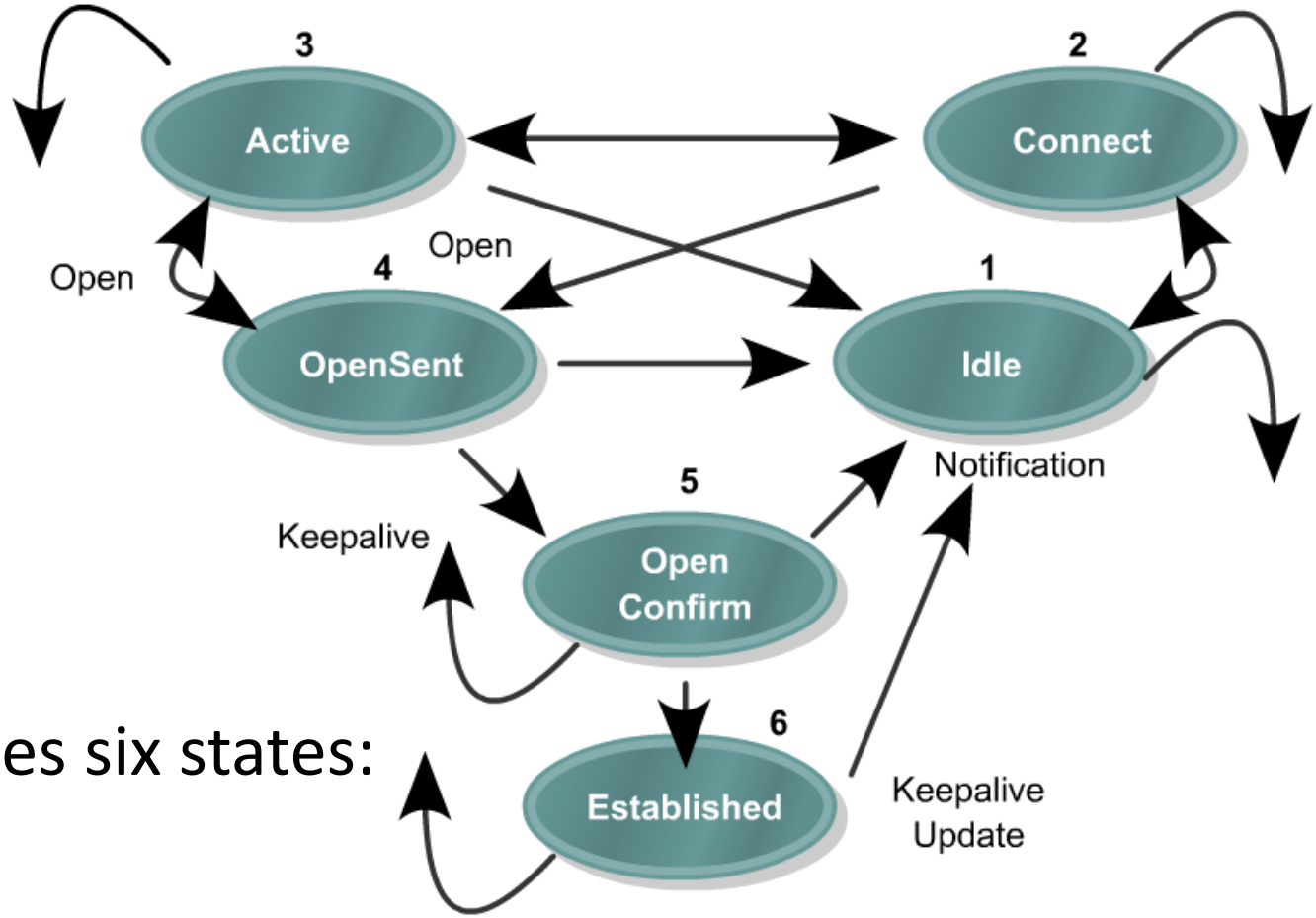
Error Code	Error Subcode
1--Message Header Error	1--Connection Not Synchronized
	2--Bad Message Length
	3--Bad Message Type
2--OPEN Message Error	1--Unsupported Version Number
	2--Bad Peer AS
	3--Bad BGP Identifier
	4--Unsupported Optional Parameter
	5--Authentication Failure
	6--Unacceptable Hold Time
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).

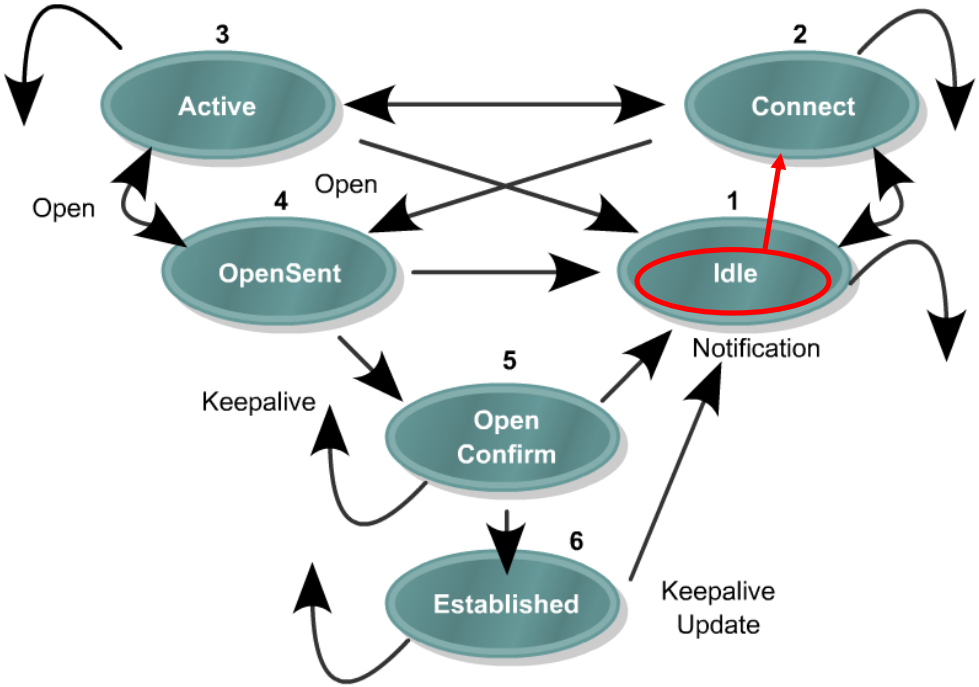
BGP FSM



BGP FSM includes six states:

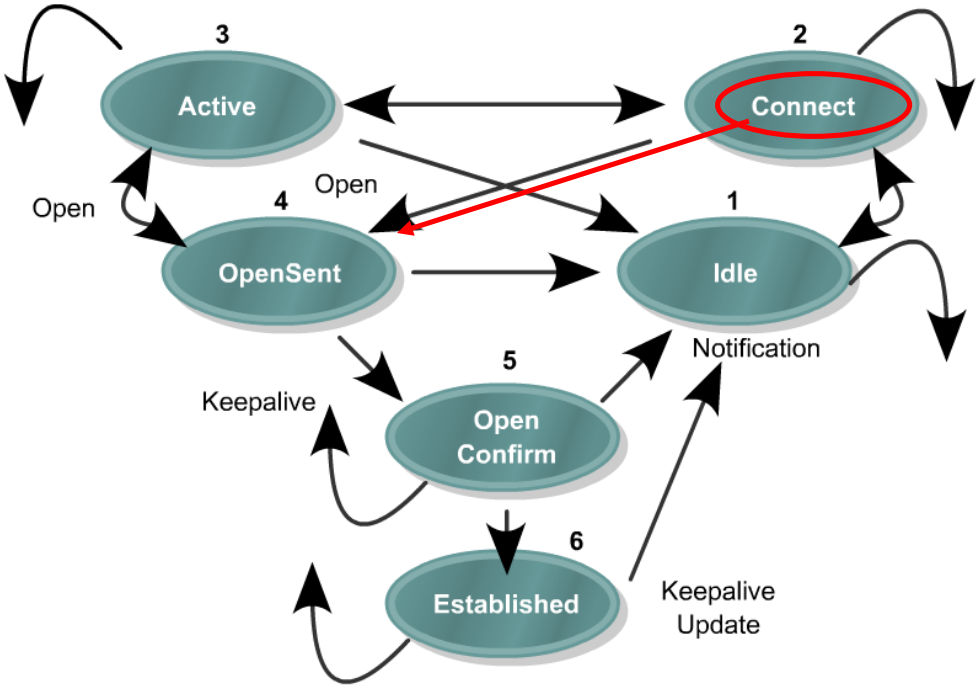
1. **Idle**
2. **Connect**
3. **Active**
4. **OpenSent**
5. **Open Confirm**
6. **Established**

Idle State



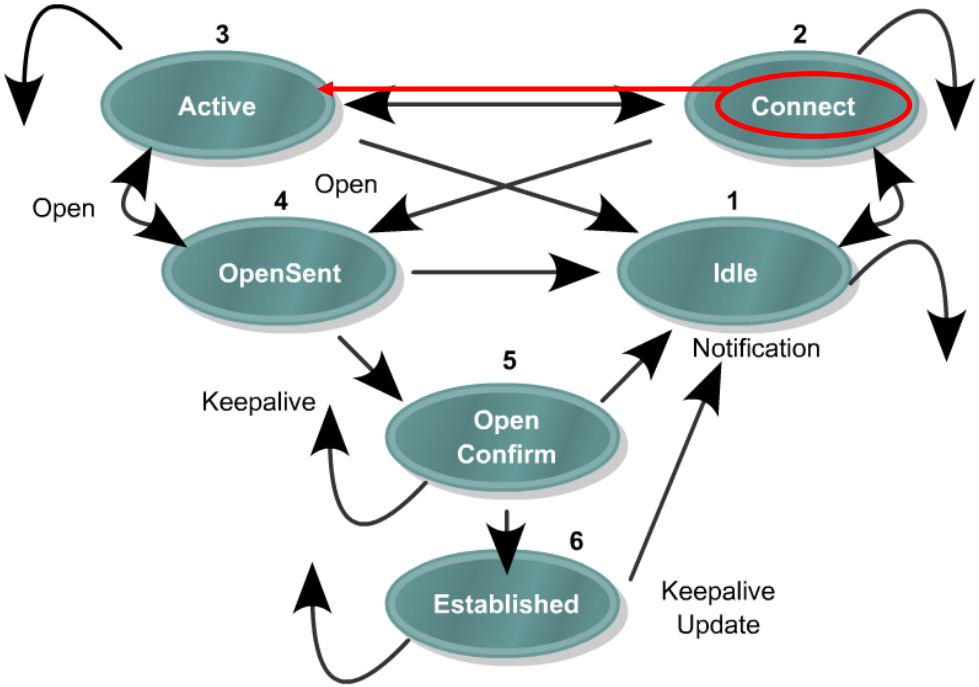
- BGP always begins in the **Idle** state, in which it refuses all incoming connections.
- It is normally initiated by an administrator or a network event.
- When Start event occurs, the BGP process:
  - Initializes all BGP resources
  - Starts the ConnectRetry timer
  - Initializes a TCP connection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

Connect State



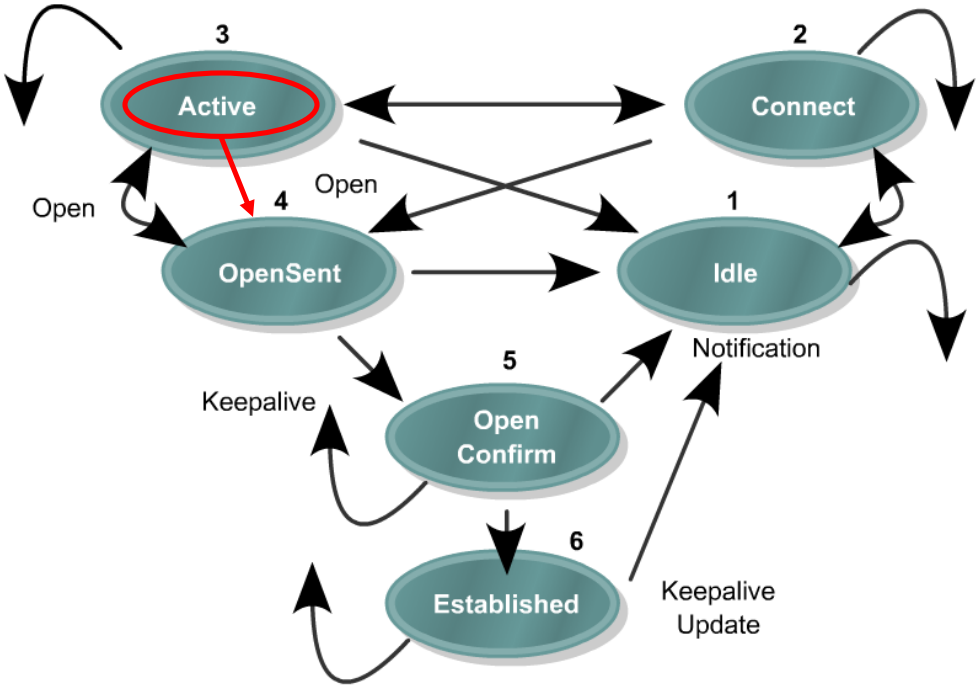
- In this state, the BGP process is waiting for the TCP connection to be completed.
- If the connection is **successful**, the BGP process:
  - Clears the ConnectRetry timer
  - Completes initialization
  - Sends an **Open message** to the neighbor
  - Transitions to the **OpenSent** state

Connect State



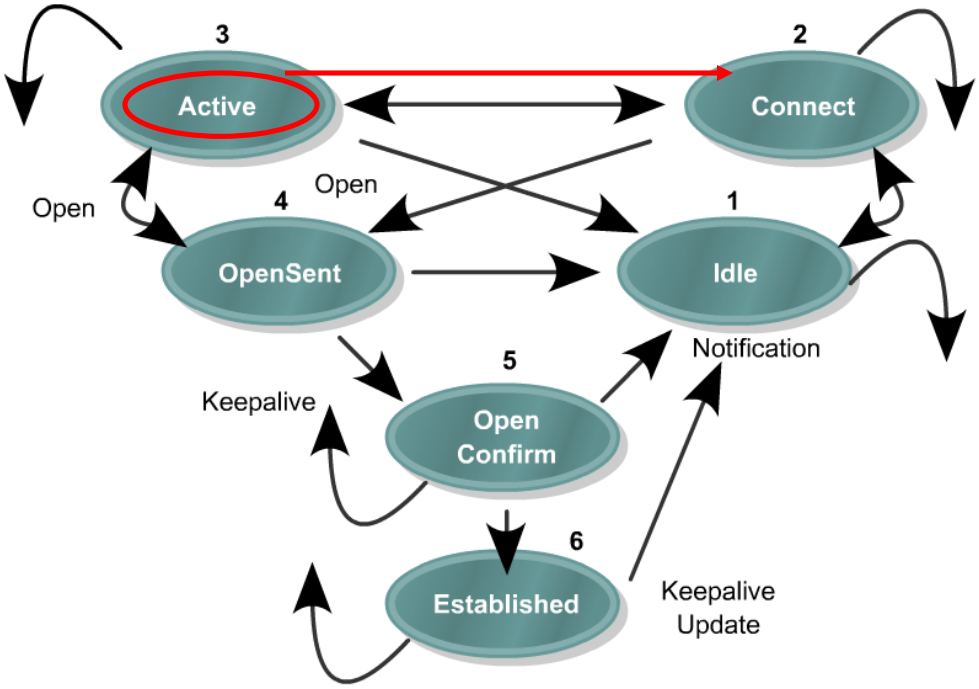
- If the connection is **unsuccessful**, the BGP process:
  - Continues to listen for a connection to be initiated by the neighbor
  - Resets the ConnectRetry timer
  - Transitions to the **Active** state

Active State



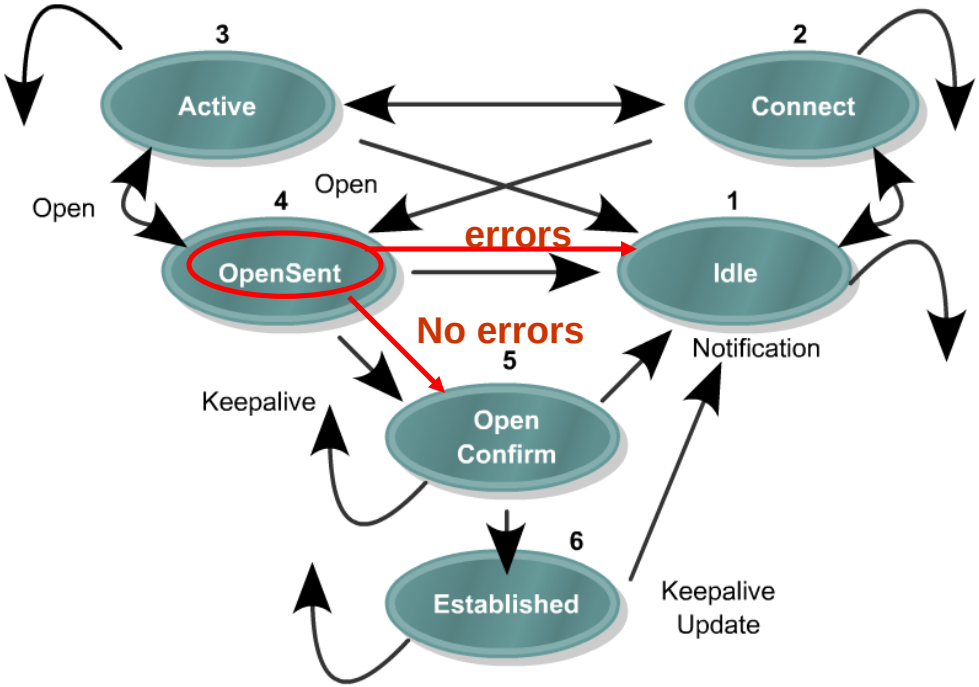
- In this state, the BGP process is trying to initiate a TCP connection with the neighbor.
- If the TCP connection is **successful**:
  - Clears the ConnectRetry timer
  - Completes initialization
  - Sends an **Open message** to the neighbor
  - Transitions to the **OpenSent** state

Active State



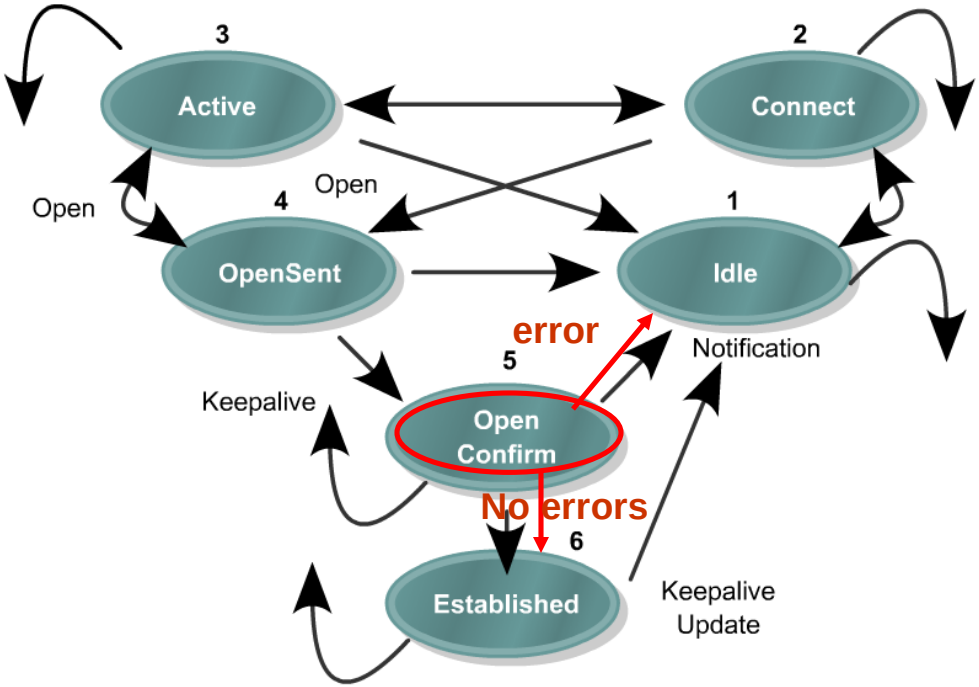
- If the ConnectRetry timer expires while BGP is in the Active State, the BGP process:
  - Transitions back to the **Connect** state
  - Resets the ConnectRetry timer
- In general, a neighbor state that is switching between "**Connect**" and "**Active**" is an indication that something is wrong and that 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



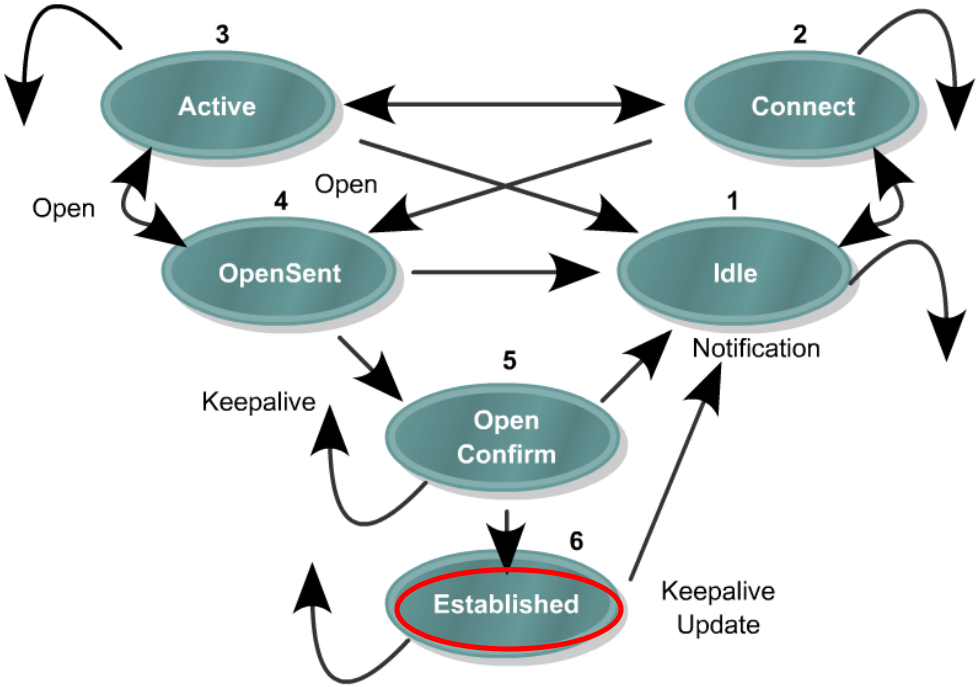
- In this state an **Open message** has been sent and BGP is waiting to hear an Open message from its neighbor.
- When an **Open message** is received, all its fields are checked.
  - **If errors** exist, a **Notification message** is sent and the state transitions to **Idle**.
  - **If no errors** exist, 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

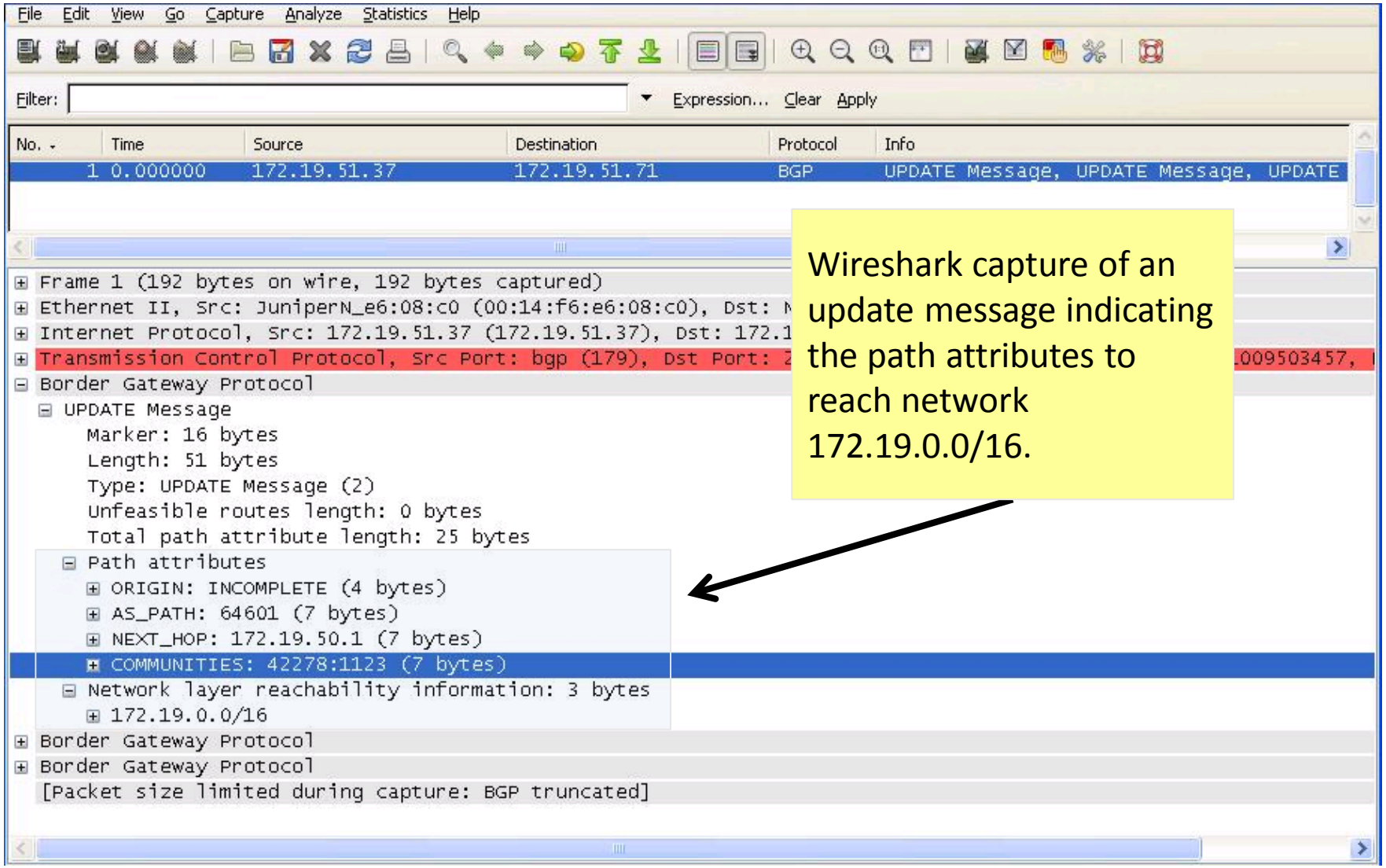
- Type code 1 ORIGIN
- Type code 2 AS_PATH
- Type code 3 NEXT_HOP
- Type code 4 MULTI_EXIT_DISC
- Type code 5 LOCAL_PREF
- Type code 6 ATOMIC_AGGREGATE
- Type code 7 AGGREGATOR
- Type code 8 Community (Cisco-defined)
- Type code 9 Originator-ID (Cisco-defined)
- Type code 10 Cluster list (Cisco-defined)

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



Attributes

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

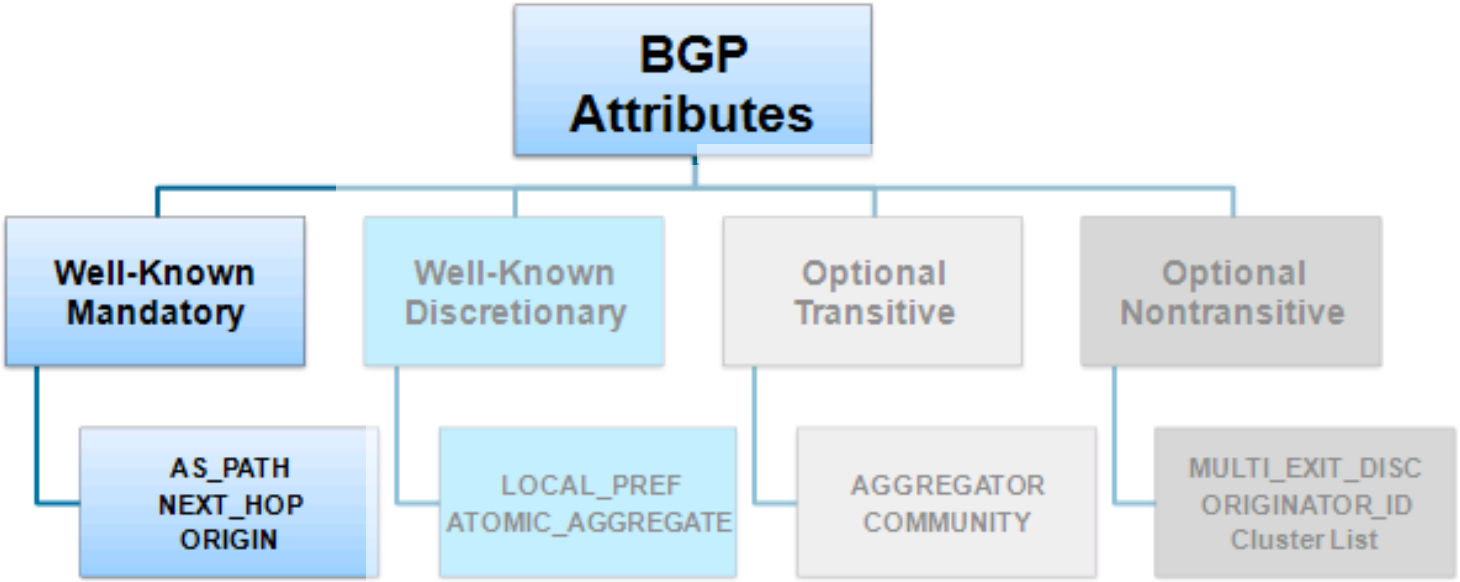
Attribute	EBGP	IBGP
AS_PATH	Well-known Mandatory	Well-known Mandatory
NEXT_HOP	Well-known Mandatory	Well-known Mandatory
ORIGIN	Well-known Mandatory	Well-known Mandatory
LOCAL_PREF	Not allowed	Well-known Discretionary
ATOMIC_AGGREGATE	Well-known Discretionary	Well-known Discretionary
AGGREGATOR	Optional 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.

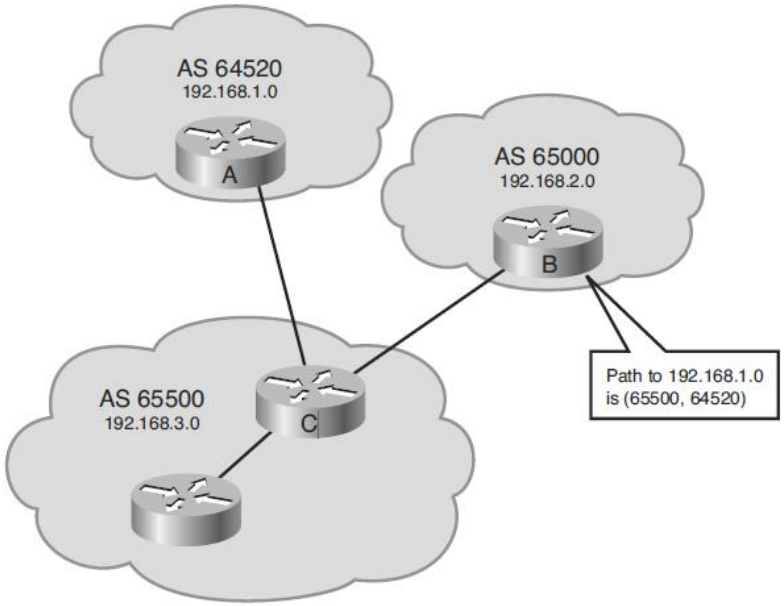
Well-Known Mandatory

- 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 ensures that all BGP implementations agree on a standard set of attributes.

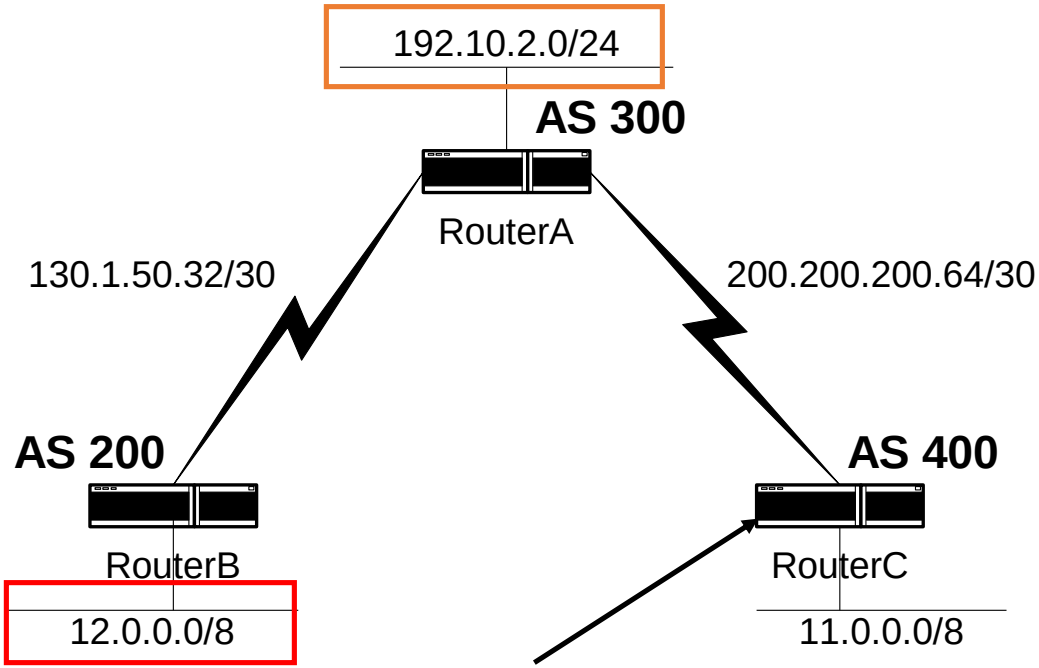


Well-Known Mandatory: AS_PATH

- 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.



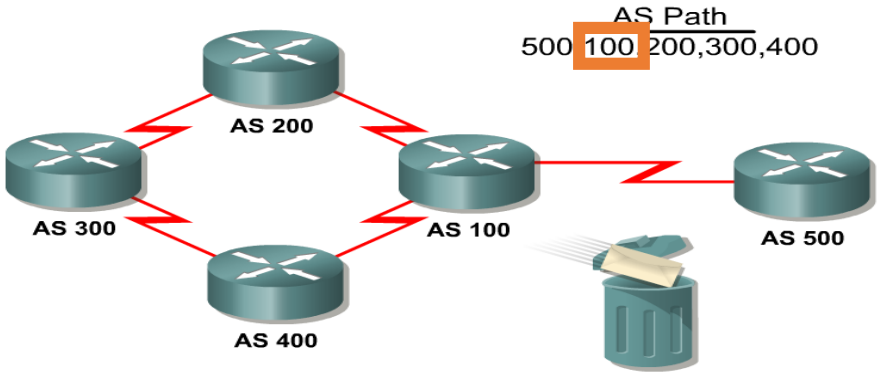
AS_PATH




```
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.



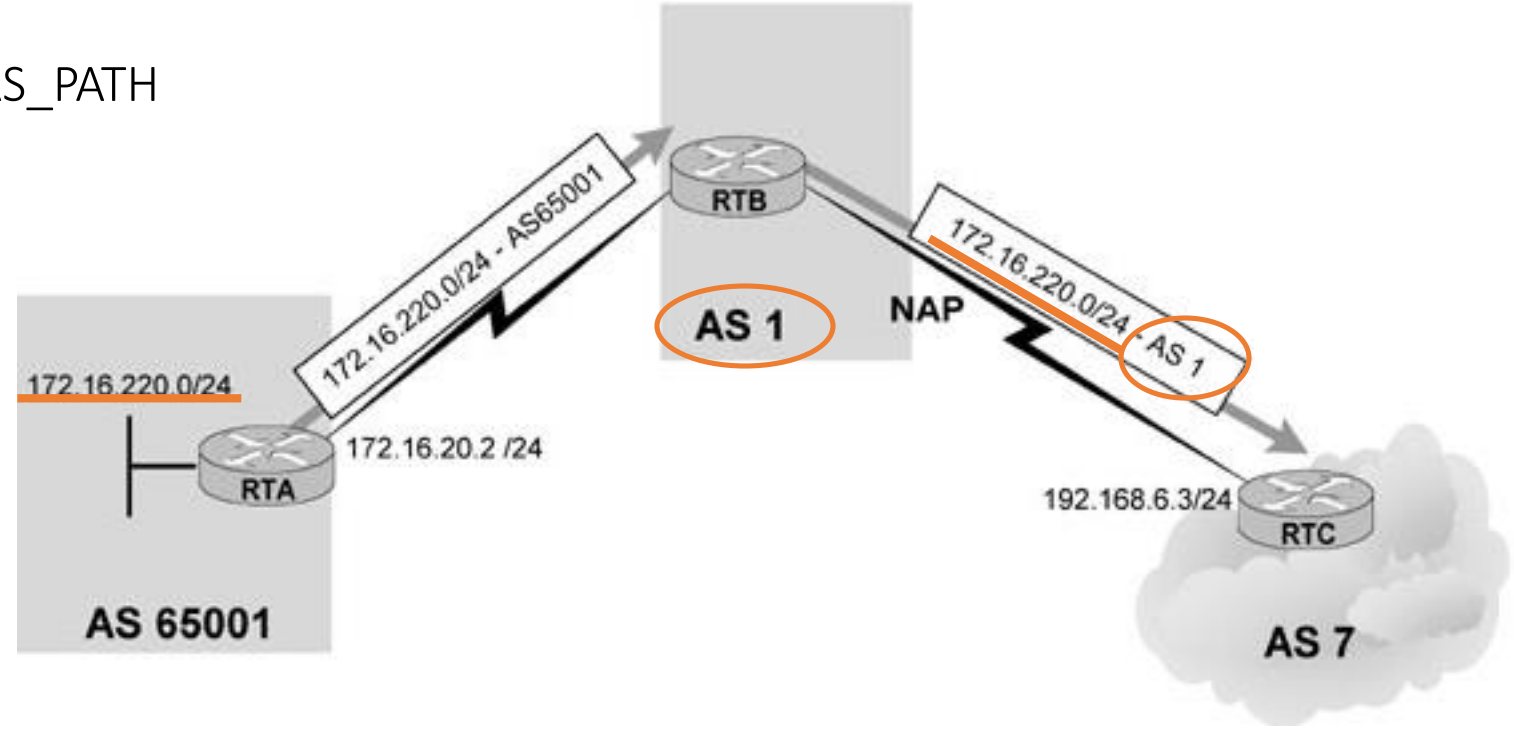
- **EBGP**: BGP speakers prepend their AS numbers when advertising routing updates to other autonomous systems (external peers).
- **IBGP**: When the route is passed to a BGP speaker within the same AS, the **AS_PATH** information is left intact.

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)

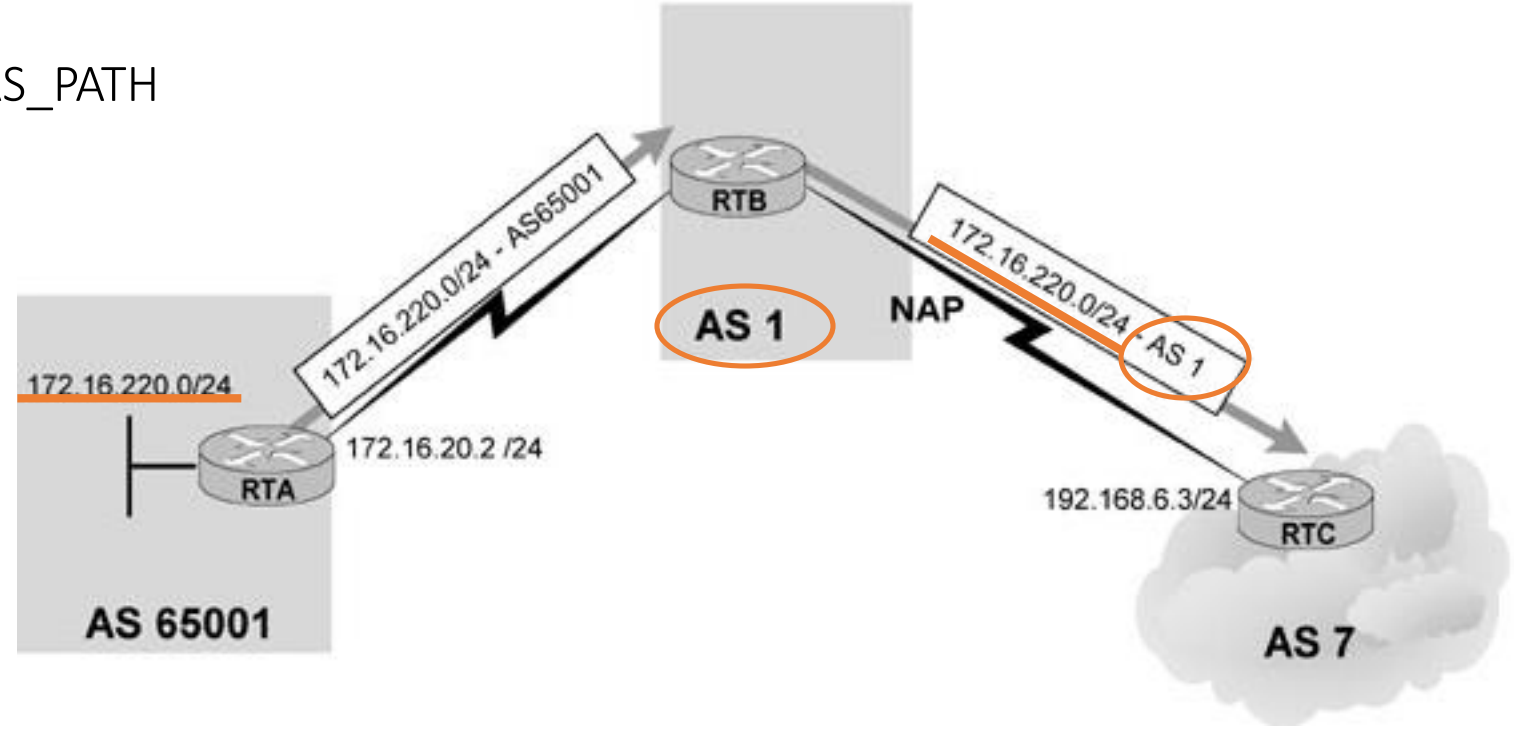
- **Private AS numbers** cannot be leaked to the Internet because they are not unique.
  - Cisco has implemented a feature, **remove-private-as,** to strip private AS numbers out of the **AS_PATH** list before the routes get propagated to the Internet.

AS_PATH



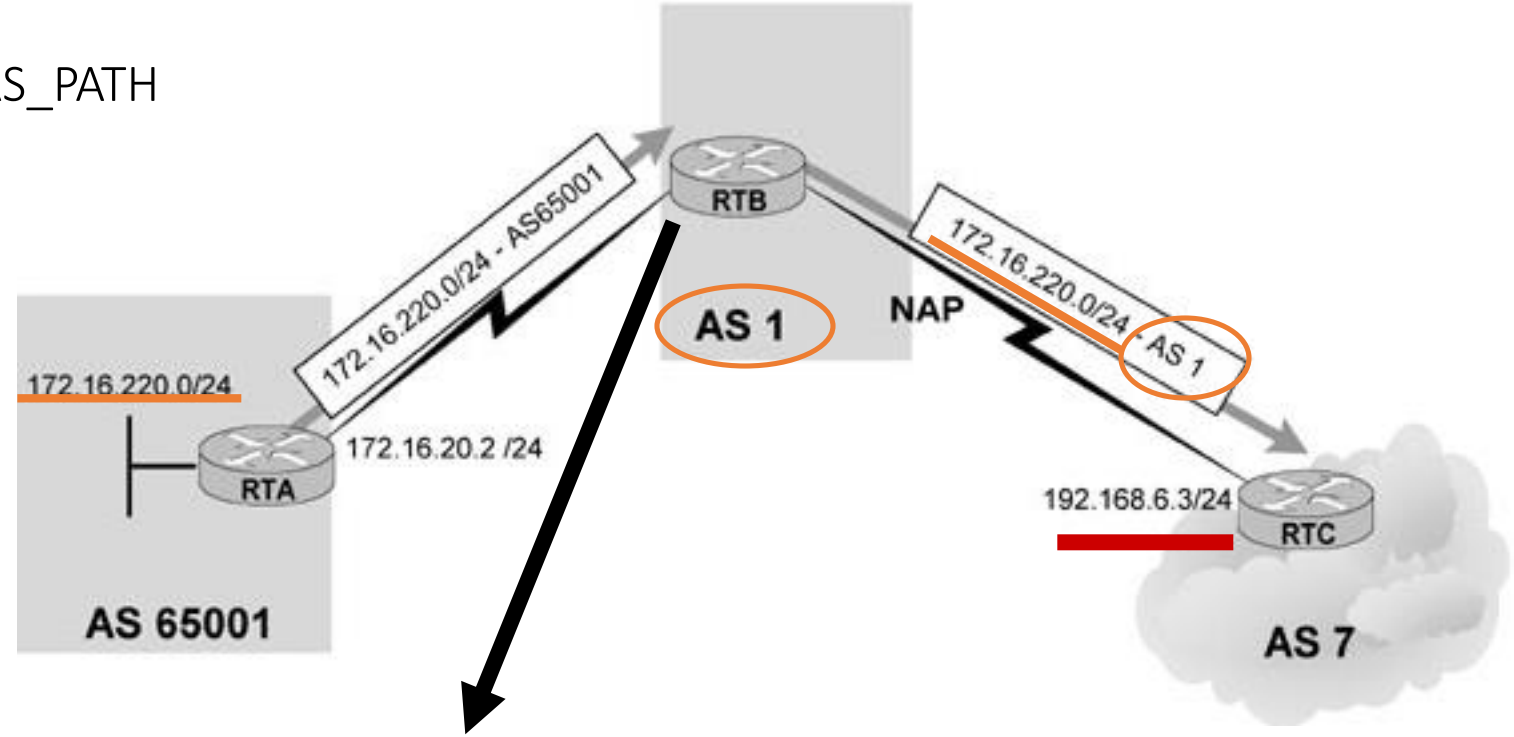
- AS1 is providing Internet connectivity to its customer AS 65001.
- Because the customer connects to only this provider and no plans to connect to an additional provider in the near future, the customer has been allocated a private AS number.
- **BGP will strip private AS numbers** only when propagating 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
```

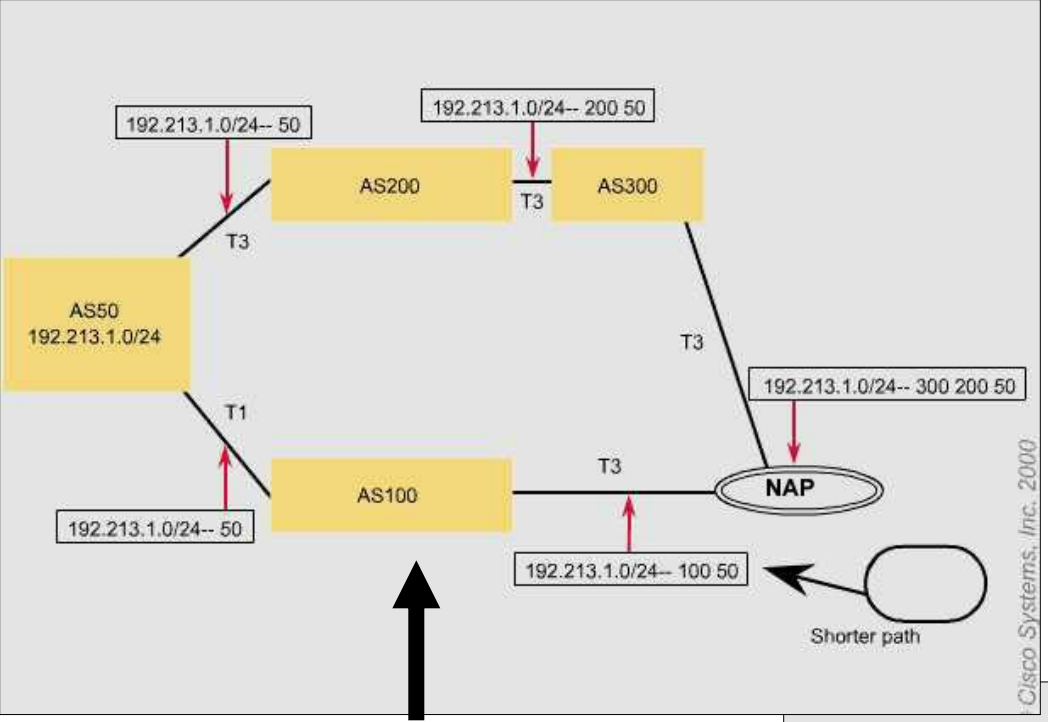
- Note how RTB is using the **remove-private-as** keyword in its neighbor connection to AS7.

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

AS_PATH - prepend

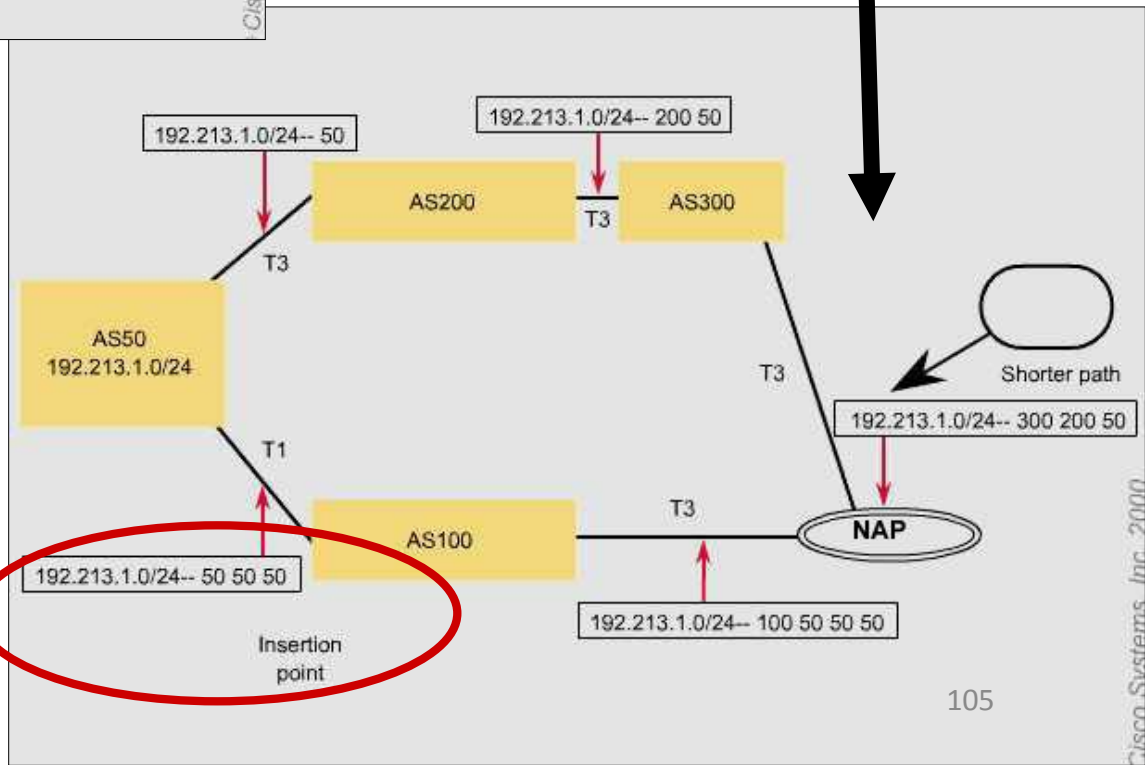
- **AS_PATH** information is manipulated to affect interdomain routing behavior.
- Because BGP prefers a shorter path over a longer one, system operators are tempted to change the path information by including dummy AS path numbers that would increase the path length and influence the traffic trajectory 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.

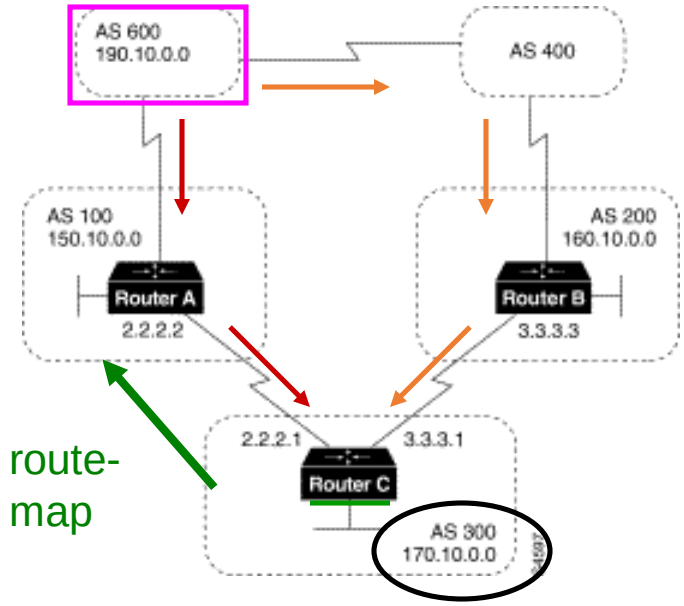
AS_PATH – prepend
Concept



Current "shorter path"

New "shorter path"





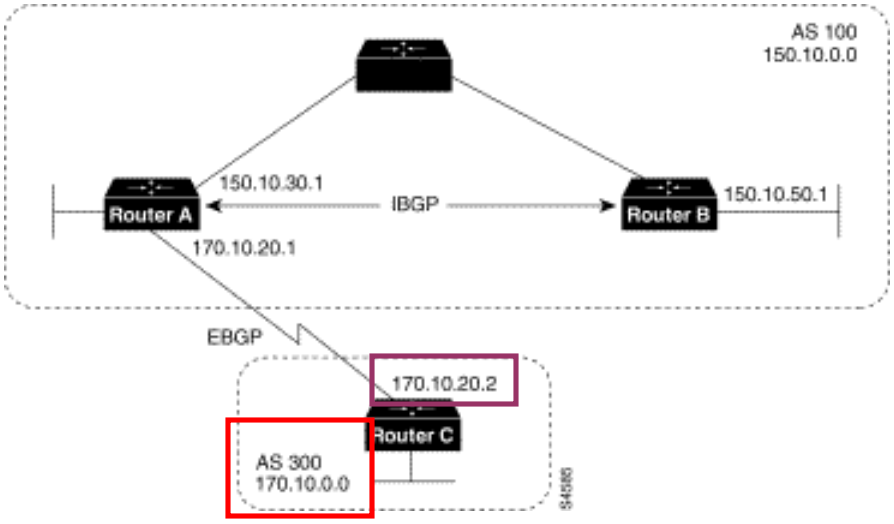
Router C

```
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
```

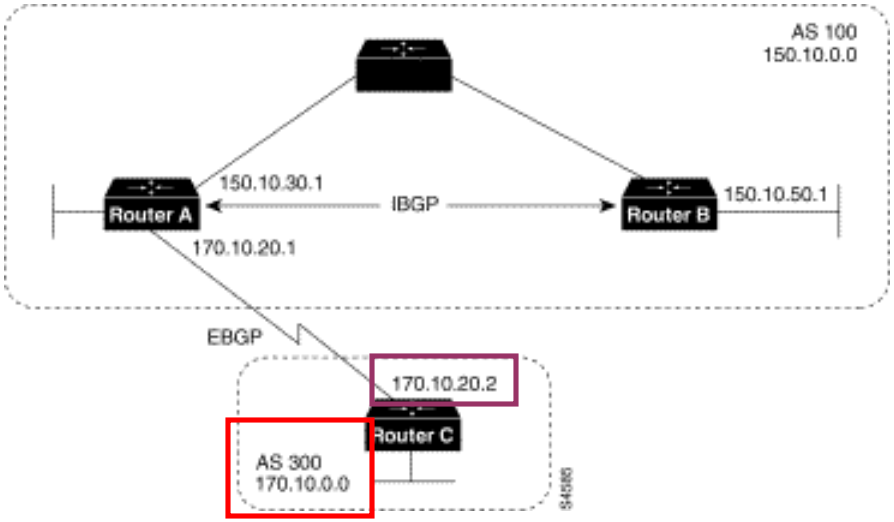
Router B

```
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.



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
```

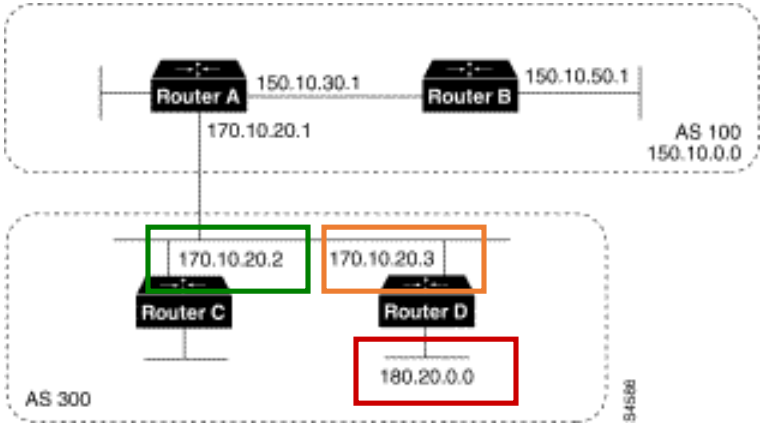
Router C

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

Summarize

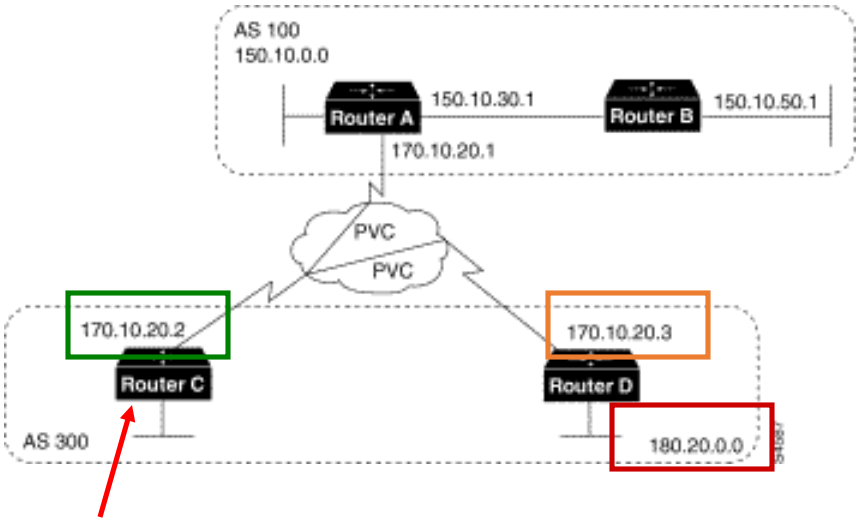
- Router C advertises **170.10.0.0** to Router A with a next hop attribute of **170.10.20.2**,
- Router A advertises **170.10.0.0** to Router B with a next hop attribute of **170.10.20.2**.
- ***The next hop of EBGP-learned routes is passed to the IBGP neighbor without modification into IBGP***.

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.

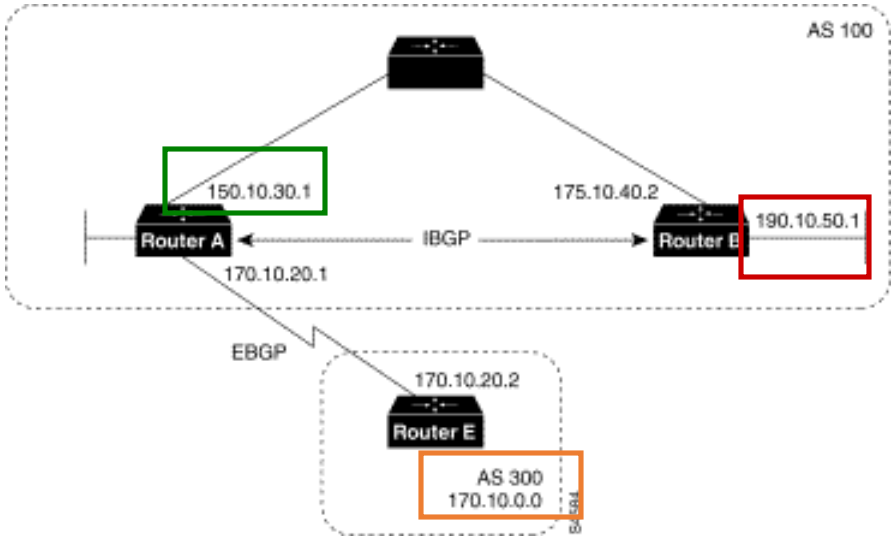
Well-Known Mandatory: ORIGIN

```
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, > b
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
*> 172.16.1.0       0.0.0.0                  0           32768  i
<output omitted>
```

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

```
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>
```

? = 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)

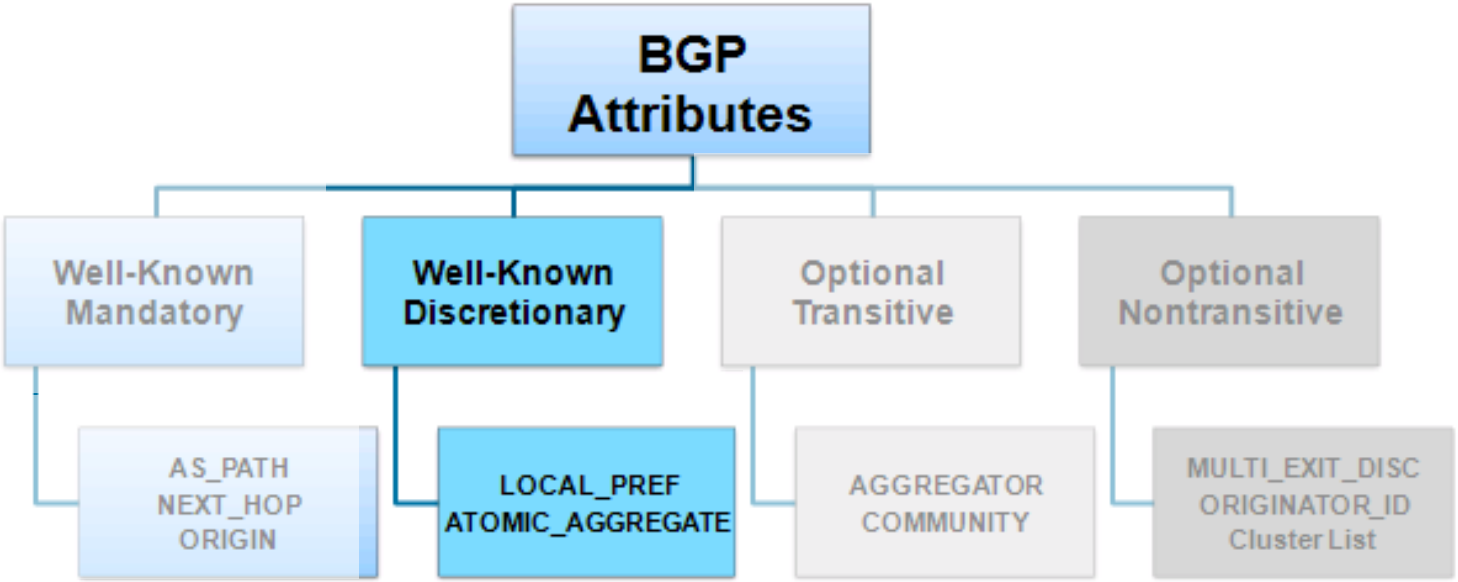
The ORIGIN attribute

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

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

Well-Known Discretionary

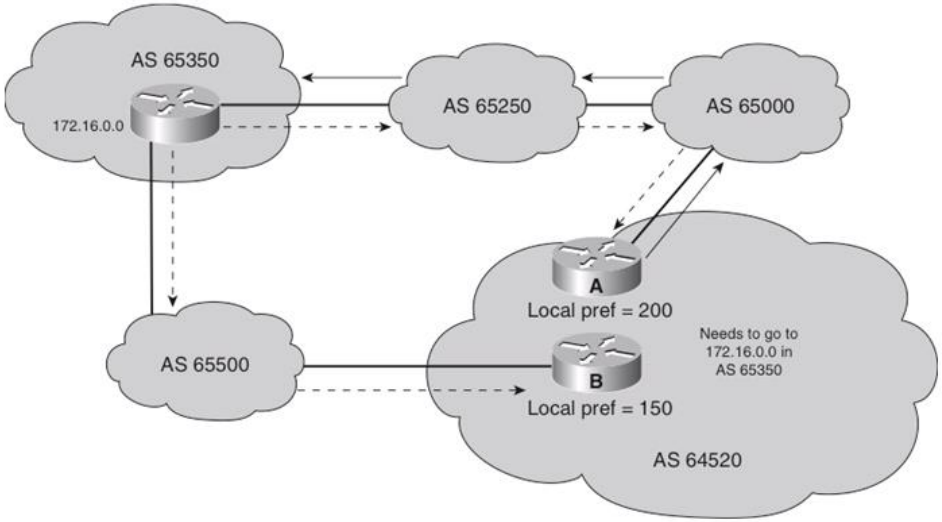
- Attribute is recognized by all implementations of BGP but may not be sent in the BGP update message.



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 IBGP 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.

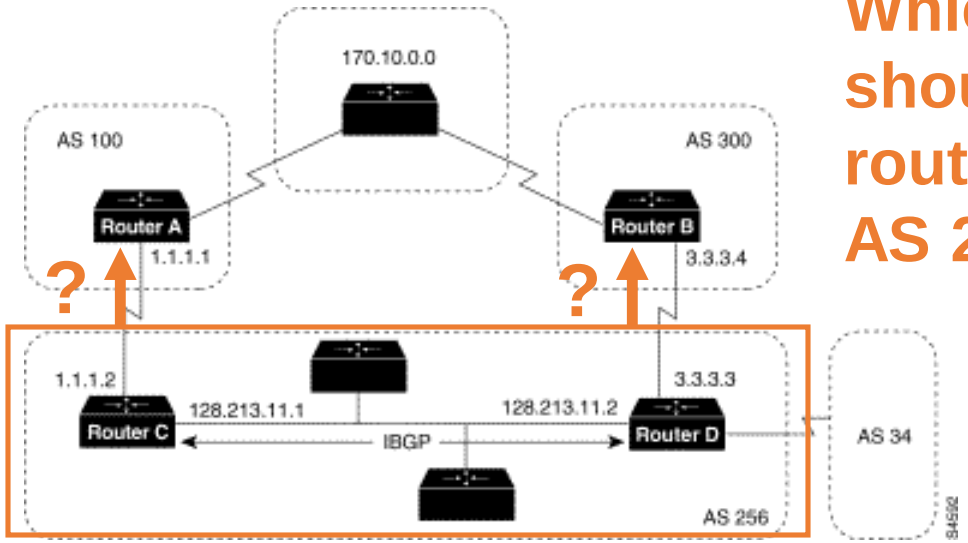
Configuring the Default Local Preference

- 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

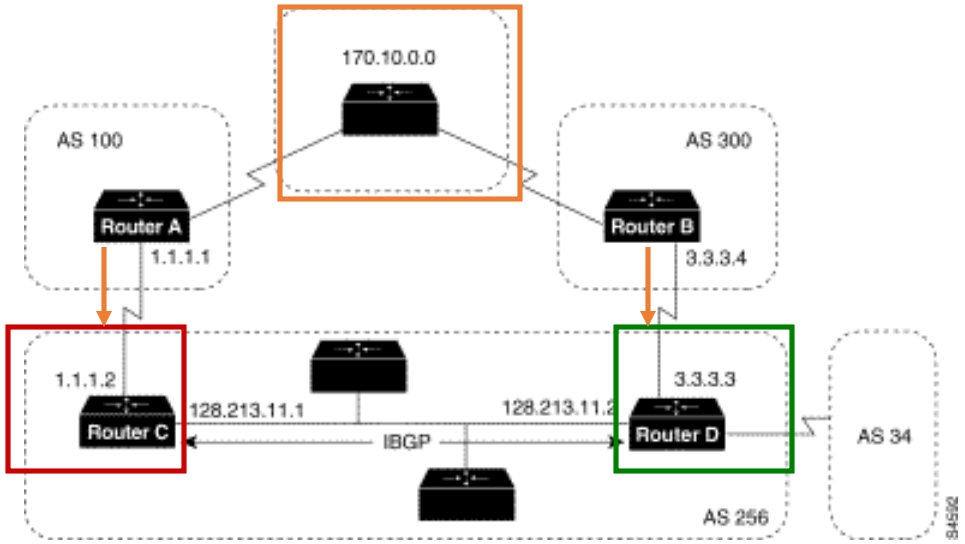
- Well-known discretionary attribute (type code 5).
- Degree of preference given to a route to compare it with other routes for the same destination
  - Higher LOCAL_PREF values are preferred
- **Local to the AS**
  - Exchanged between IBGP peers only
  - It is not advertised to EBGP peers
- **Routers 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

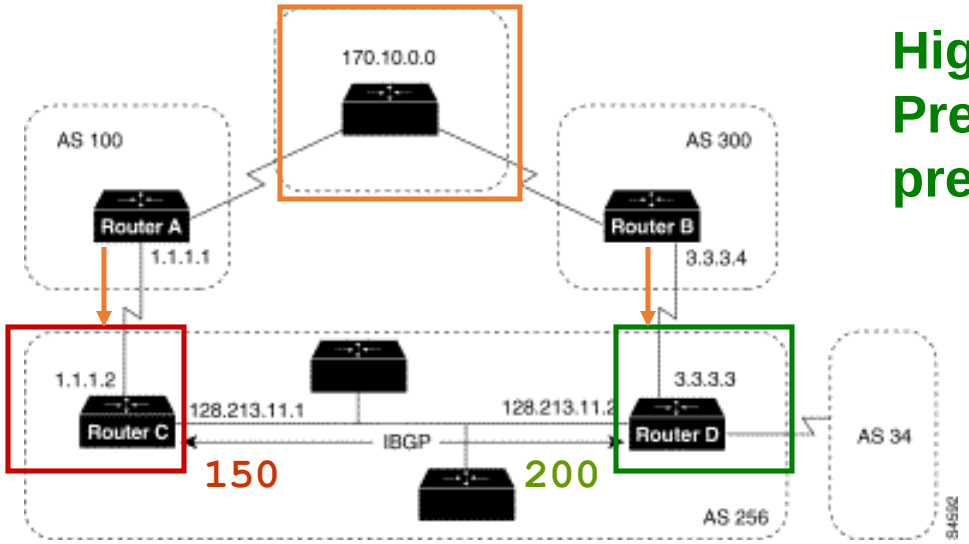


Which exit should all the routers within AS 256 use?

- 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:

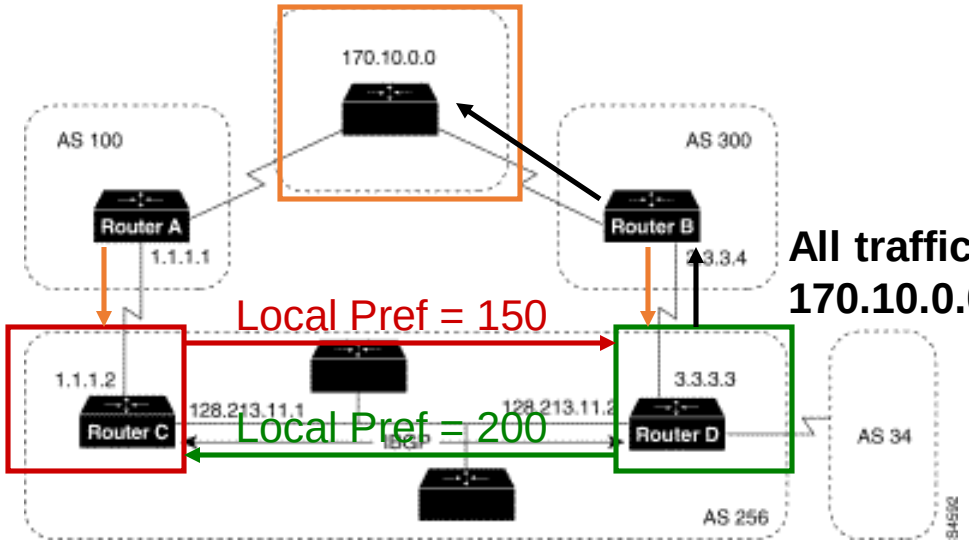
Router C

```
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!



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

Router C

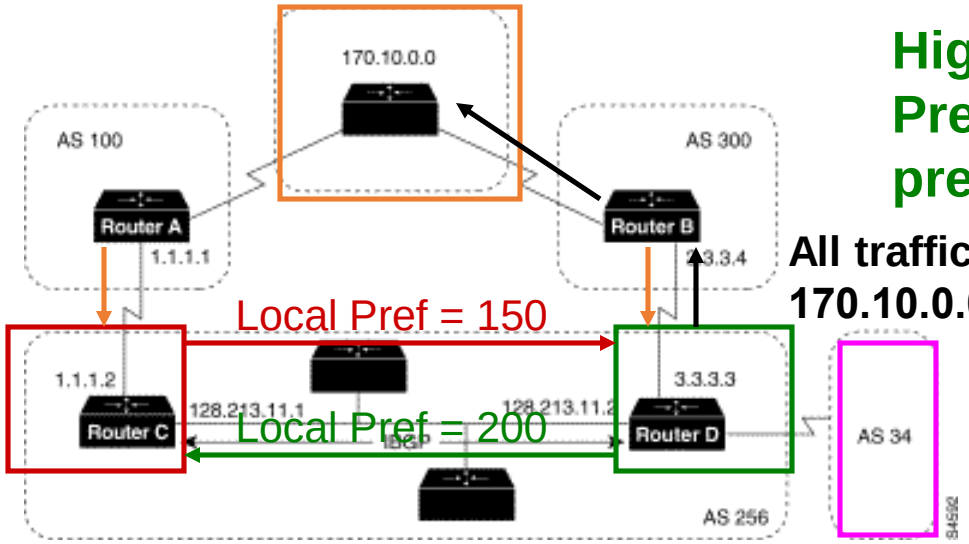
```
router bgp 256
   bgp default local-preference 150
```

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**.

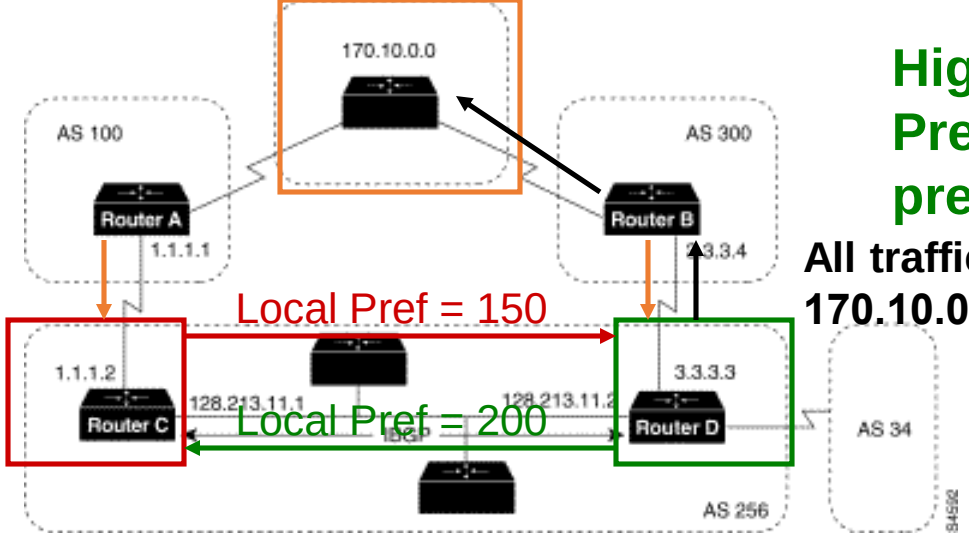


Higher Local Preference is preferred!

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.

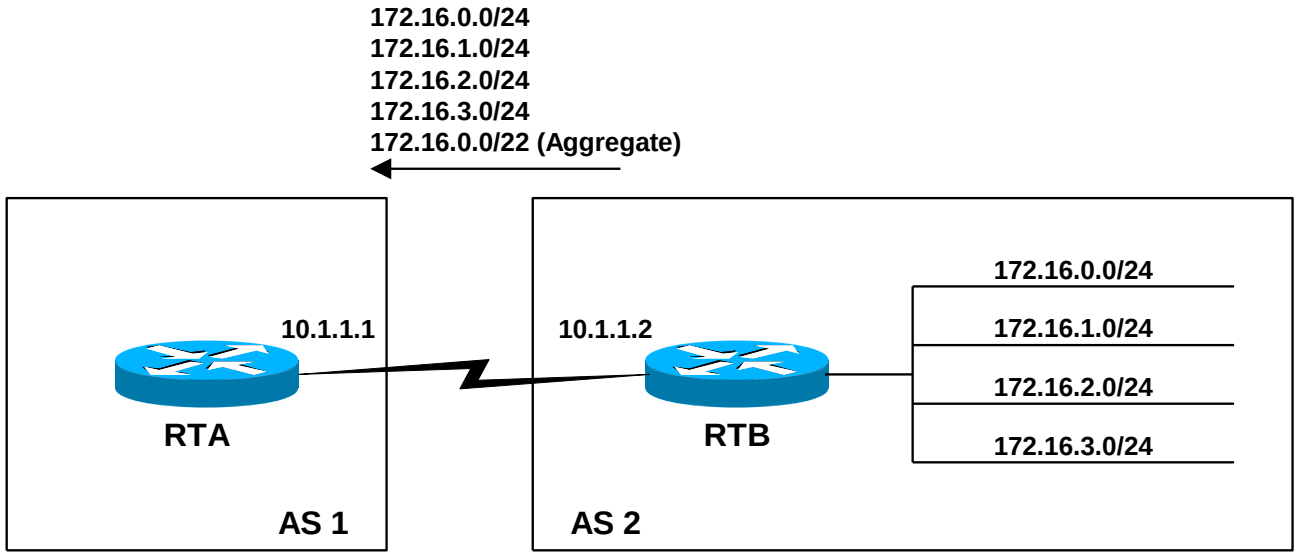
ATOMIC_AGGREGATE

- The **ATOMIC_AGGREGATE** is a well-know 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:

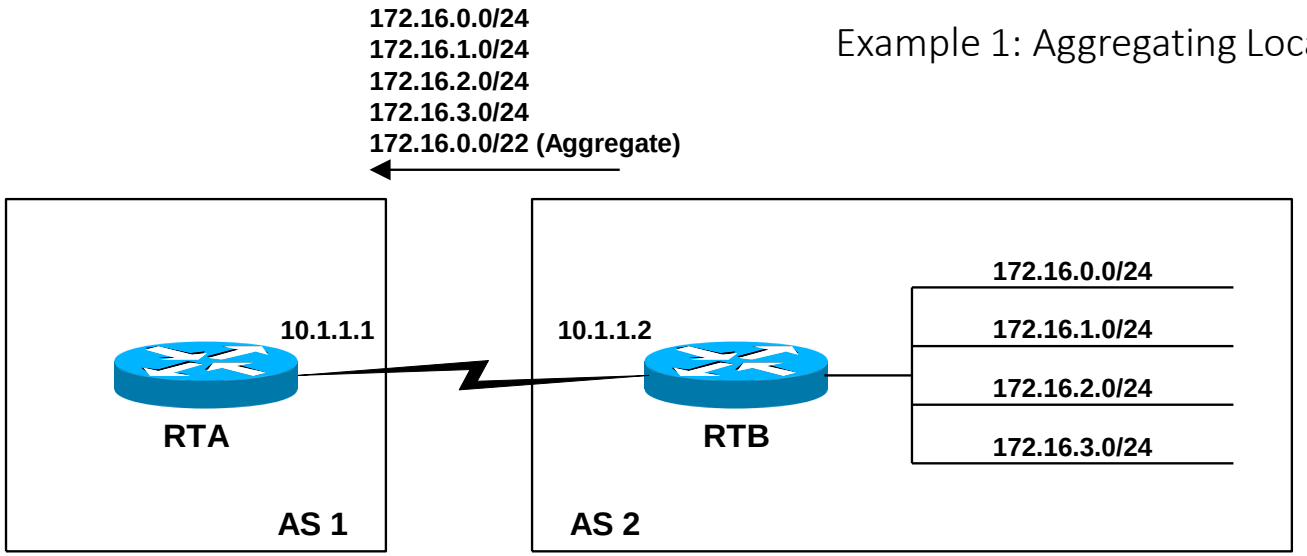
```
Router(config-router)#aggregate-address address mask [as-
  set][summary-only] [suppress-map map-name][advertise-map map-name]
  [attribute-map map-name]
```

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.

Example 1: Aggregating Local Routes



RTA

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

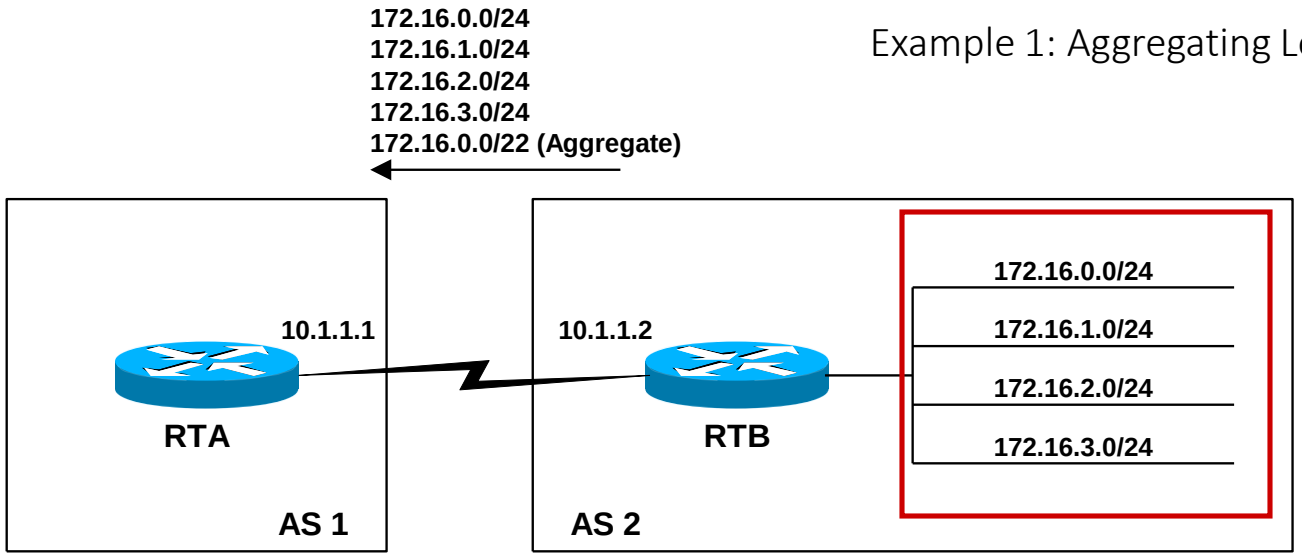
RTB

```
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

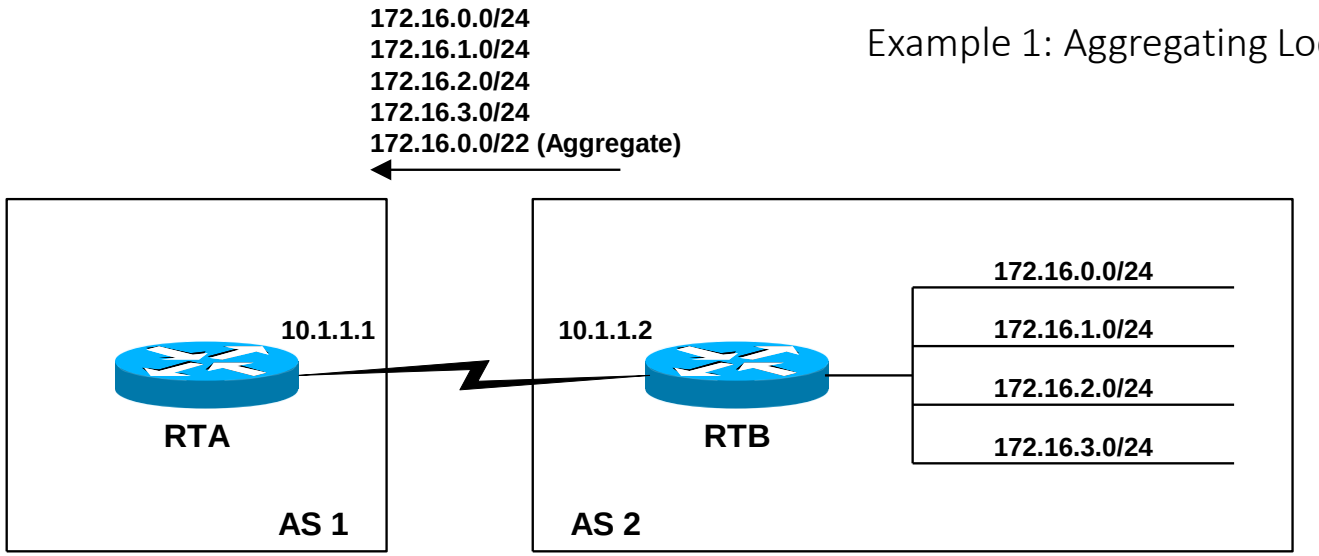


RTB

```
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

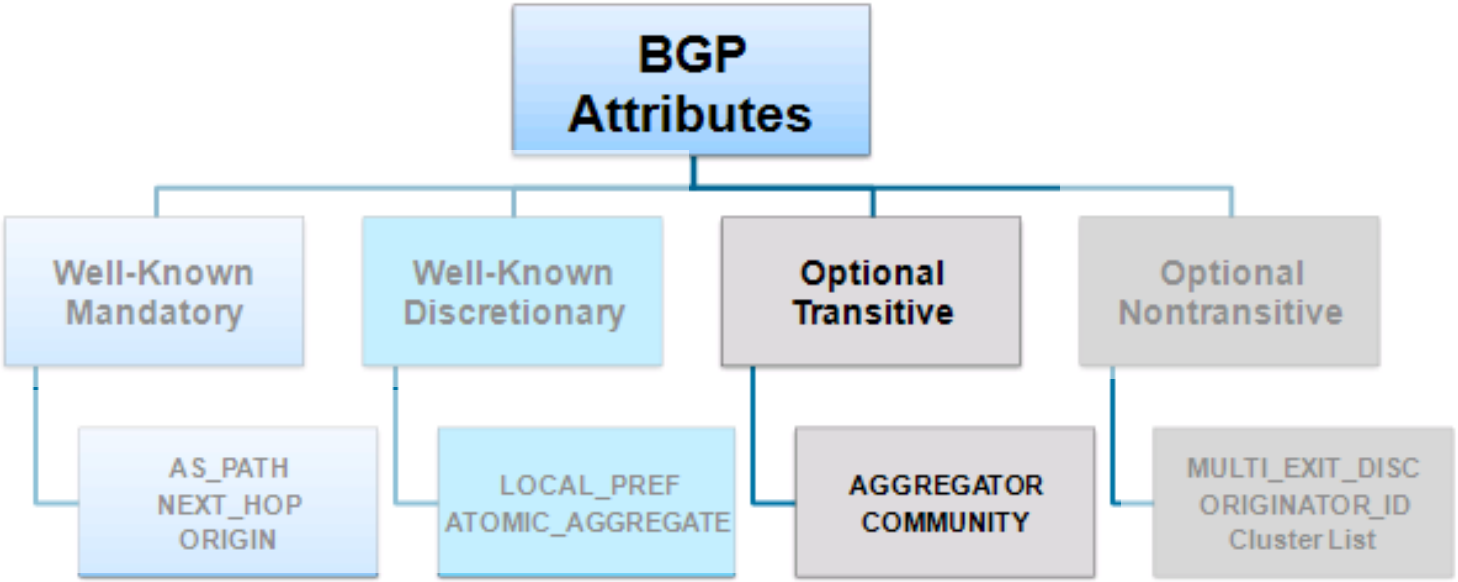


show ip bpg 172.16.0.0 will display that this route has the "atomic-aggregate" attribute set.

```
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.

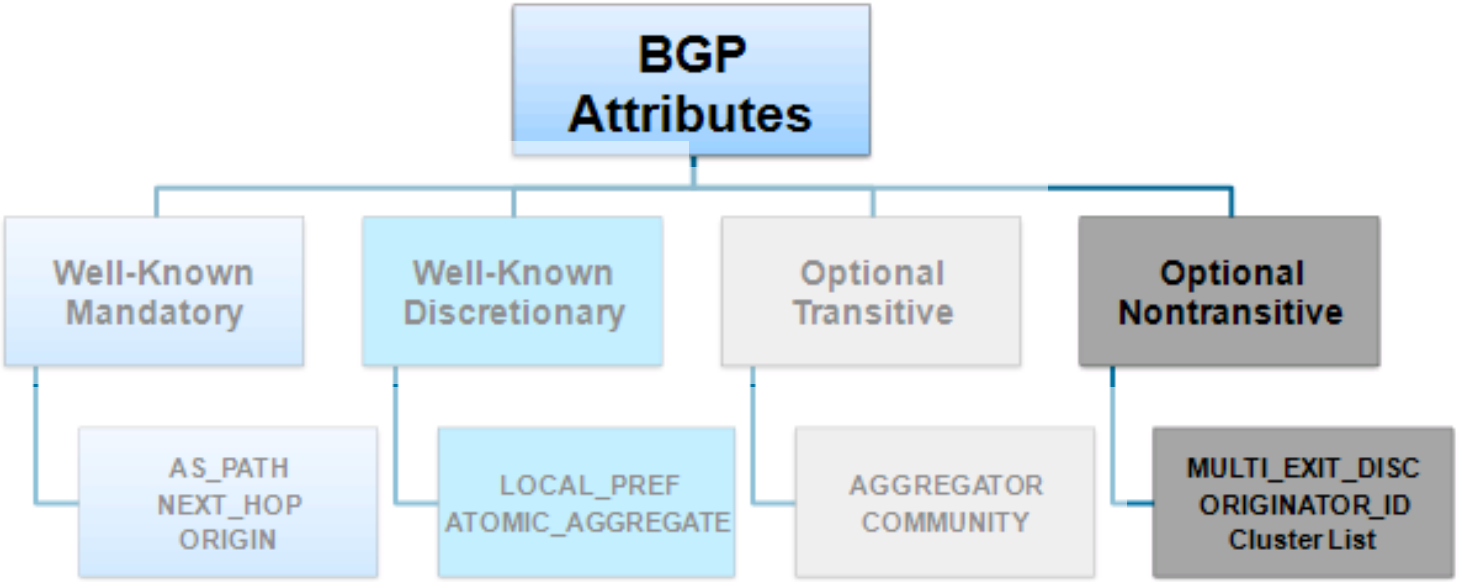


Optional Transitive: Community

- 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

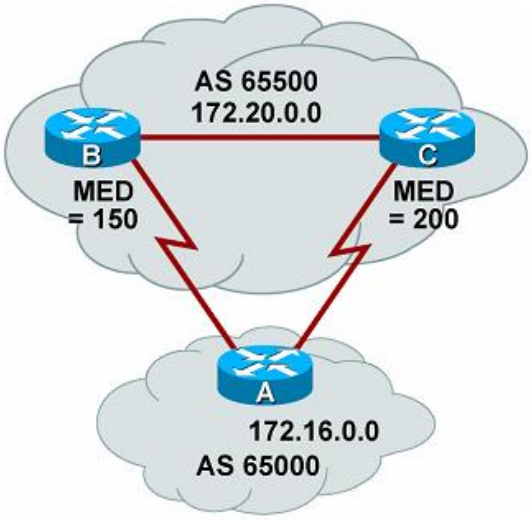
- Attribute that may or may not be recognized by all BGP implementations.
- Whether or not the receiving BGP router recognizes the attribute, it is nontransitive and is not passed along to other BGP peers.



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

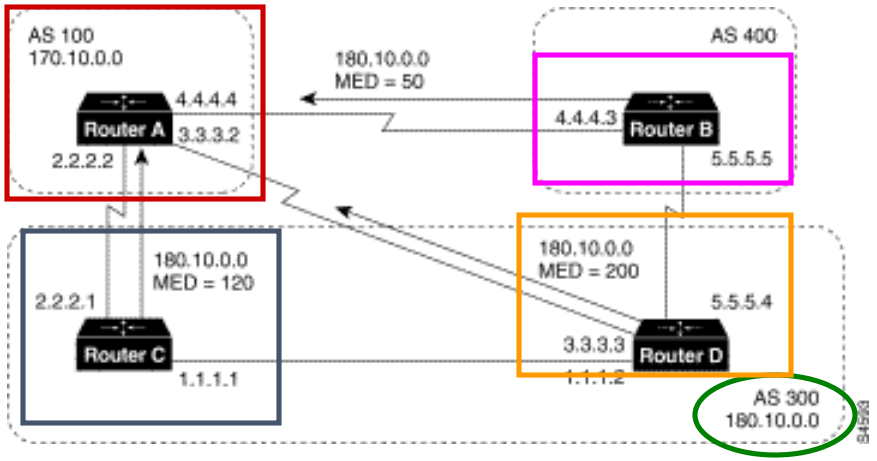
- The MULTI_EXIT_DISC (Multi-Exit Discriminator) attribute is an optional non-transitive attribute (type code 4).
- **Informs external neighbors about the preferred path** into an AS that has multiple entry points.
- A lower MULTI_EXIT_DISC (or MED) is preferred over a higher MED.

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
```

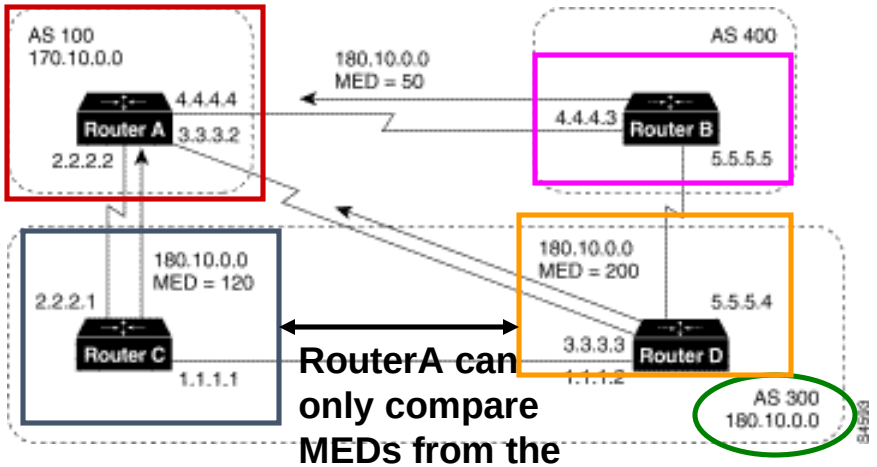
Router C

```
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
```

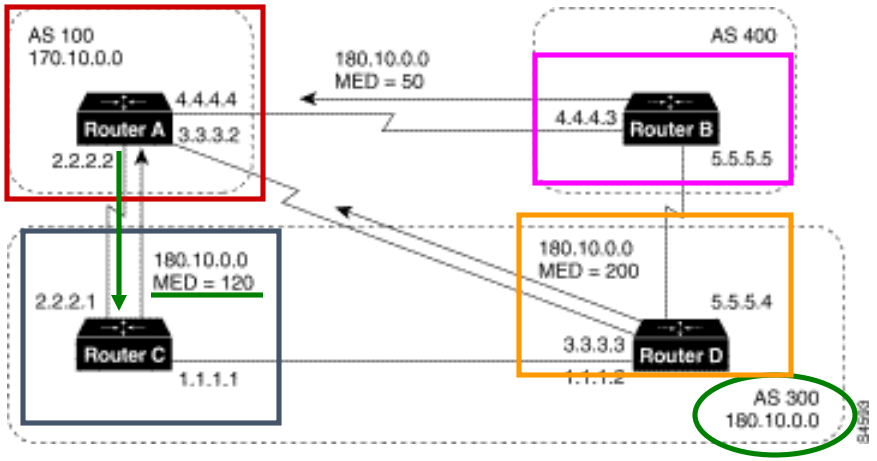
Router C

```
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
```

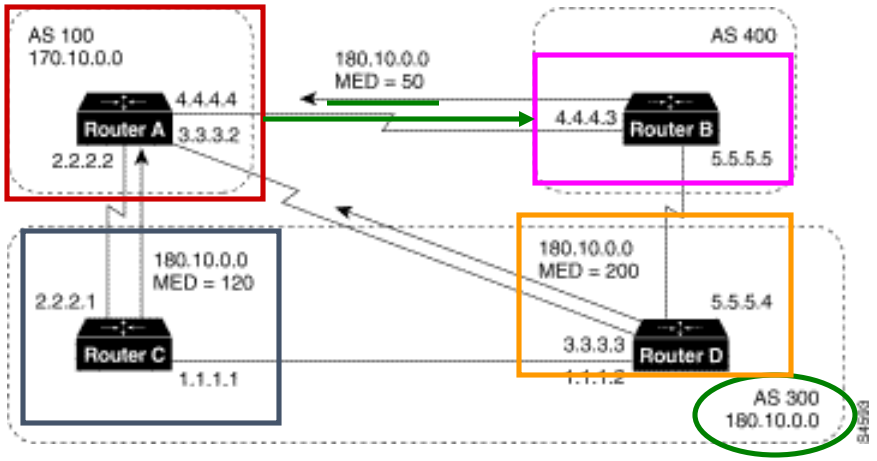
Router C

```
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
```

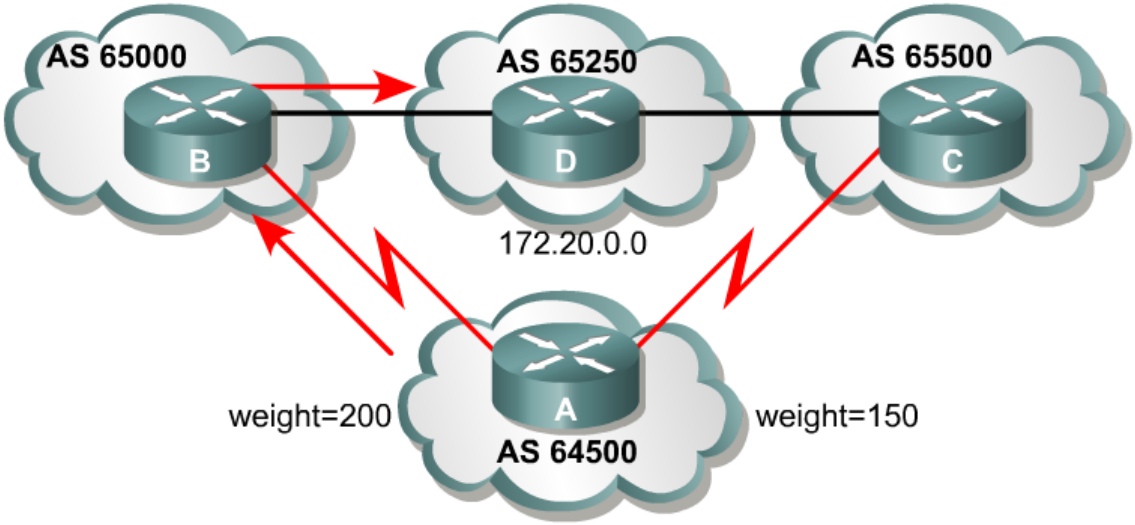
Router D

```
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

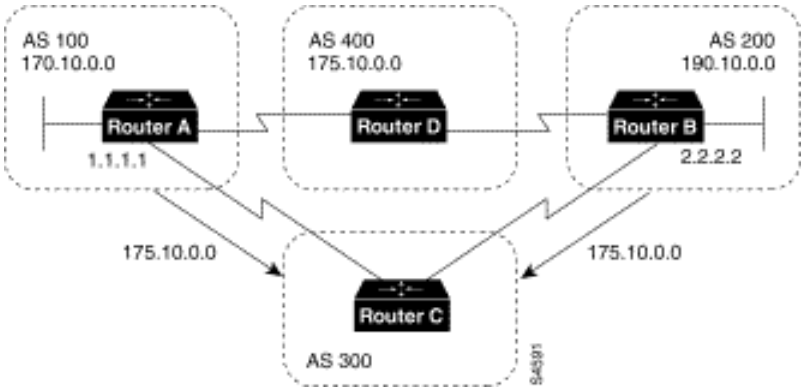
- The Weight attribute is a Cisco proprietary attribute.
- Similar in function to the local preference, the weight attribute applies when 1 router has multiple exit points.
  - Local preference is used when 2+ routers provide multiple exit points.
- It is configured locally on a router and is not propagated 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.

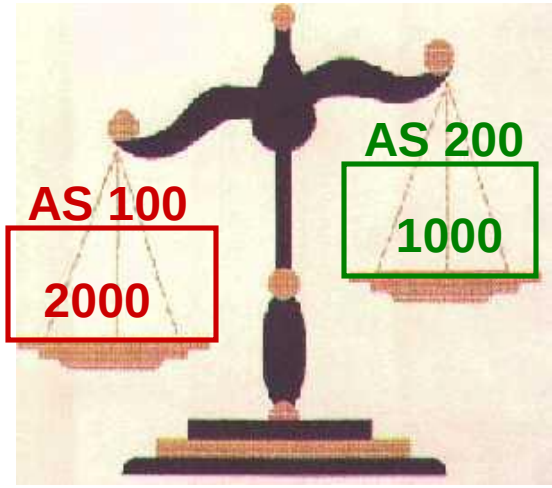
Cisco Weight Attribute



- 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.

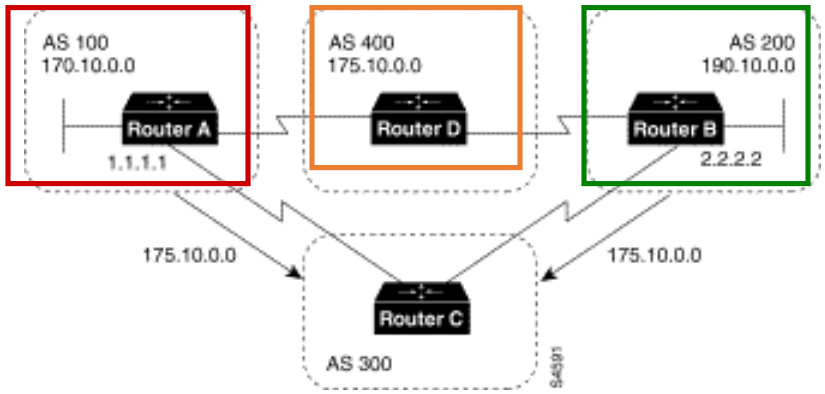
The WEIGHT attribute

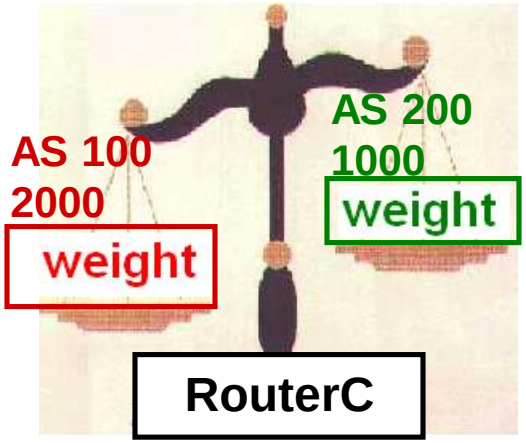




- The weight attribute is a **special Cisco attribute** that is used in the path selection process **when there is more than one route to the same destination**.
- The weight attribute is **local to the router on which it is assigned**, and it is **not propagated** in routing updates.
- By **default**, 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.

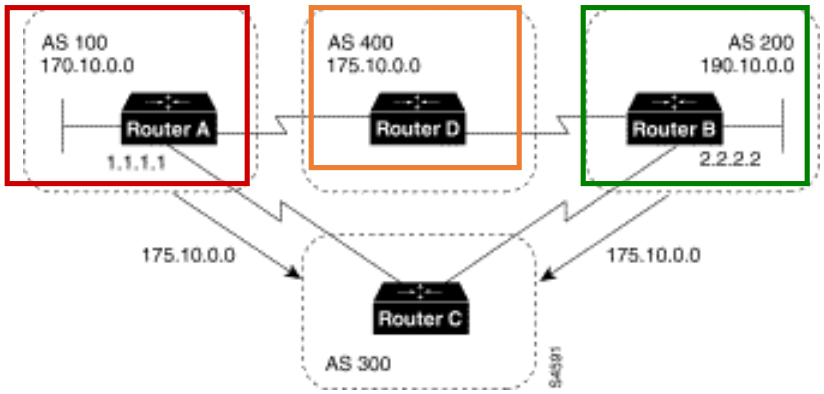
The WEIGHT attribute

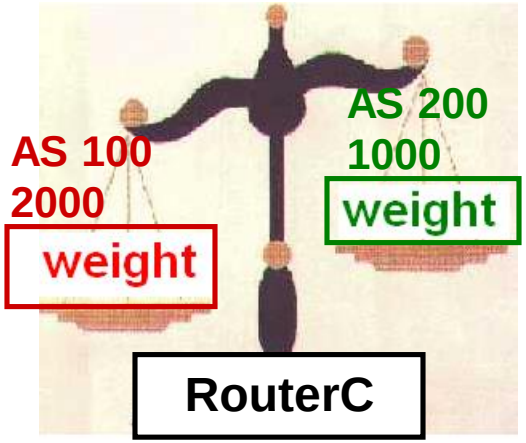




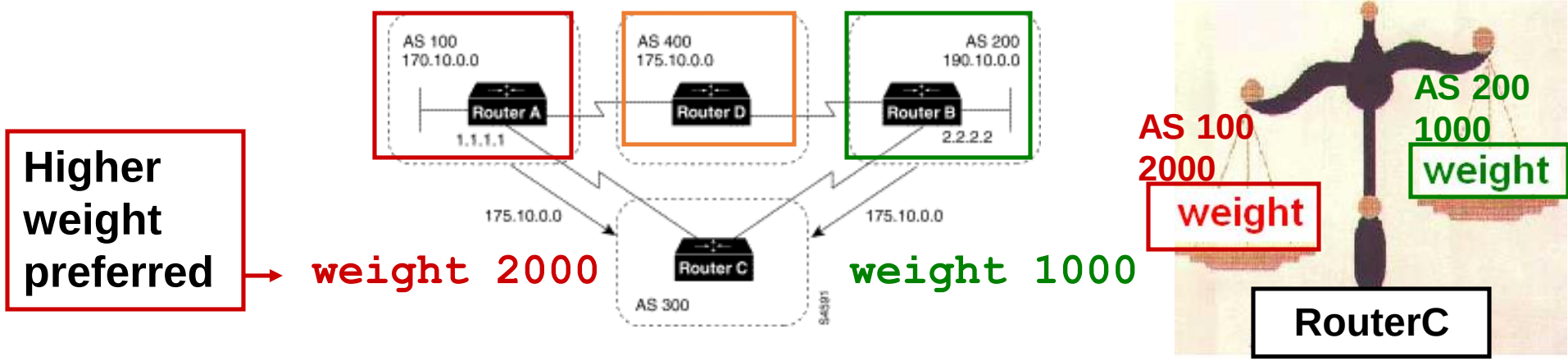
- **Router A** and **Router B** learn about network **175.10.0.0 from AS 400**, and each propagates the update to **Router C**.
- **Router C** has two routes for reaching **175.10.0.0** and has to decide which route to use.
- If, on **Router C**, you set the weight of the updates coming in from **Router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



Using the neighbor weight Command to Set the Weight Attribute

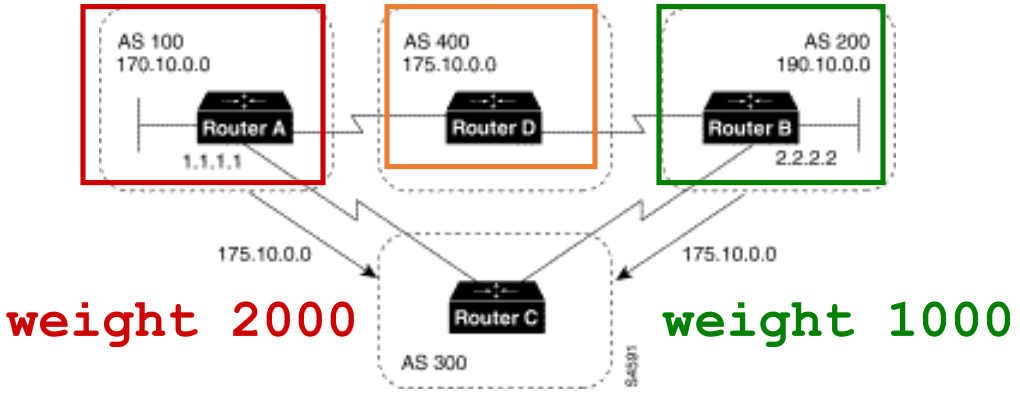
- The following configuration for Router C uses the **`neighbor weight`** router configuration command:

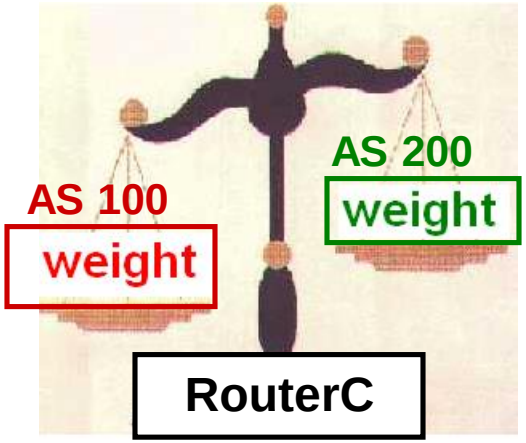
`Router C`

```
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

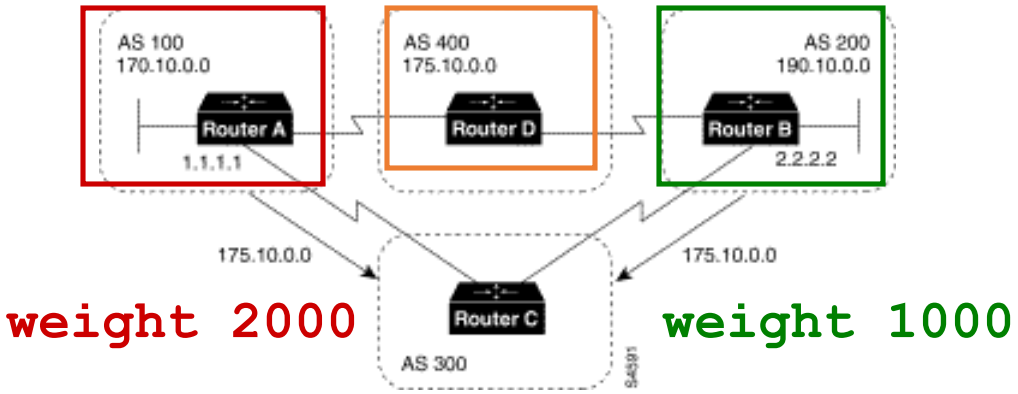
```
router bgp 300
    neighbor 1.1.1.1 remote-as 100
    neighbor 1.1.1.1 filter-list 5 weight 2000
    neighbor 2.2.2.2 remote-as 200
    neighbor 2.2.2.2 filter-list 6 weight 1000
ip as-path access-list 5 permit ^100$
ip as-path access-list 6 permit ^200$
```

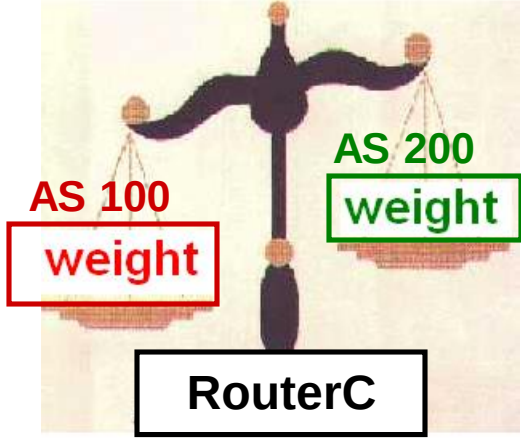
The WEIGHT attribute – Access list

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.

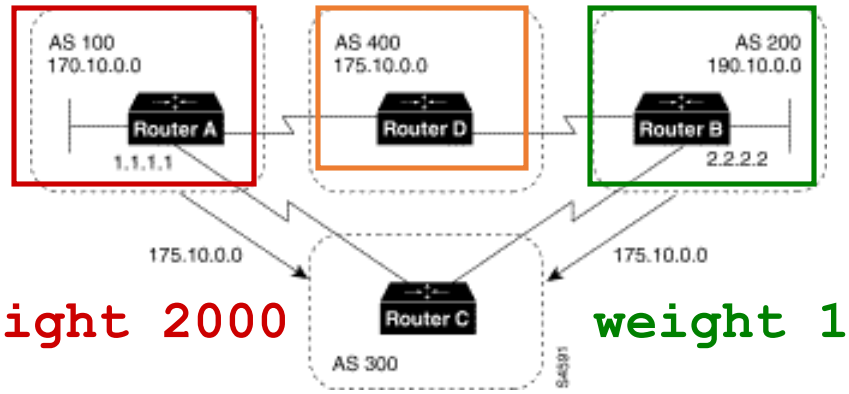


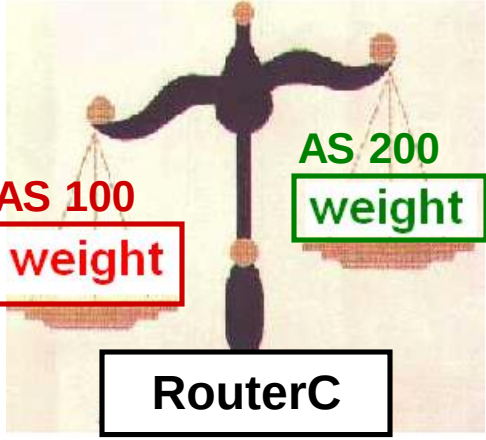


Using a Route Map to Set the Weight Attribute - FYI

- 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
```





weight 2000

weight 1000

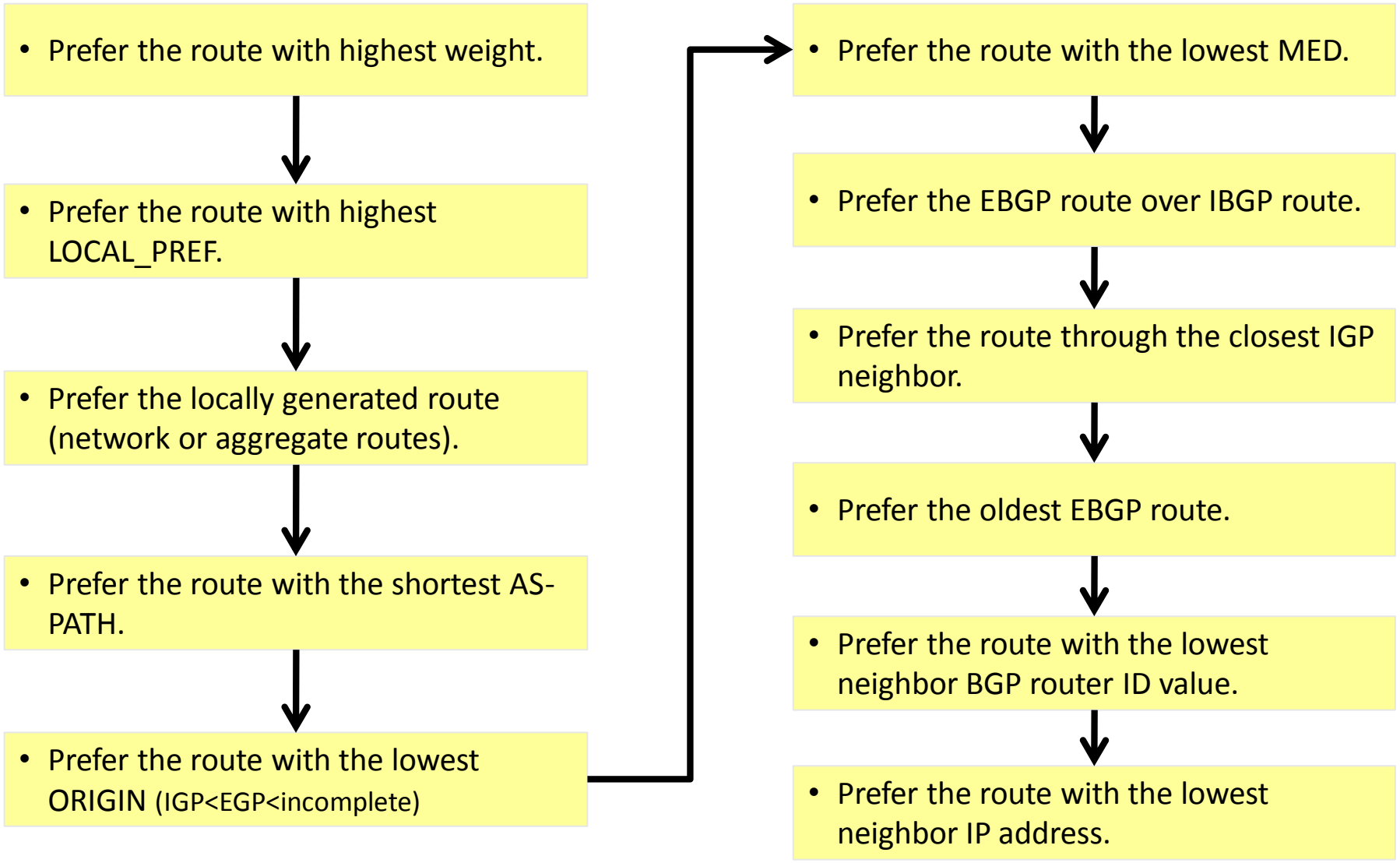
```
Router C
router bgp 300
    neighbor 1.1.1.1 remote-as 100
    neighbor 1.1.1.1 route-map SETWEIGHTIN in
    neighbor 2.2.2.2 remote-as 200
    neighbor 2.2.2.2 route-map SETWEIGHTIN in
ip as-path access-list 5 permit ^100$
route-map SETWEIGHTIN permit 10
    match as-path 5
    set weight 2000
route-map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



http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html

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

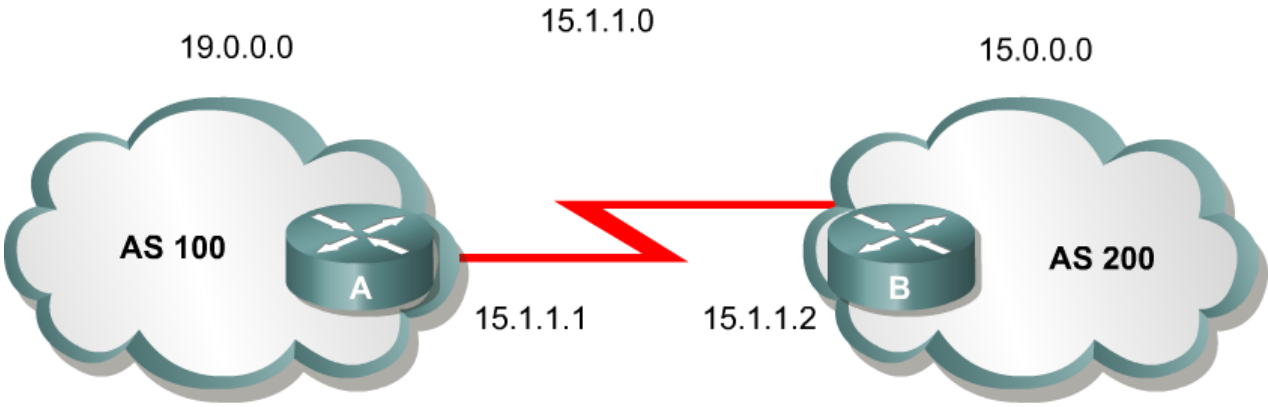
Verifying BGP

- 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

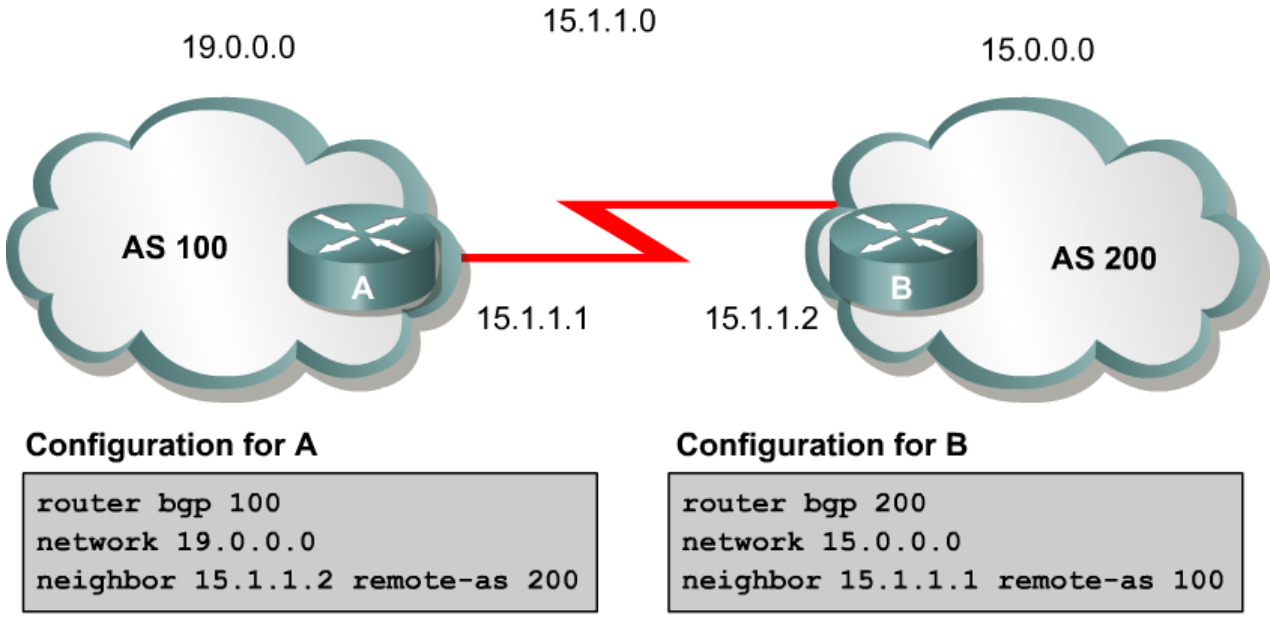
```
router bgp 200
network 15.0.0.0
neighbor 15.1.1.1 remote-as 100
```

- 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.**

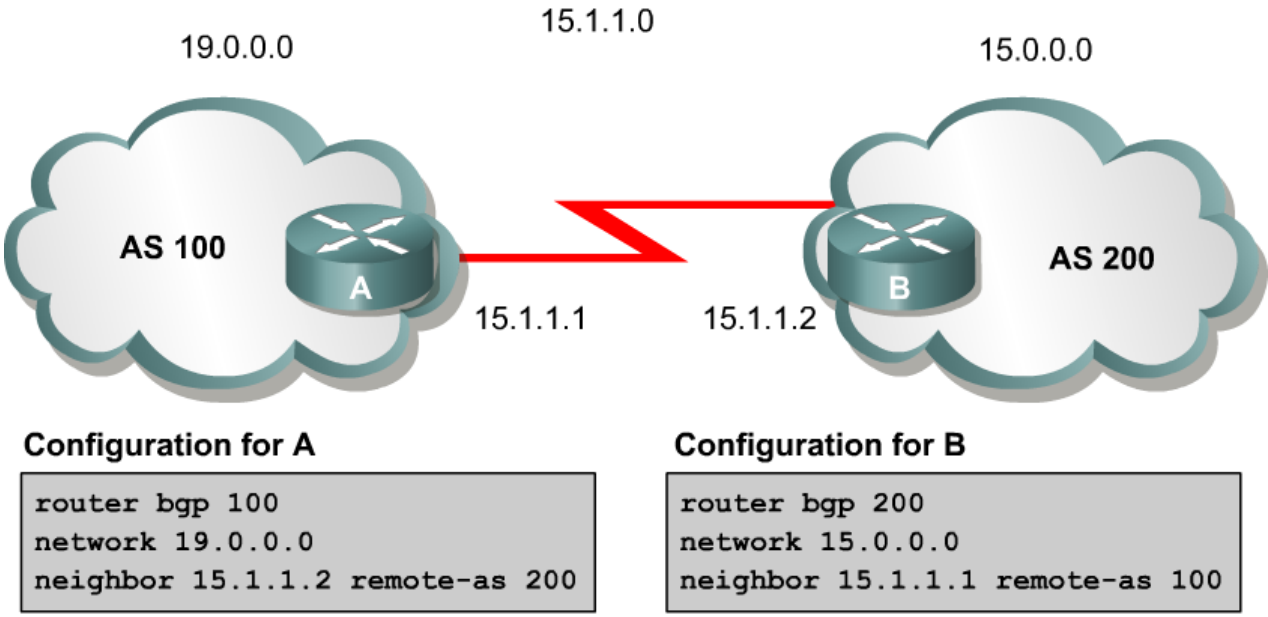
BGP Configuration



`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



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.

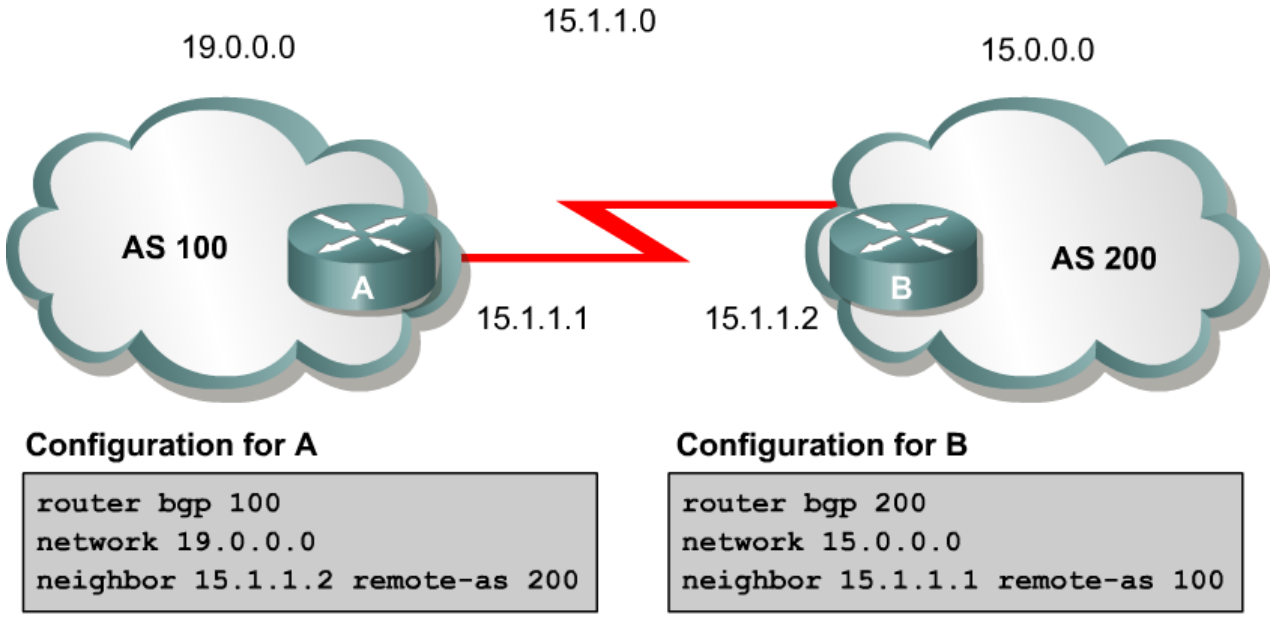
BGP Route Must Be in IP Routing Table

- 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.

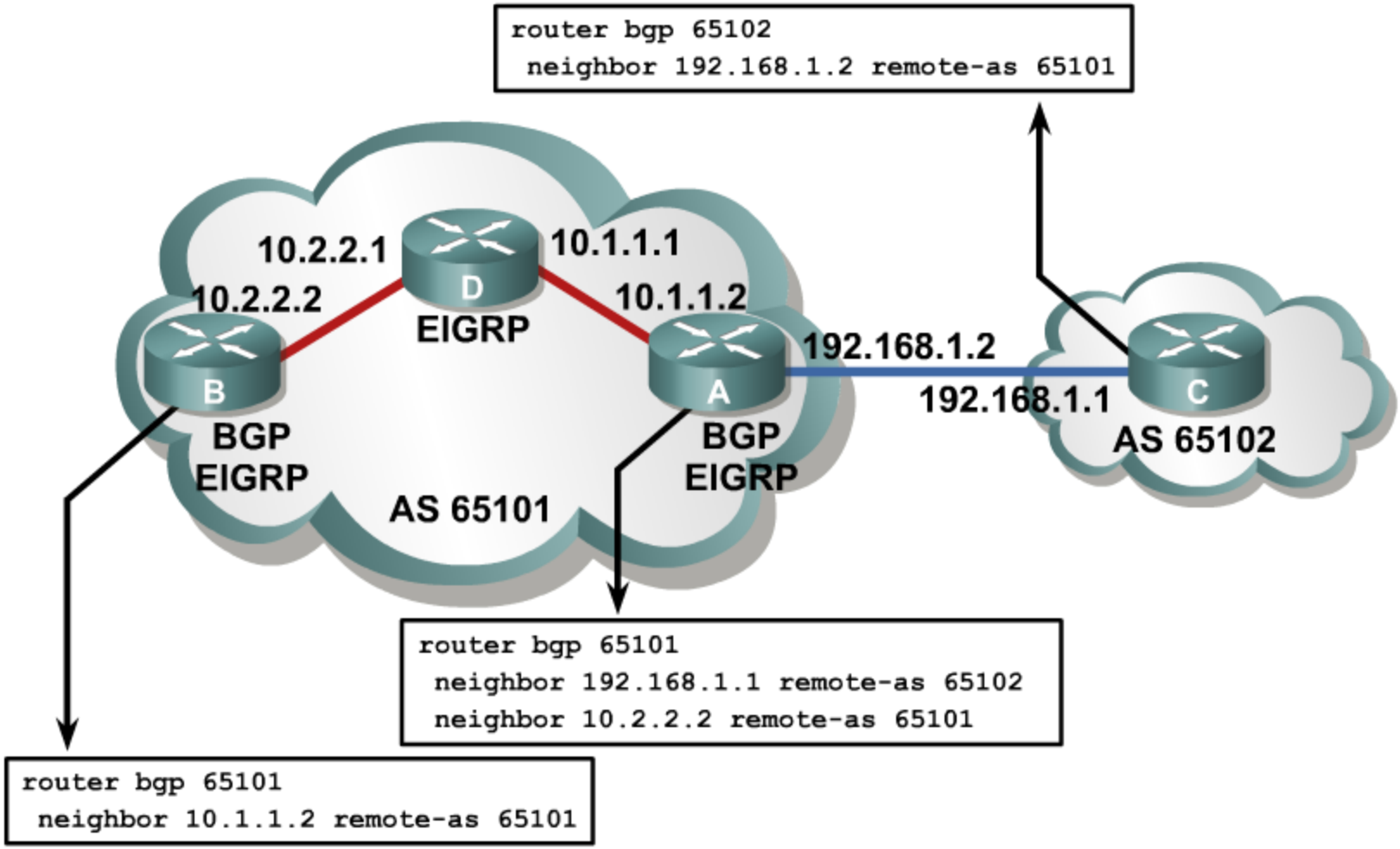
BGP Configuration



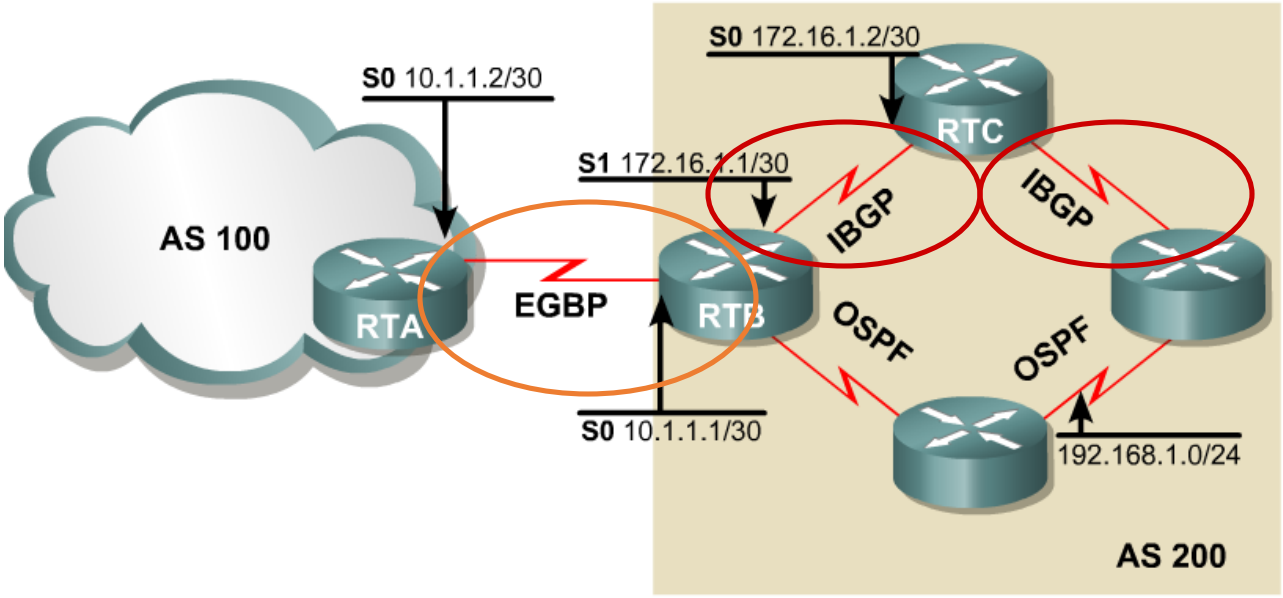
`Router(config-router)#`**`neighbor ip-address remote-as AS-number`**

- 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.

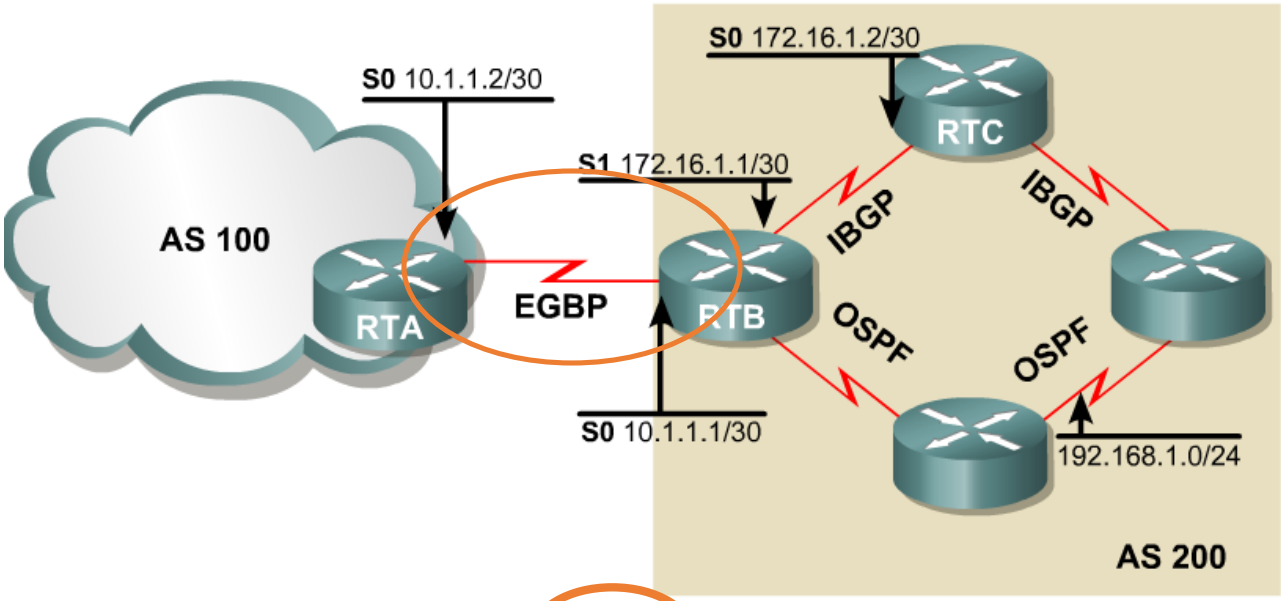
BGP Configuration



BGP Configuration



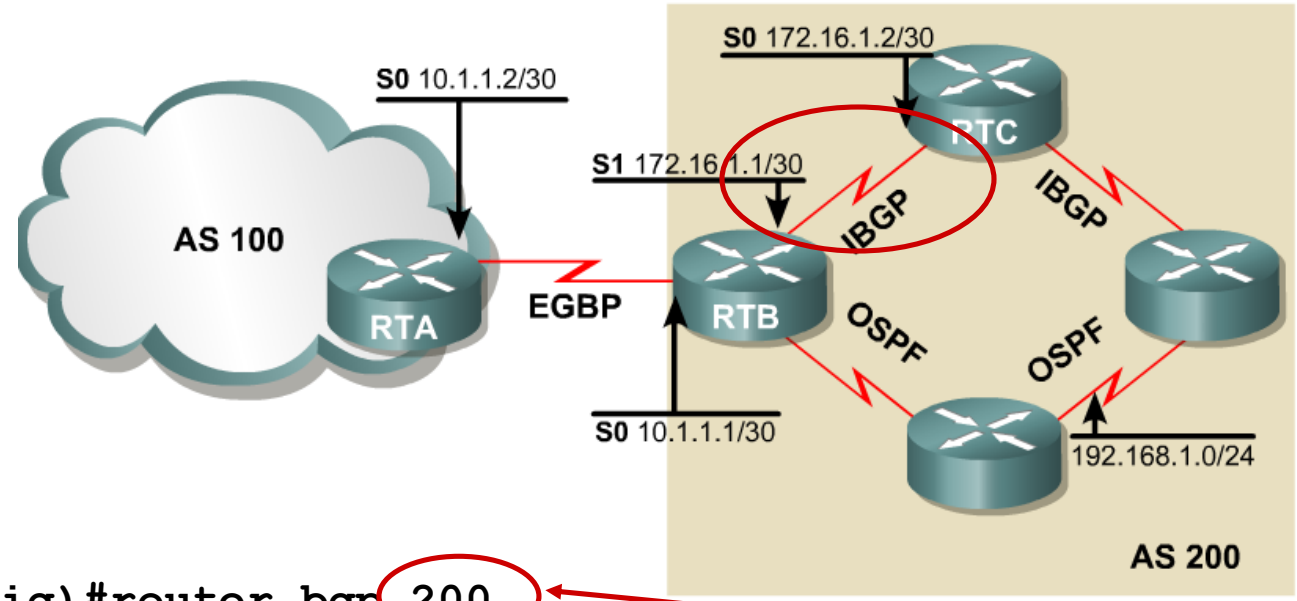
- If the ***AS-number*** configured in the **router bgp** command is **identical** to the ***AS-number*** configured in the **neighbor** statement, BGP will initiate an **internal session - IBGP**.
- If the field values are **different**, BGP will build an **external session - EBGP**.



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
```

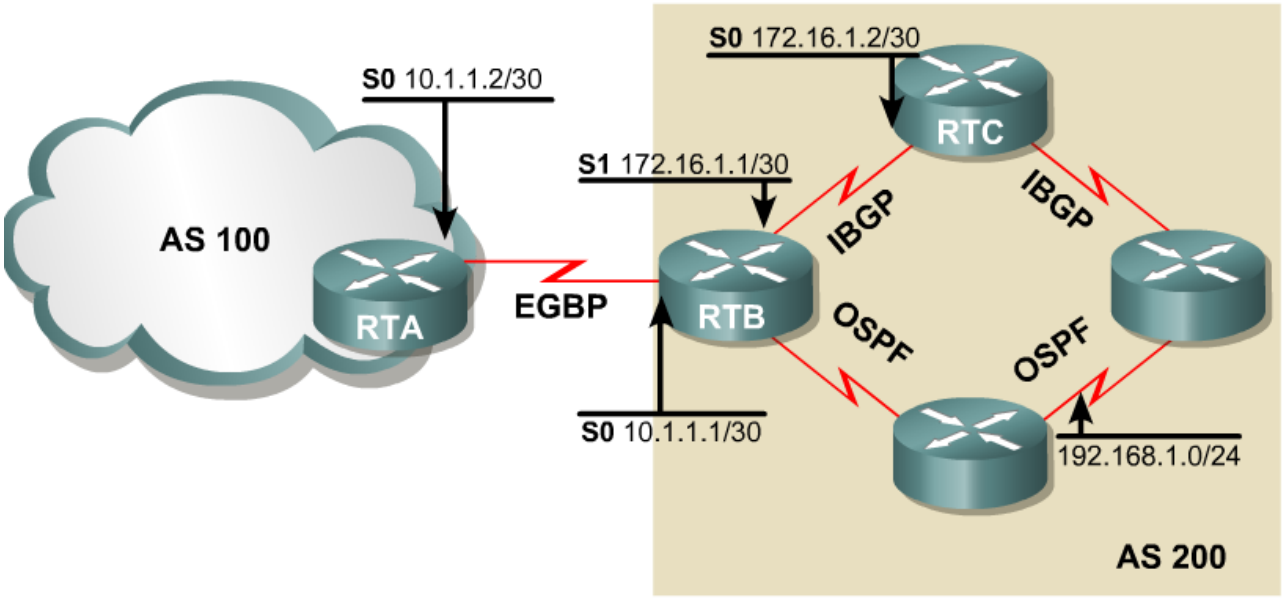
- RTB: Note that the **neighbor** command's **remote-as** value, 100, is different from the AS number specified by the **router bgp** command (200).
- Because the two AS numbers are different, BGP will start an **EBGP** connection with RTA.
- Communication will occur between autonomous systems.



IBGP

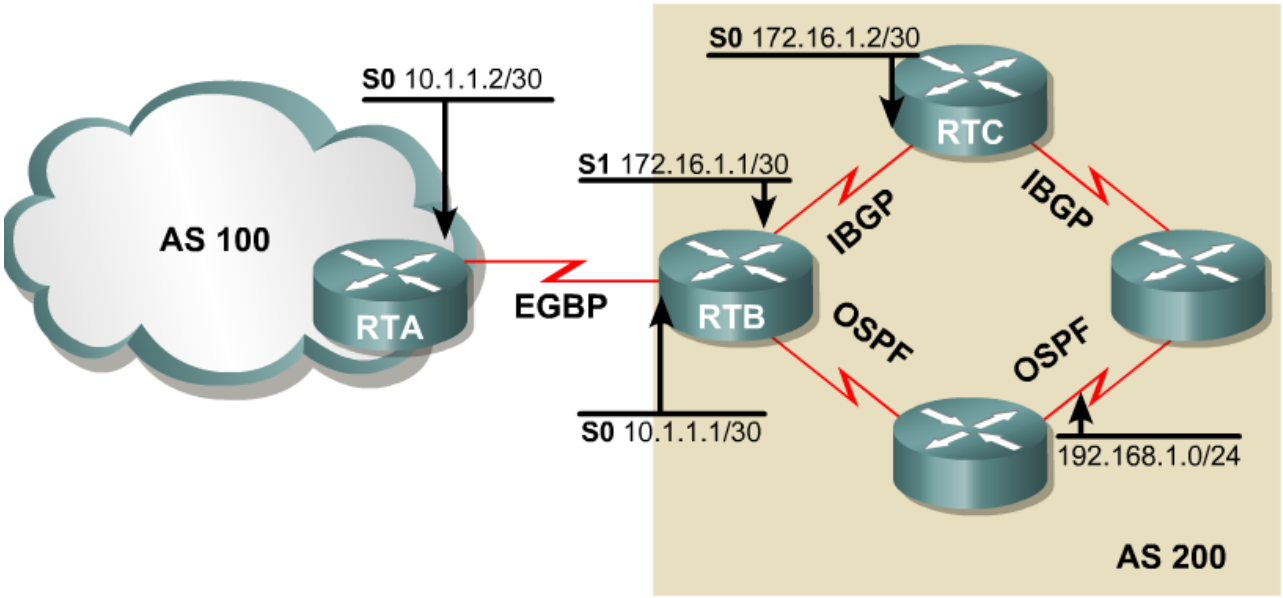
```
RTB(config)#router bgp 200
RTB(config-router)#neighbor 172.16.1.2 remote-as 200
RTB(config-router)#neighbor 172.16.1.2 update-source loopback 0

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
```

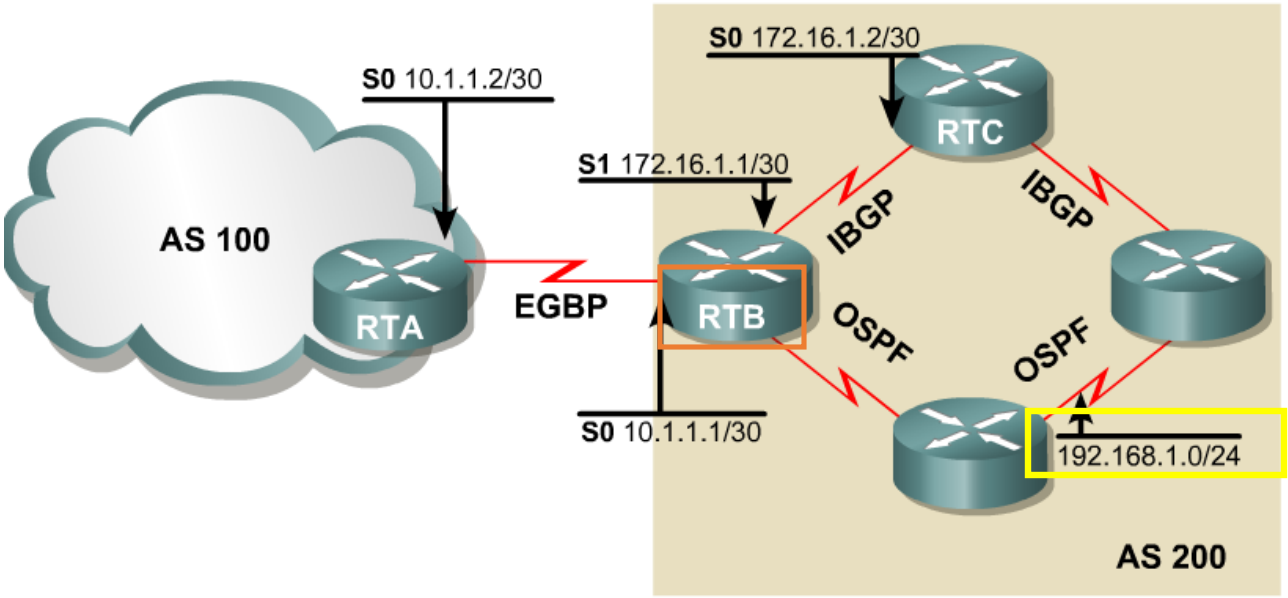
- The **update-source loopback 0** command is used to instruct the router to use *any* operational interface for TCP connections (as long as Lo0 is up and configured with an IP address).
- Without the **update-source loopback 0** command, BGP routers can use only the closest IP interface to the peer.
- The ability to use any operational interface provides BGP with robustness in the event the link to the closet 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.



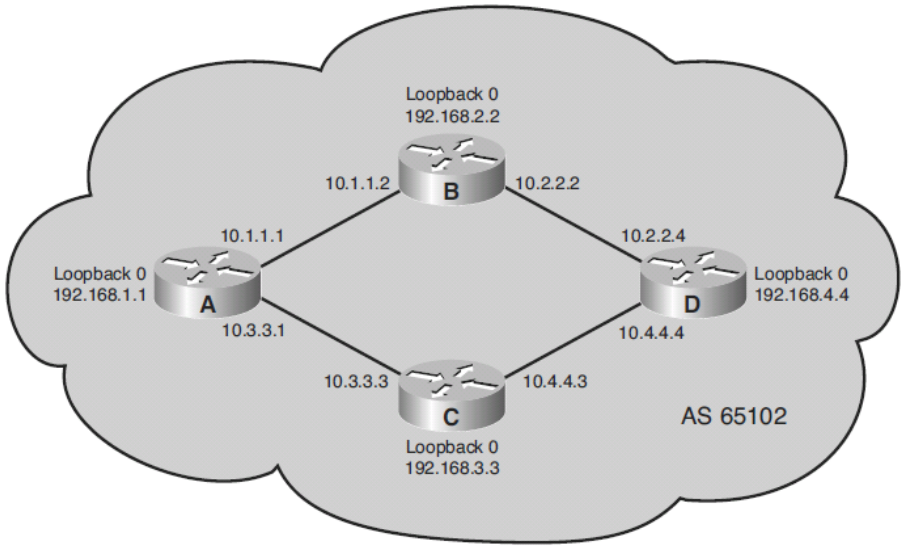

```
RTB(config)#router bgp 200
RTB(config-router)#network 172.16.1.0 mask 255.255.255.254
RTB(config-router)#network 10.1.1.0 mask 255.255.255.254
RTB(config-router)#network 192.168.1.0
```

- The first two **network** commands in 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

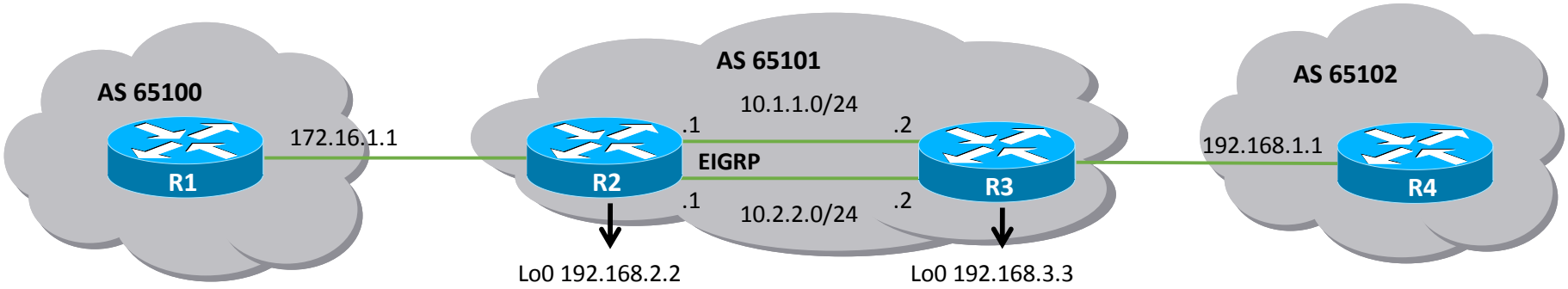
- Establish the IBGP session using a loopback interface.

```
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-multihop`

- 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

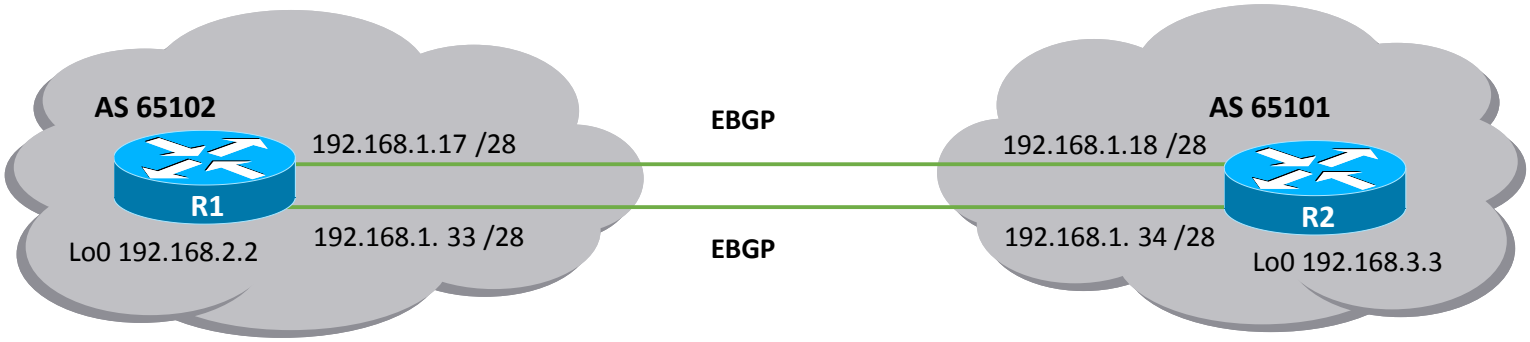
- Increase the time-to-live (TTL) for EBGP connections.

```
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.

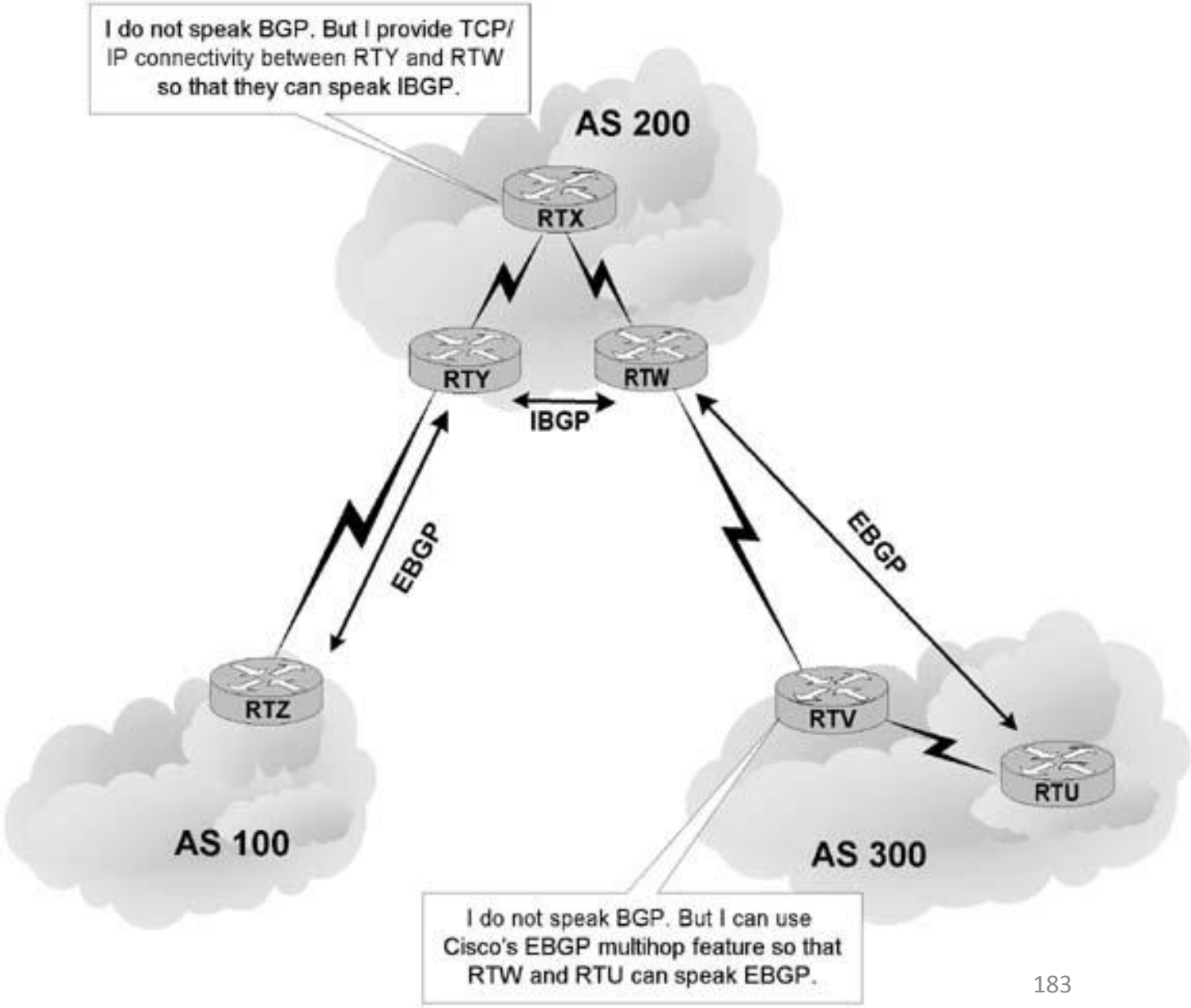
Multihop EBGP Example



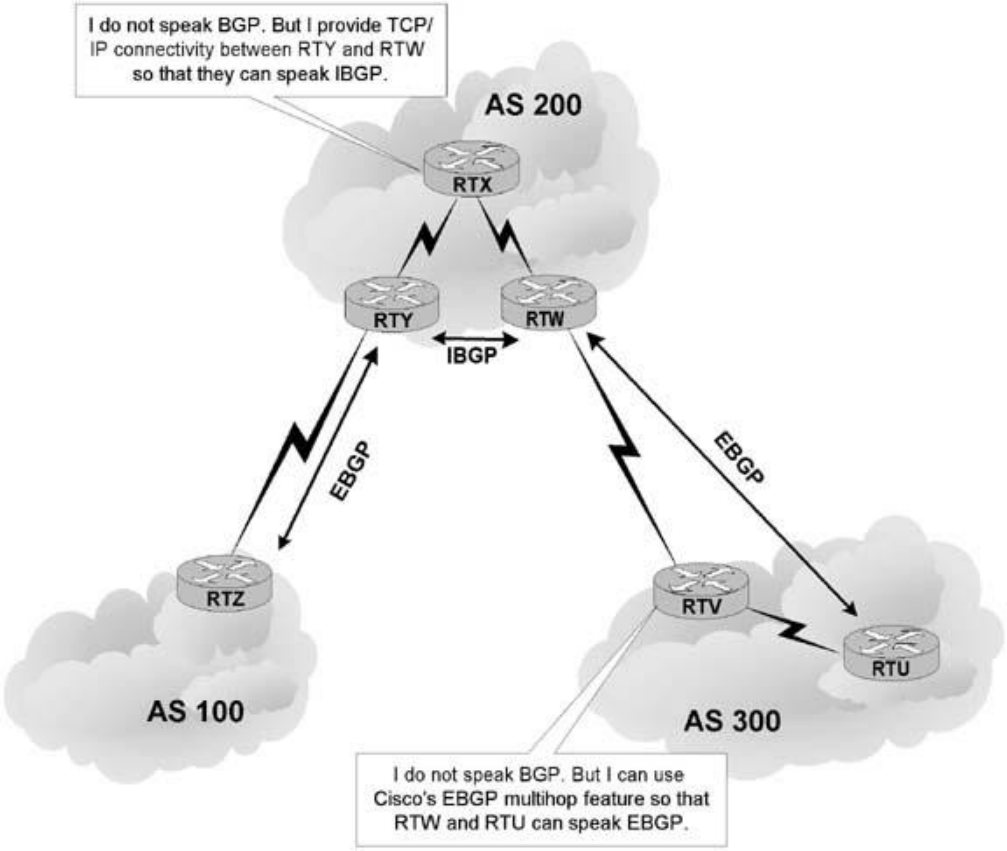

```
R1(config)# router bgp 65102
R1(config-router)# neighbor 172.16.1.1 remote-as 65101
R1(config-router)# neighbor 172.16.1.1 update-source loopback0
R1(config-router)# neighbor 172.16.1.1 ebgp-multihop 2
R1(config-router)#  exit
R1(config)# ip route 172.16.1.1 255.255.255.255 192.168.1.18
R1(config)#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

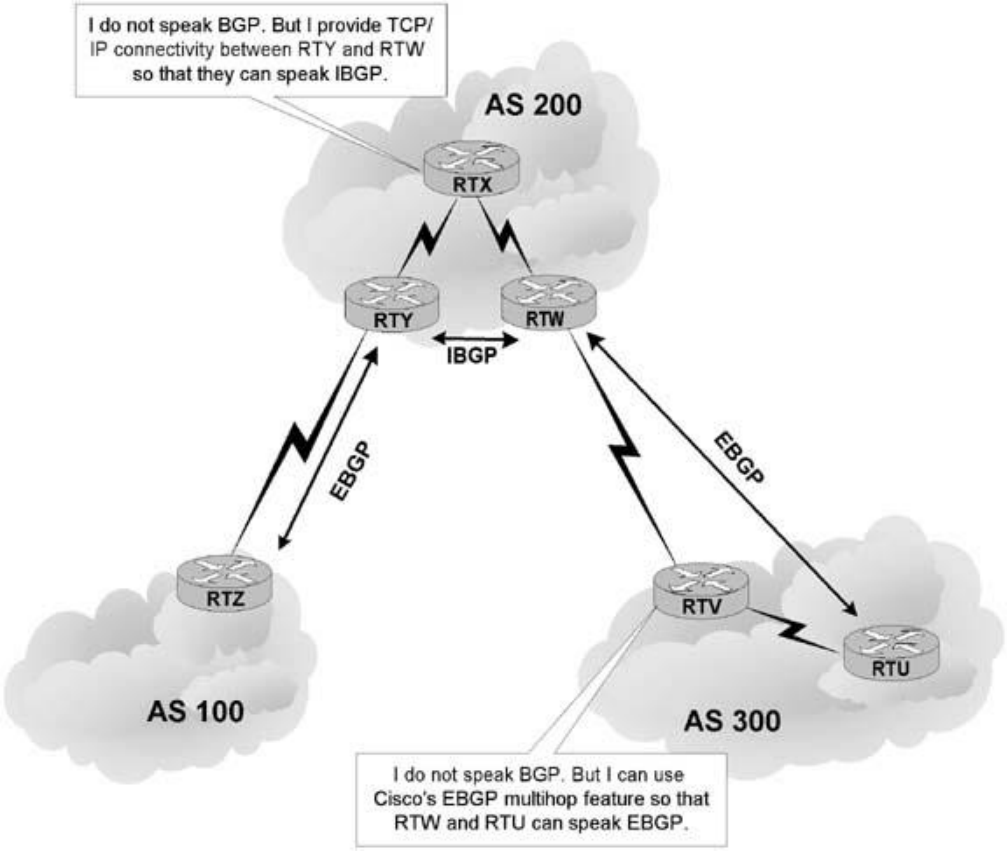


EBGP vs IBGP (Multihop)



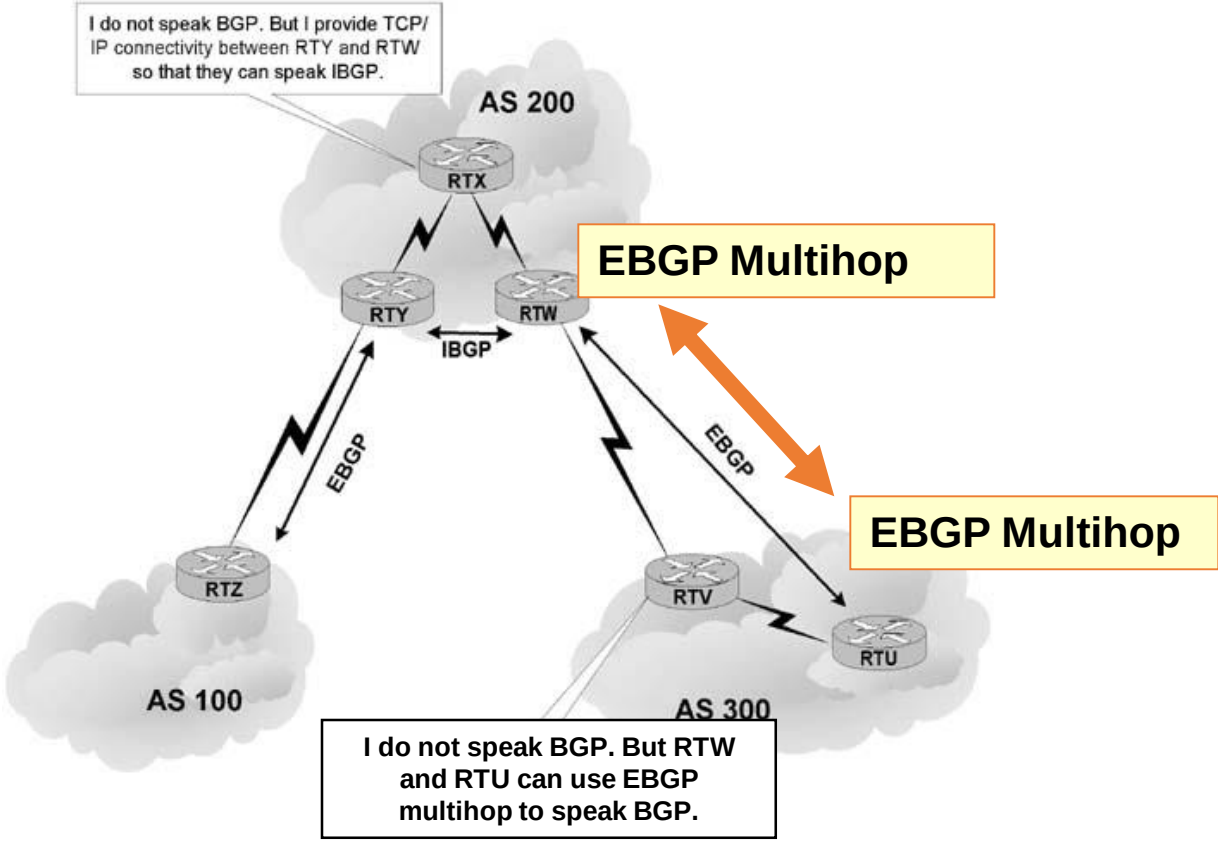
- **EBGP** peers must be **directly connected**, but there are certain exceptions to this requirement.
- In contrast, **IBGP** peers merely **require TCP/IP connectivity** within the same AS.
  - As long as RTY can communicate with RTW using TCP, both routers can establish an IBGP session.
  - If needed,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

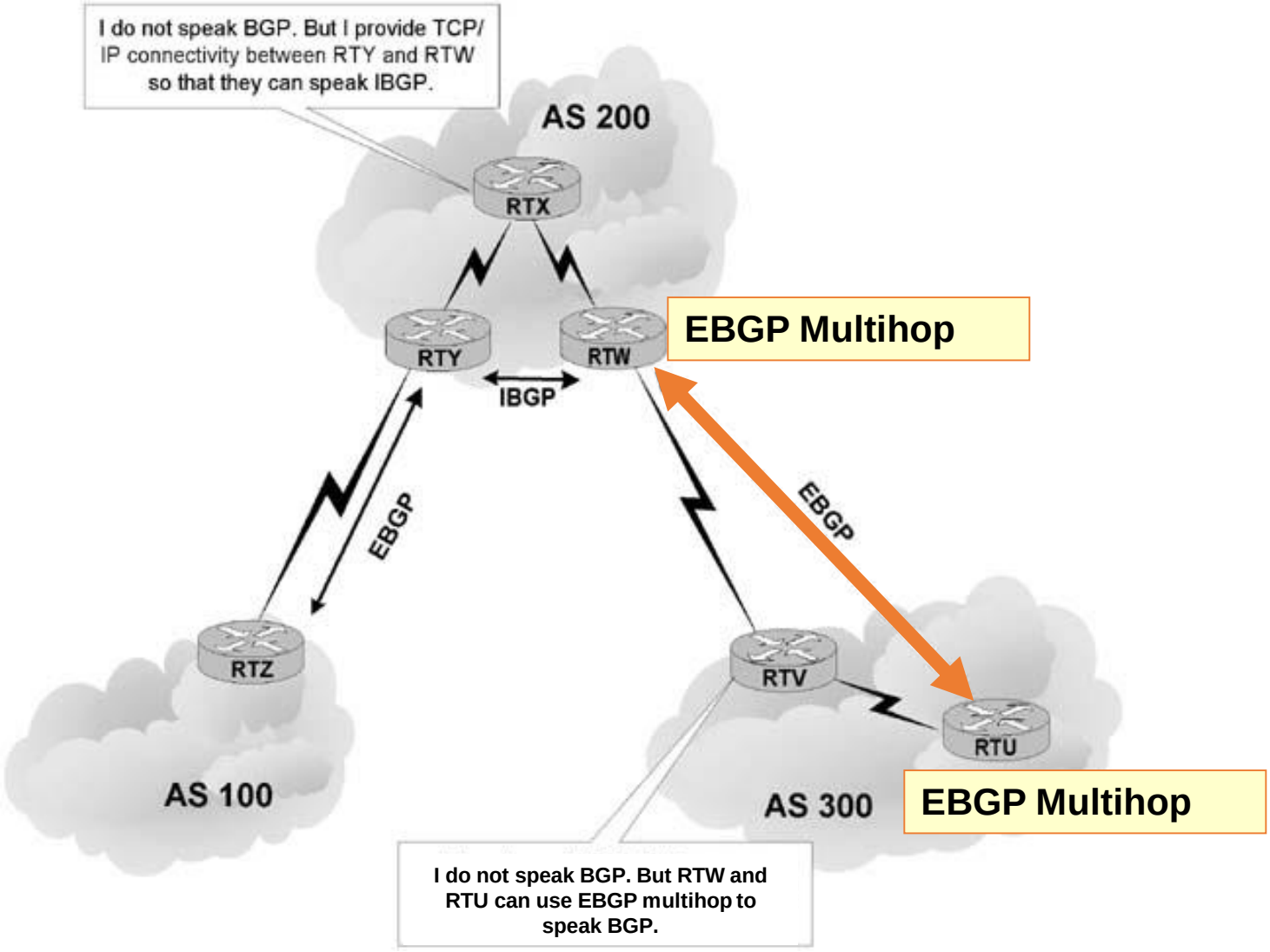
EBGP (Multihop)



- **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:

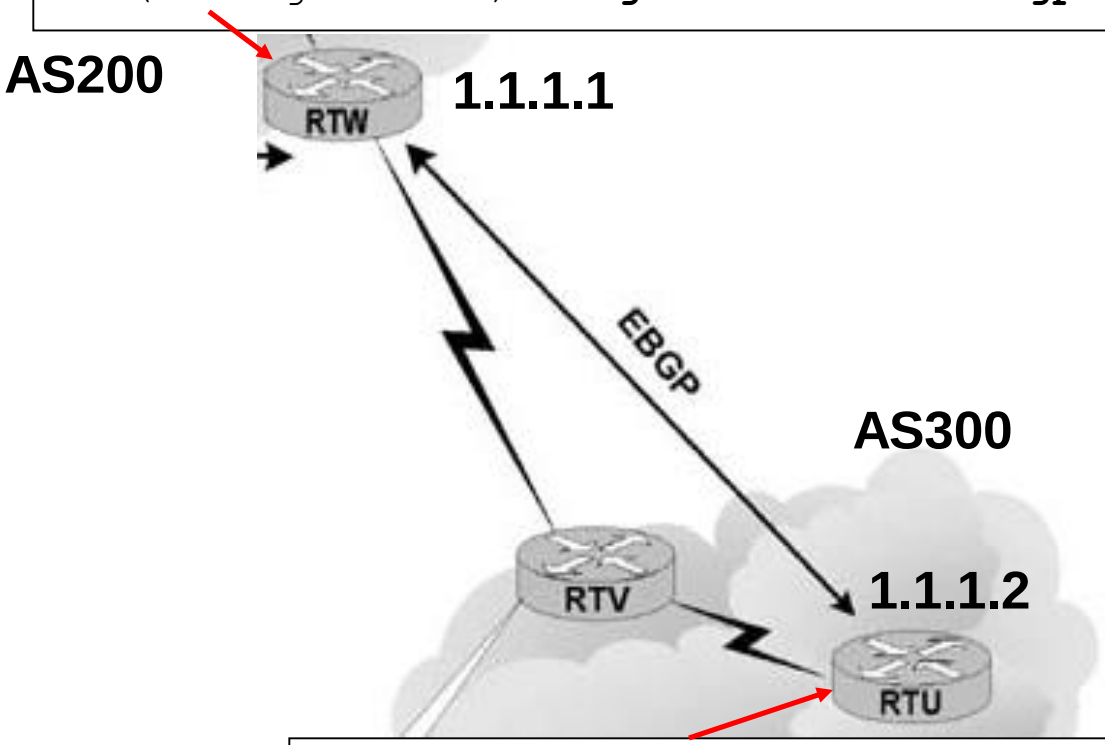
```
Router(config-router)#neighbor IP-address ebgp-multihop [hops]
```

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
```




```
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

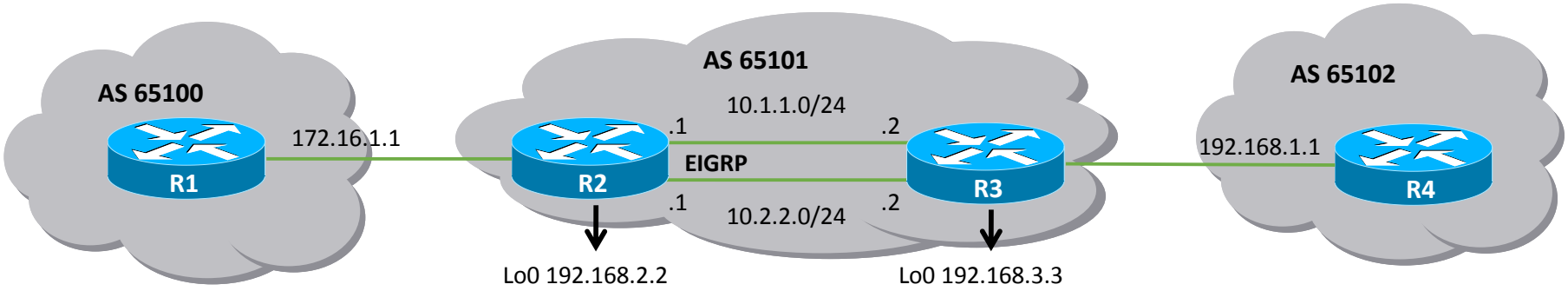
`neighbor next-hop-self` Command

- 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 it 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

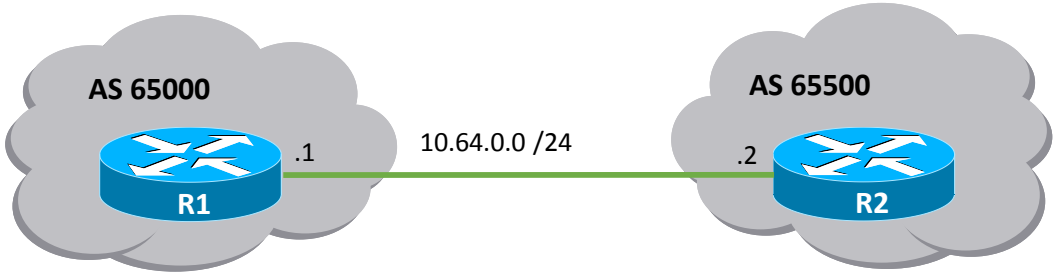
- Enable MD5 authentication between two BGP peers.

```
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 65500
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.

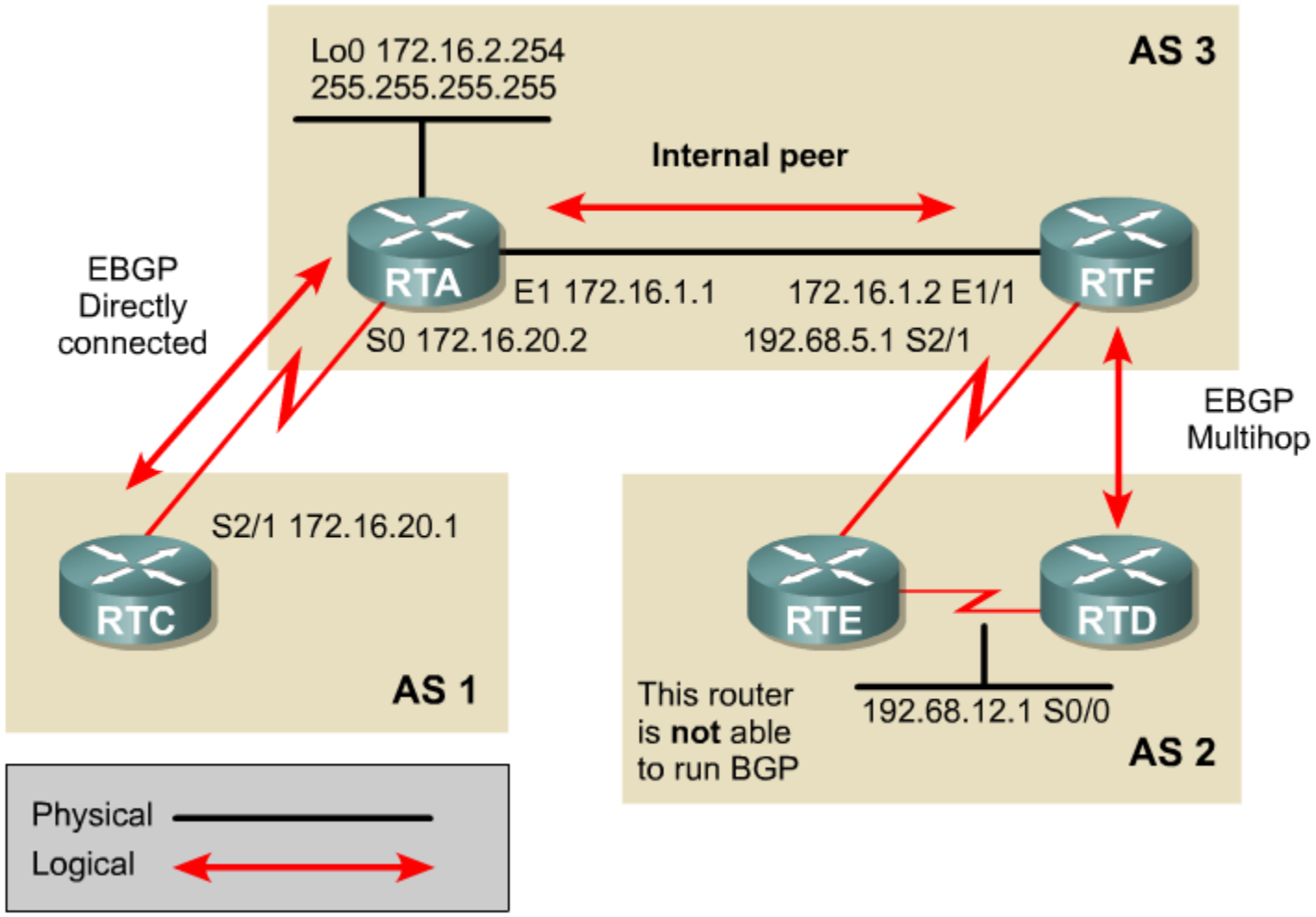
Soft Reset Inbound: Method #2

- Also called route refresh.

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

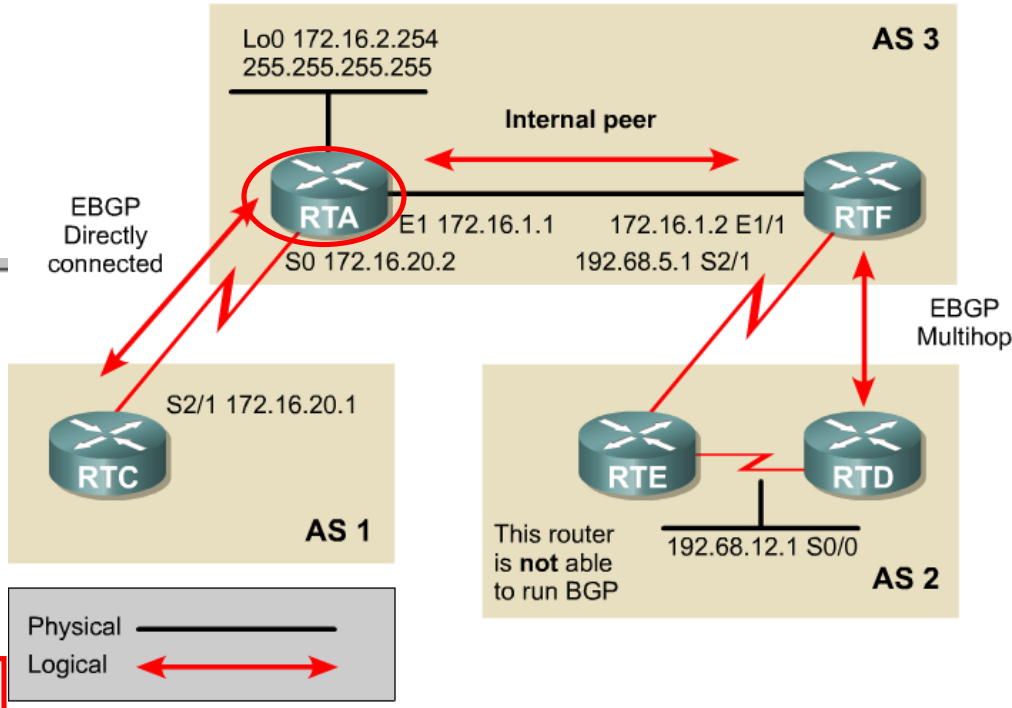
- This dynamically soft resets inbound updates.
- Unlike method #1, this method requires no preconfiguration and requires significantly less memory.

BGP Configuration Example #0



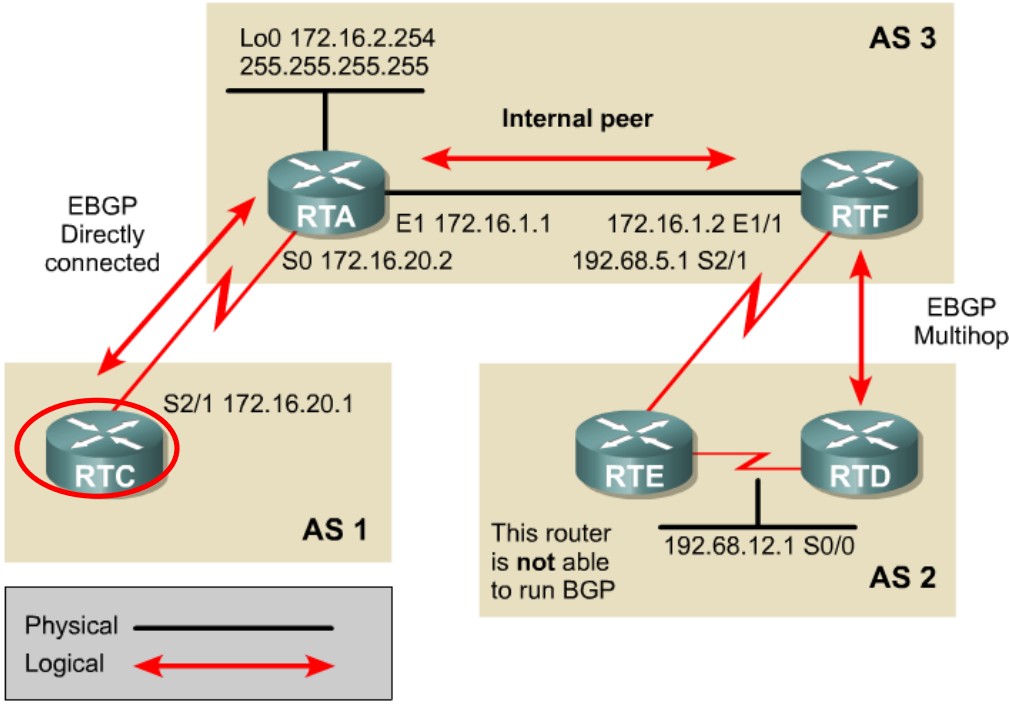
BGP Configuration Example #0

```
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#
```



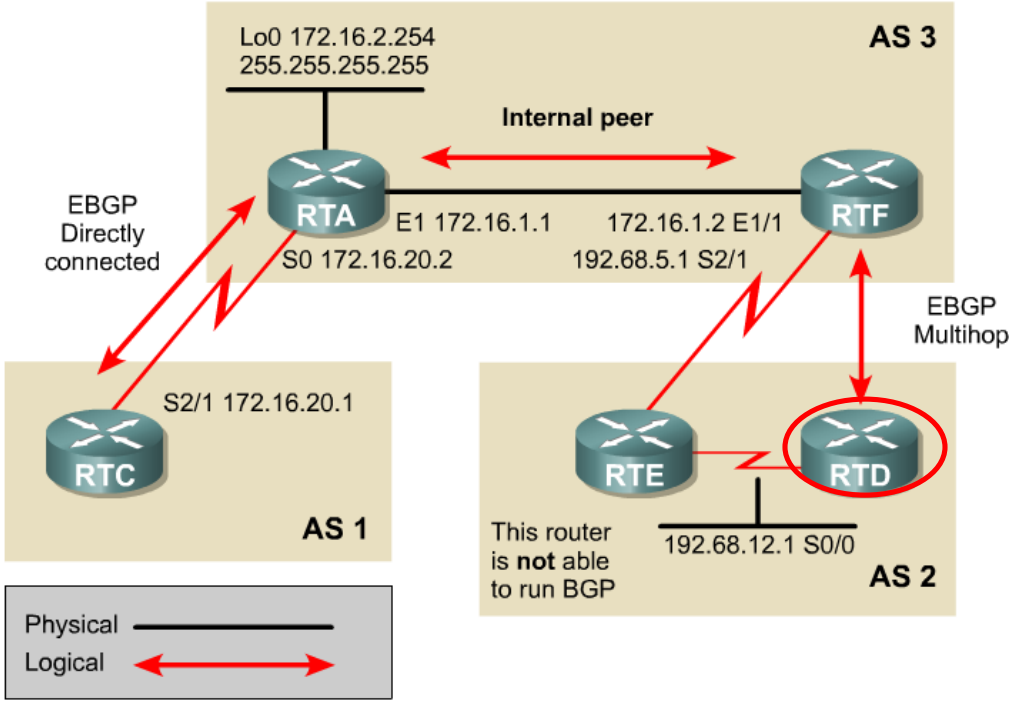
BGP Configuration Example #0

```
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#
```

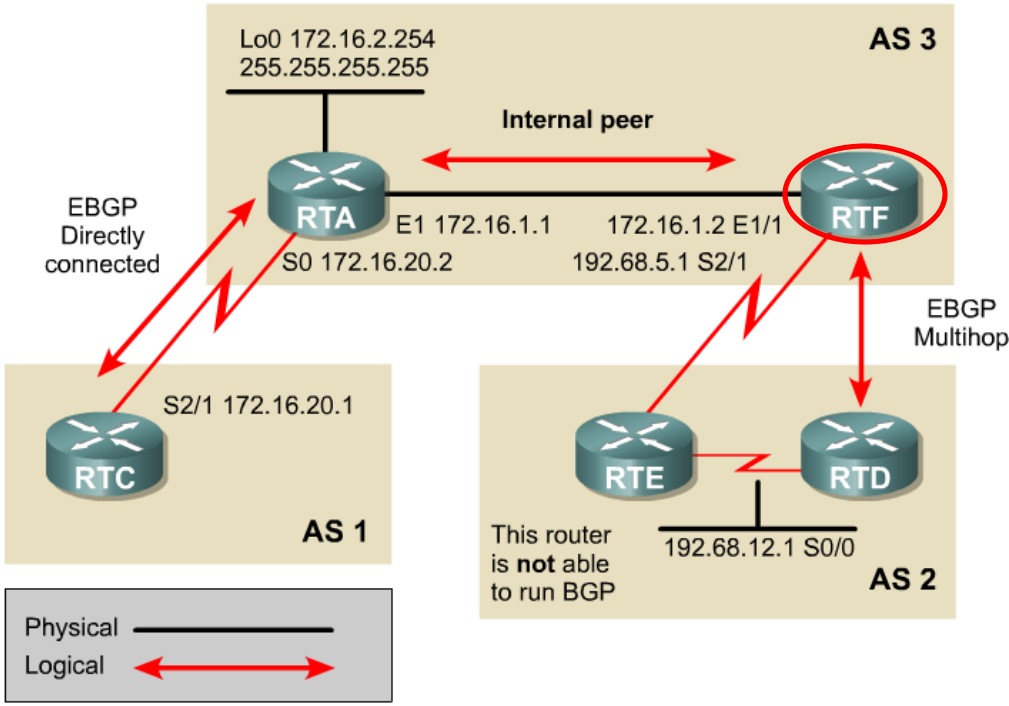


BGP Configuration Example #0

```
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.

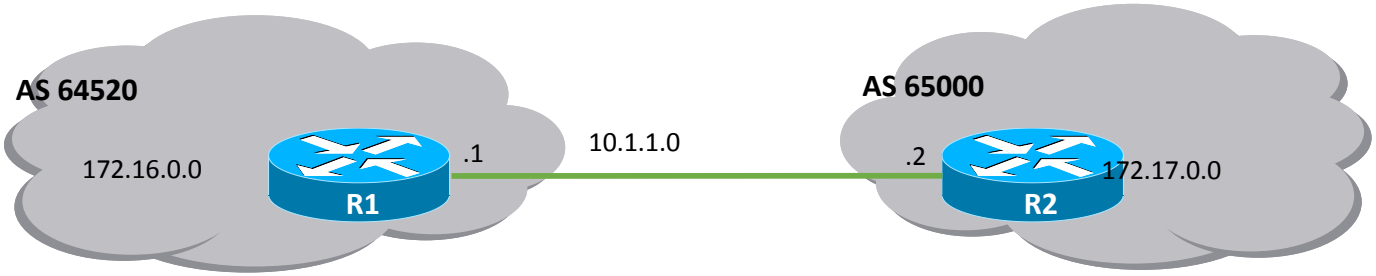
```
RTA#show ip bgp
BGP table version is 3, local router ID is 10.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   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>
```

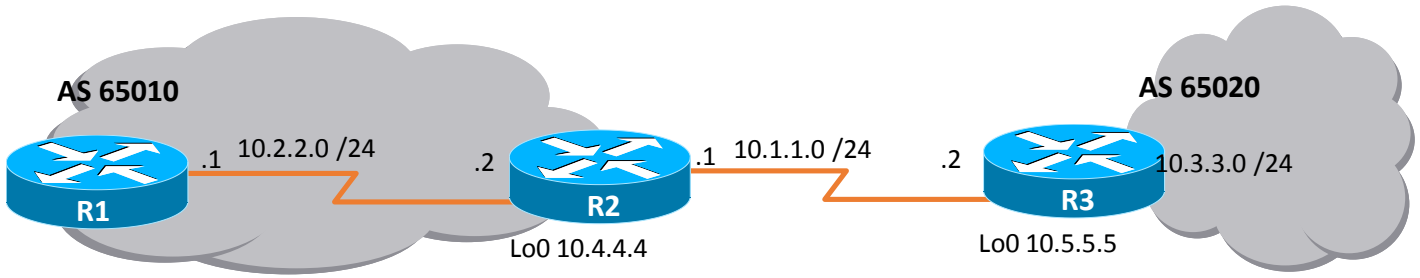
BGP Configuration Example #1




```
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)#
```

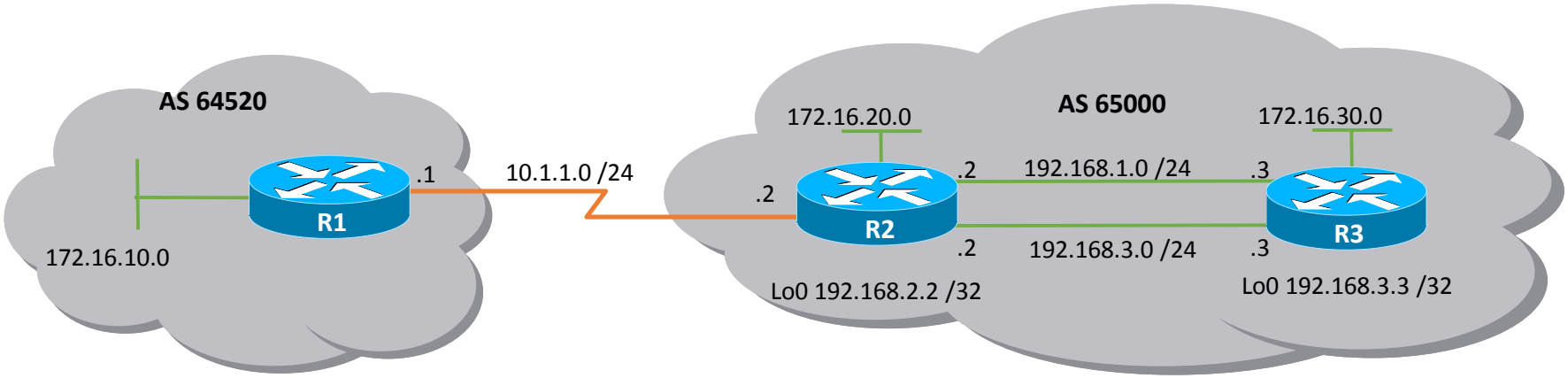
```
R2(config)# router bgp 65500
R2(config-router)# neighbor 10.1.1.1 remote-as 64520
R2(config-router)# network 172.17.0.0
R2(config-router)#
```

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

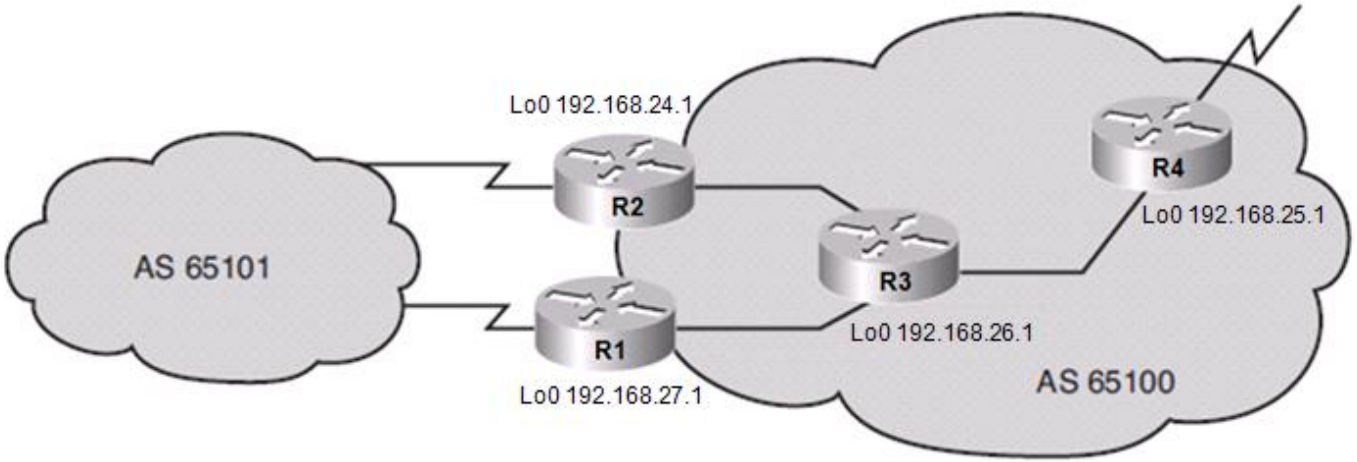
- Assign neighbors as part of the peer group.

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

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

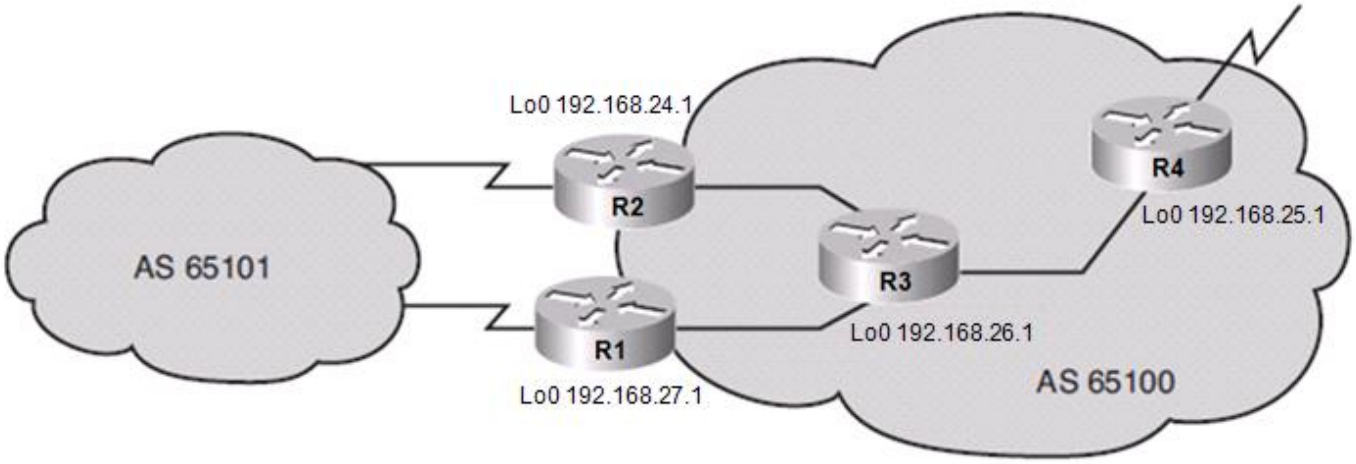
- The *ip-address* is the IP address of the neighbor that is to be assigned as a member of the peer group.
- The *peer-group-name* must already exist.
  - Note: The **clear ip bgp peer-group** *peer-group-name* EXEC command can be used to reset the BGP connections for all members of a peer group.

BGP Without Peer Group Example



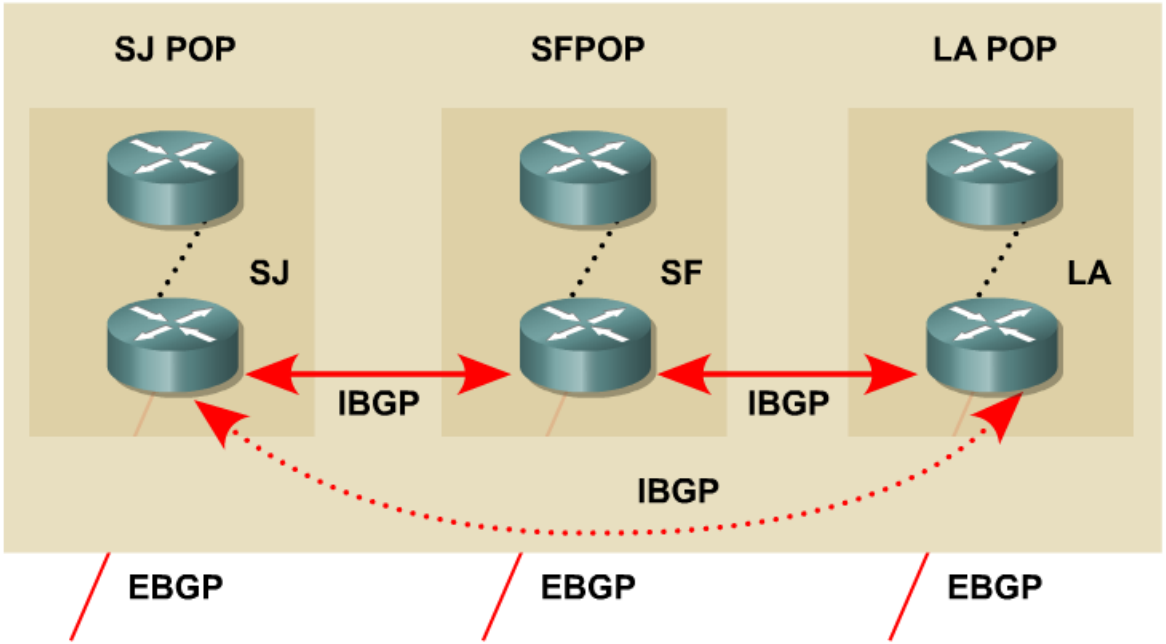

```
R1(config)# router bgp 65100
R1(config-router)# neighbor 192.168.24.1 remote-as 65100
R1(config-router)# neighbor 192.168.24.1 update-source loopback 0
R1(config-router)# neighbor 192.168.24.1 next-hop-self
R1(config-router)# neighbor 192.168.24.1 distribute-list 20 out
R1(config-router)#
R1(config-router)#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

BGP With Peer Group Example



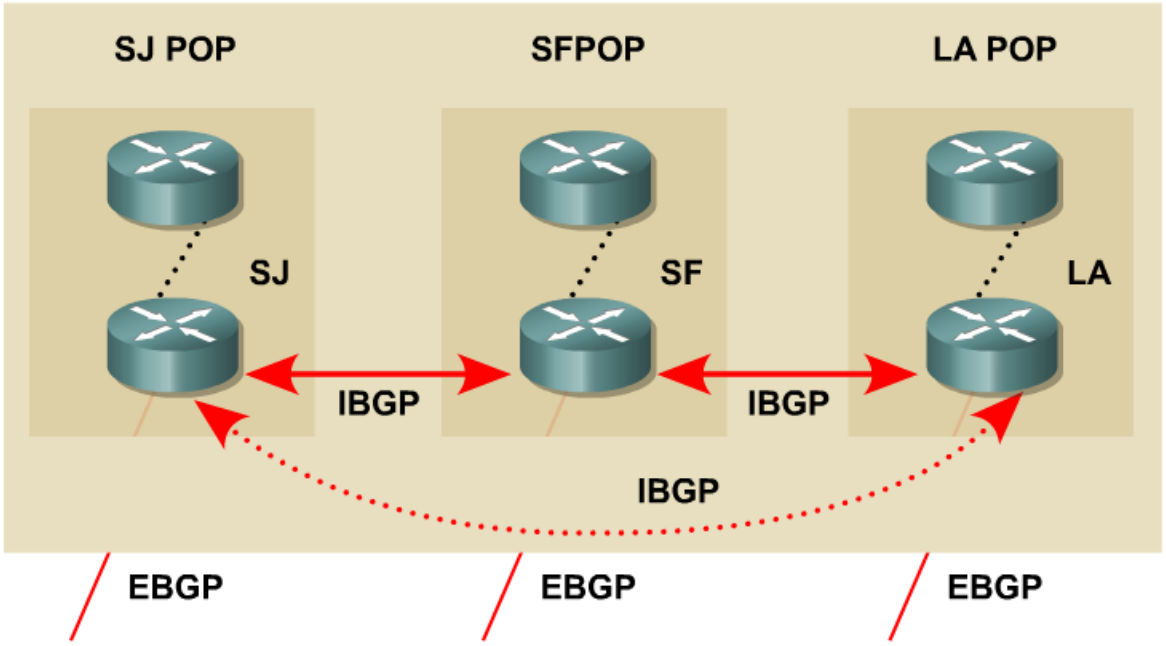

```
R1(config)# router bgp 65100
R1(config-router)# neighbor INTERNAL peer-group
R1(config-router)# neighbor INTERNAL remote-as 65100
R1(config-router)# neighbor INTERNAL update-source loopback 0
R1(config-router)# neighbor INTERNAL next-hop-self
R1(config-router)# neighbor INTERNAL distribute-list 20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

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

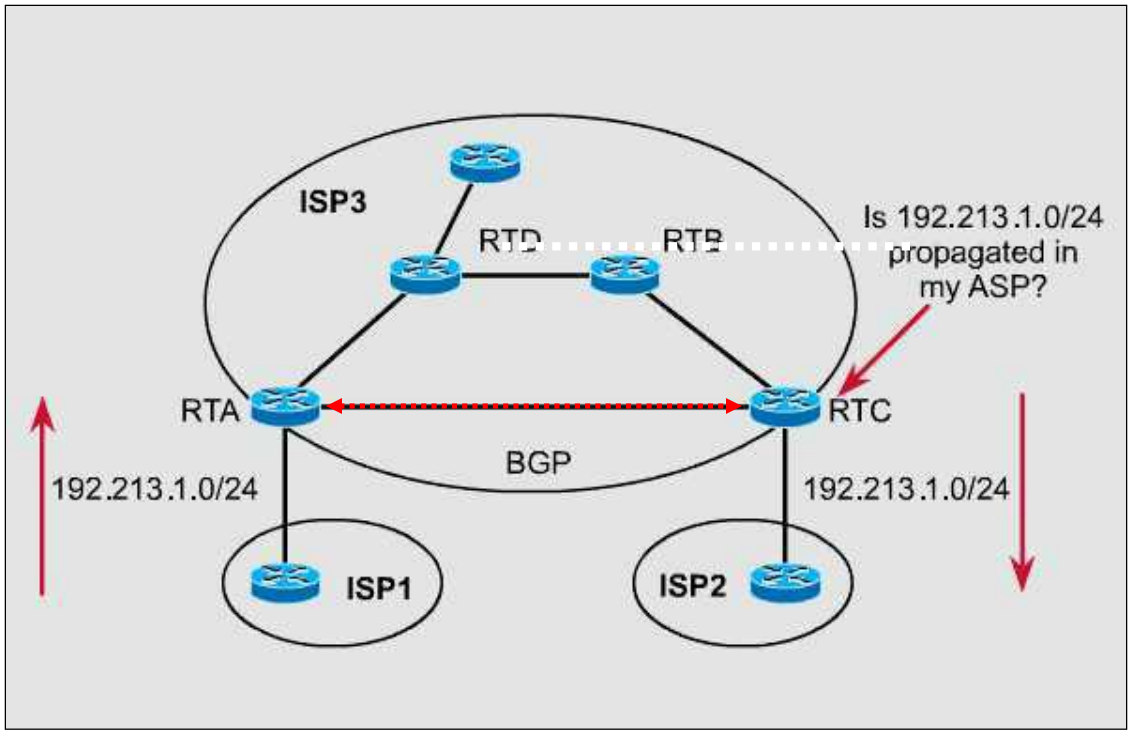


- Without dotted connection, routing in this scenario is not complete.
- EBGP routes learned by way of San Jose will not be given to Los Angeles, and EBGP routes learned by way of Los Angeles will not be given to San Jose.
- This is because the San Francisco router will not advertise IBGP routes between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

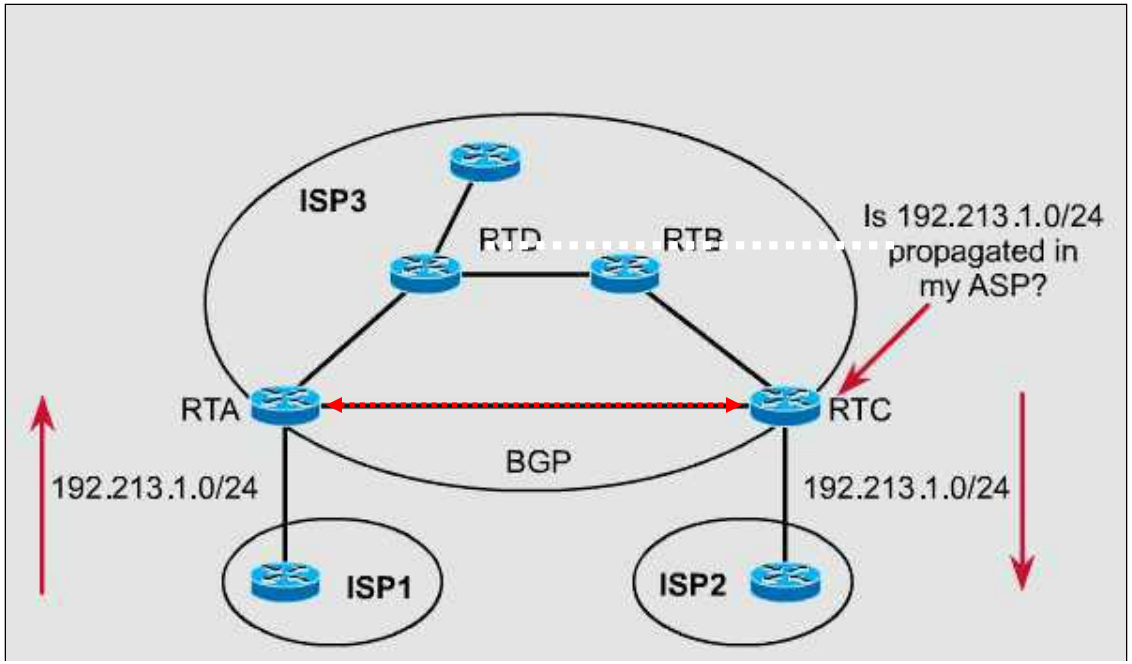
- Recall that the BGP synchronization rule states that:
  - "*A BGP router should not use, or advertise a route learned by IBGP, unless that route is local or is learned from the IGP*."
- By default synchronization disabled, therefore BGP can use and advertise to an external BGP neighbor routes learned 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



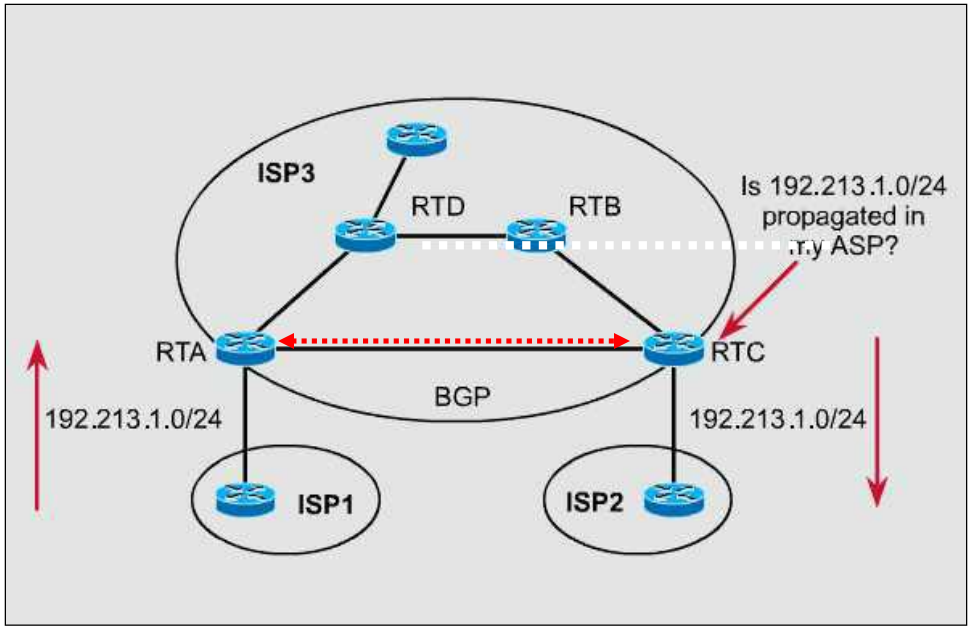
- When an IBGP router receives an update about a destination from an IBGP peer, it tries to verify reachability to that destination via an **IGP**, such as RIP or OSPF.
- If the IBGP router can't find the destination network in it's **IGP** routing table, it **will not** advertise the destination to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

BGP Synchronization



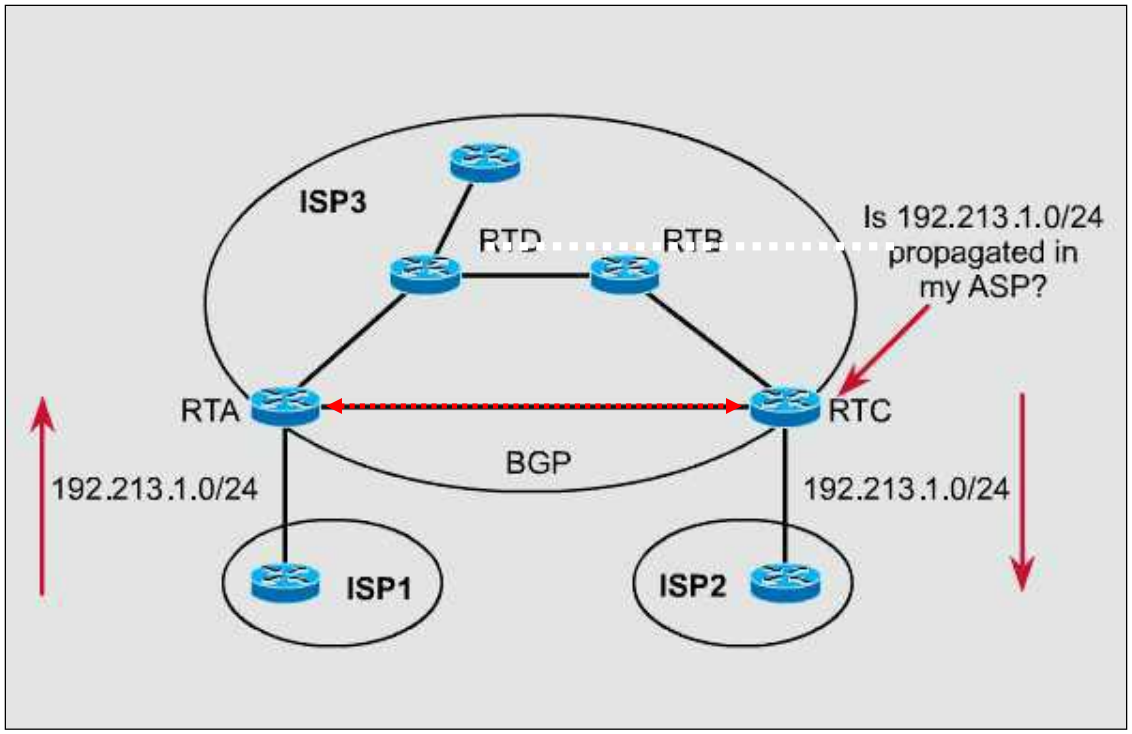
- If the route is **not** reachable through the **IGP** running within the AS, non-BGP routers will not be able to route traffic passing through the AS towards this destination.
- It is pointless to advertise destinations to external peers if traffic sent through this AS is going to be dropped by some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

BGP Synchronization



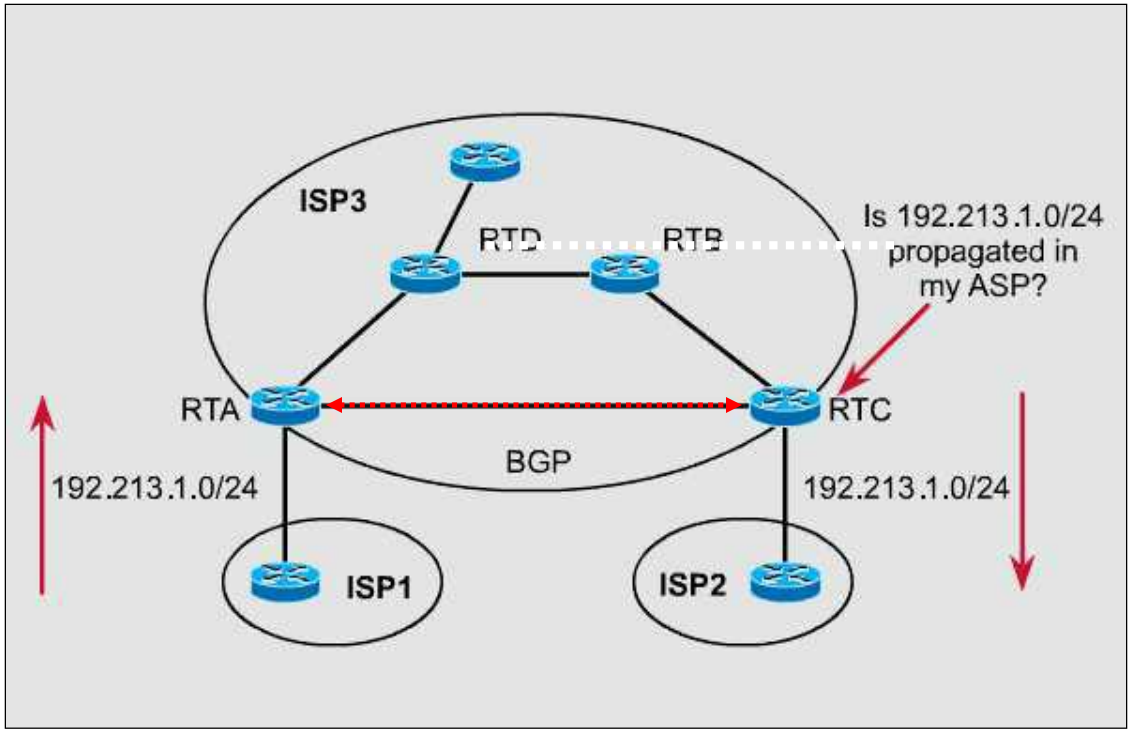
- The **BGP synchronization rule** states that a BGP router **(RTC)** should not advertise to external neighbors **(ISP2)** destinations **(192.213.1.0/24)** learned from inside BGP neighbors **(RTA)** unless those destinations are also known via an **IGP** **(RTD and RTB)**.
- If a router knows about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

BGP Synchronization



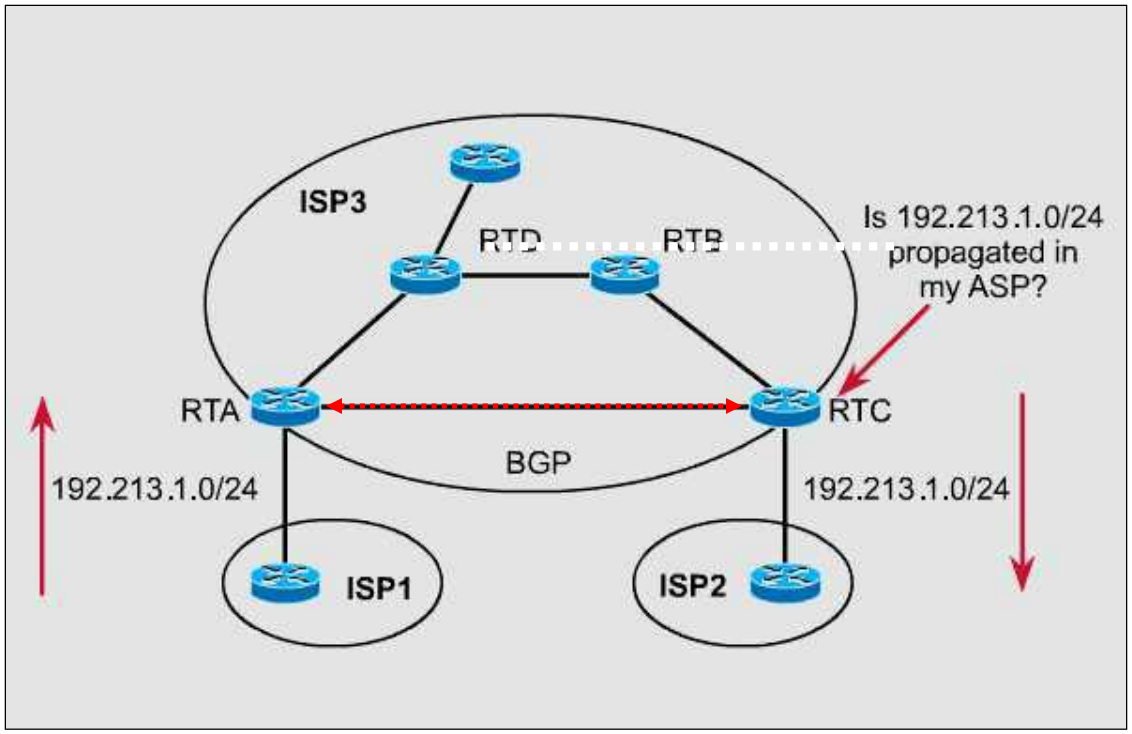
- If the IBGP router **(RTC)** does have an **IGP** route to this destination, the route is considered ***synchronized***, and the router will announce it to other BGP peers **(ISP2).**
- *Otherwise*, the router will treat the route as not being synchronized with the **IGP** and will not advertise it.

BGP Synchronization



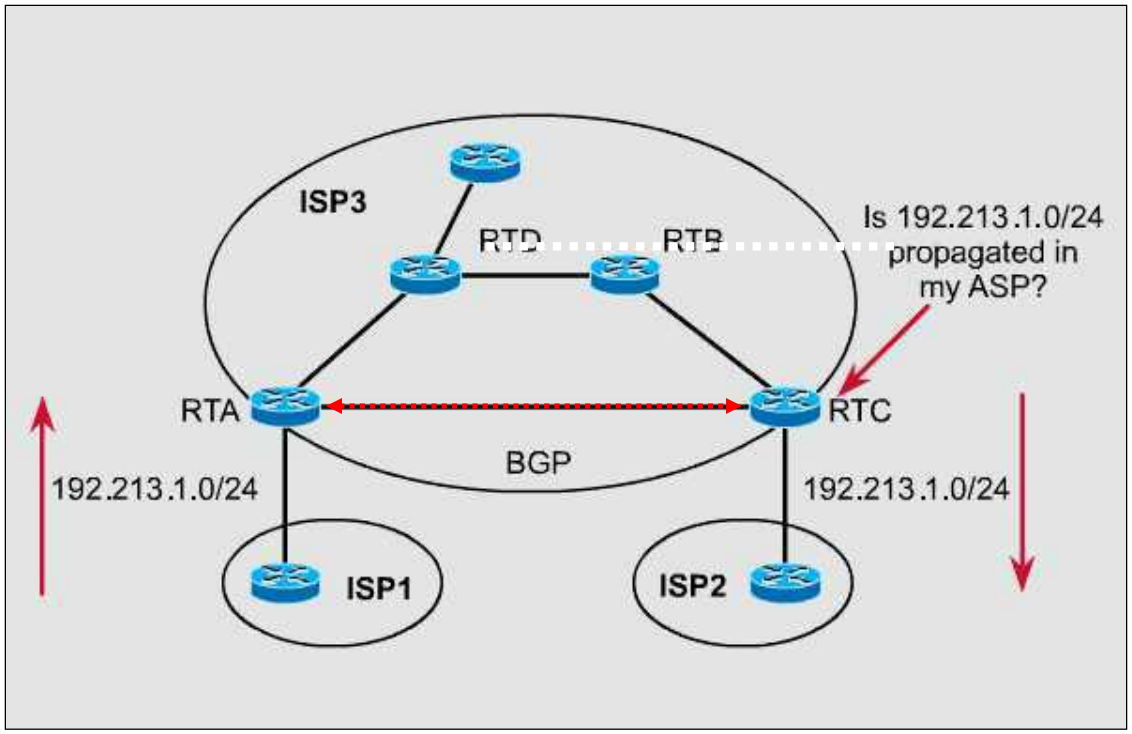
- The consequence of injecting BGP routes inside an AS is costly.
- Redistributing routes from BGP into the **IGP** will result in major overhead on the internal routers, which might not be equipped to handle that many routes.
- Besides, carrying all external routes inside an AS is not really necessary.

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

Monitoring Received BGP Routes

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

Verifying BGP: `show ip bgp`

Display the BGP topology database (the BGP table).

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

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

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

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

This route is offered to the IP routing table.

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

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

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

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

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

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

If blank the route is from the current autonomous system.

The last column displays the ORIGIN attribute).

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

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

Verifying BGP: show ip 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.

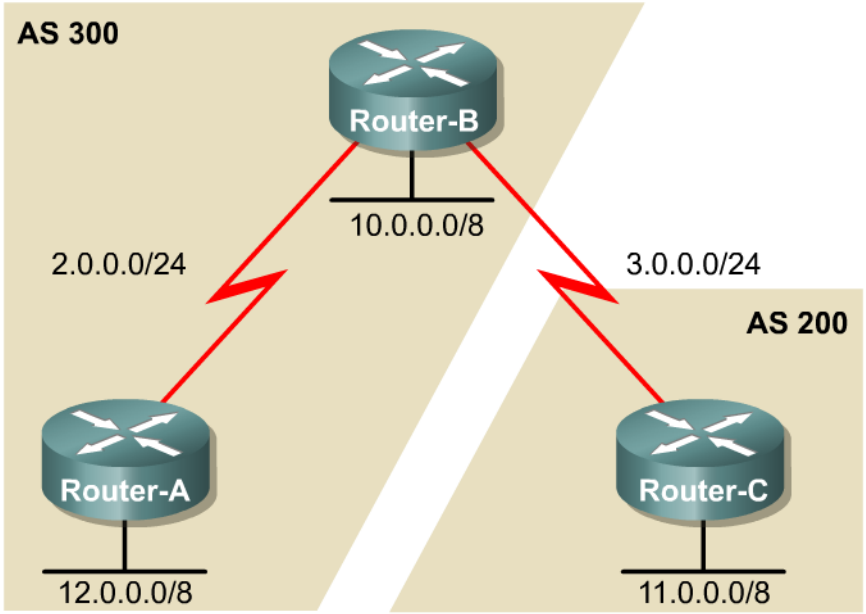
```
R1# show ip bgp summary
BGP router identifier 10.1.1.1, local AS number 65001
BGP table version is 124, main routing table version 124
9 network entries using 1053 bytes of memory
22 path entries using 1144 bytes of memory
12/5 BGP path/bestpath attribute entries using 1488 bytes of memory
6 BGP 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
```

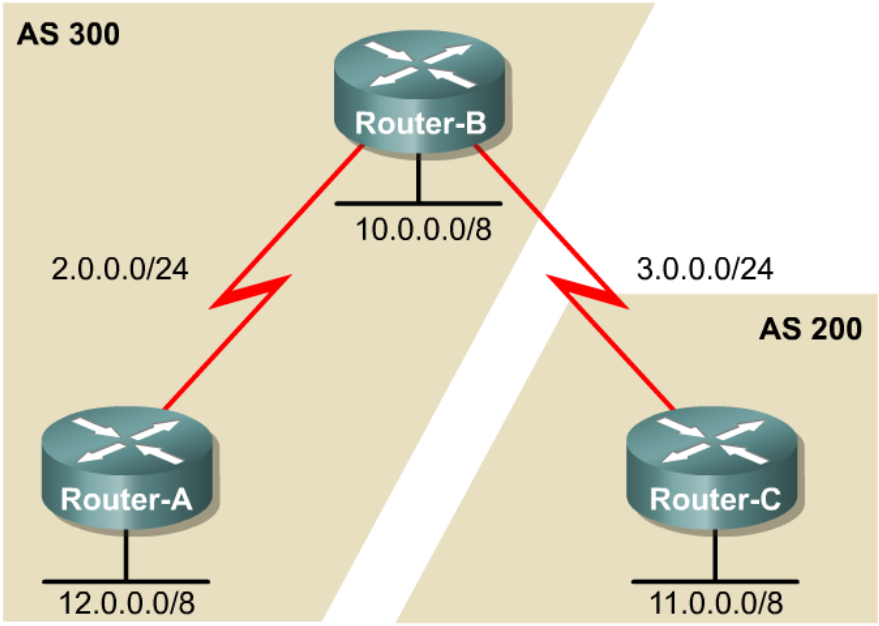
Example




```
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 - inte
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
Router-A#
```

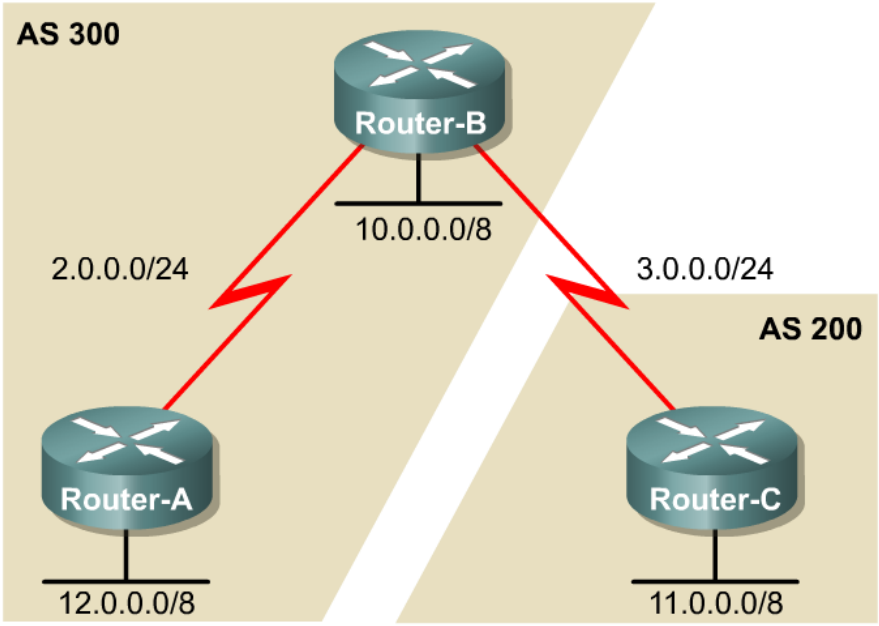
Example




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

** Partial list*

`match as-path` Command

- Match a BGP autonomous system path access list.

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

match *as-path path-list-number*

- The *`path-list-number`* is the AS path access list.
  - It can be an integer from 1 to 199.
- The value set by this command 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
access-list-number | *name*	The number or name of a standard or extended access list to be used to test incoming packets. If multiple access lists are specified, matching any one results in a match.
prefix-list *prefix-list-name*	Specifies the name of a prefix list to be used to test packets. If multiple prefix lists are specified, matching any one results in a match.

`set` Commands Used in BGP

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

** Partial list*

`set weight` Command

- Specify the BGP weight for the routing table.

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

```
set weight number
```

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

`set local-preference` Command

- Specify a preference value for the AS path.

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

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

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

`set as-path` Command

- Modify an AS path for BGP routes.

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

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

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

`set metric` Command

- Specify a preference value for the AS path.

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

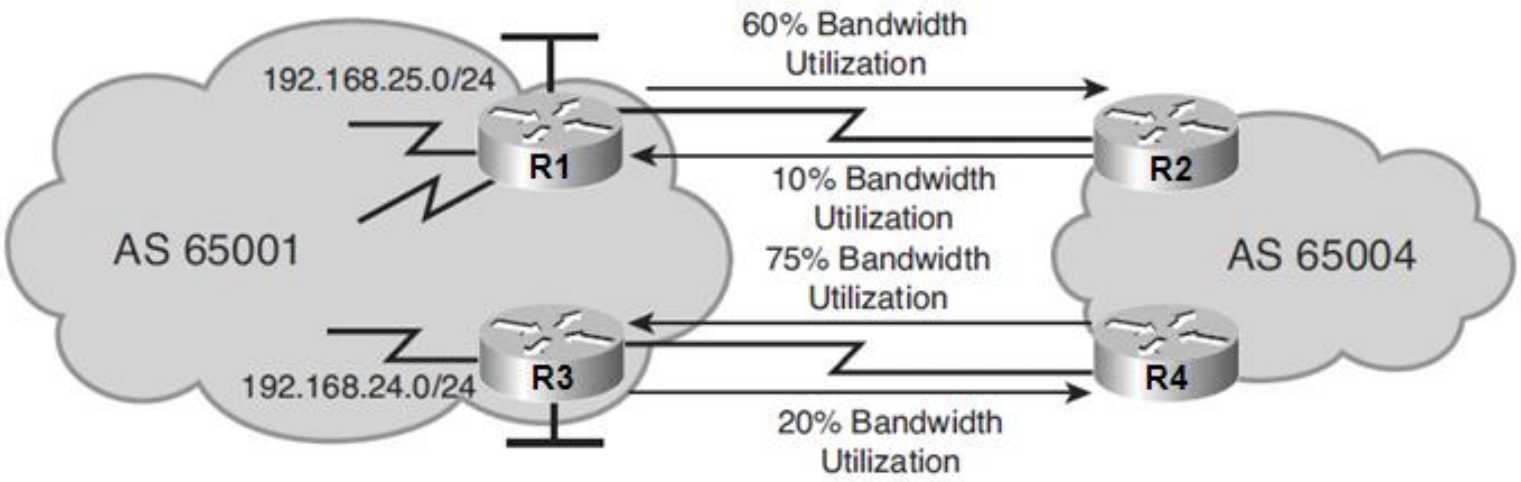
```
set metric metric-value
```

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

BGP Path Manipulation

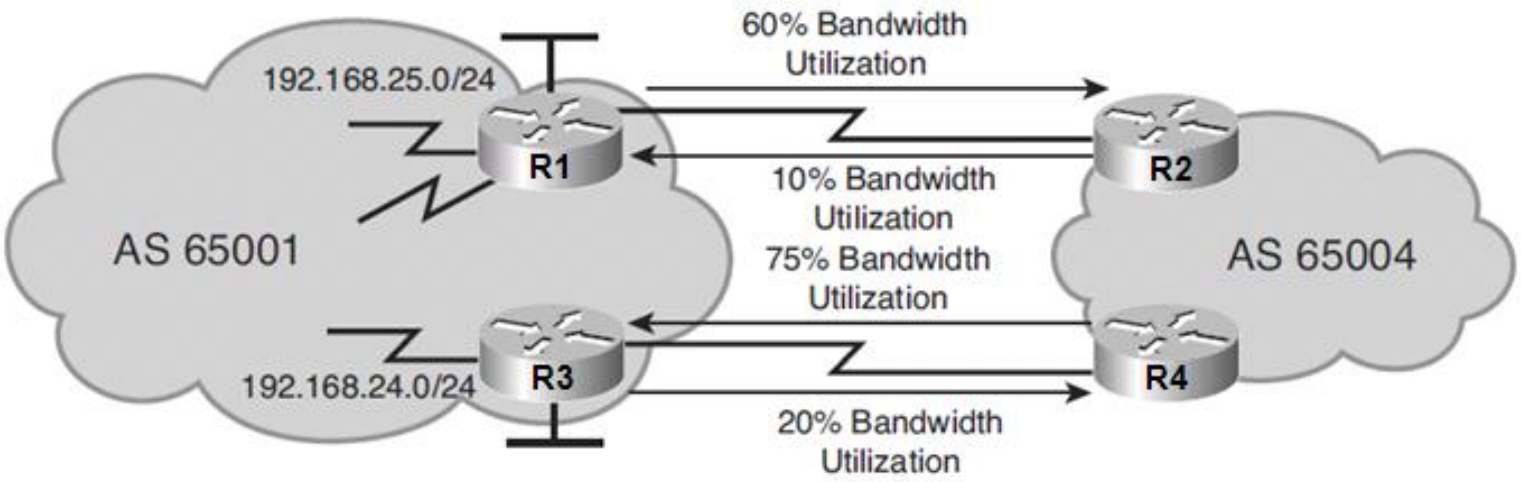
- Unlike IGPs, BGP was never designed to choose the quickest path.
- BGP was designed to manipulate traffic flow to maximize or minimize bandwidth use.

BGP Without Routing Policy Example #1



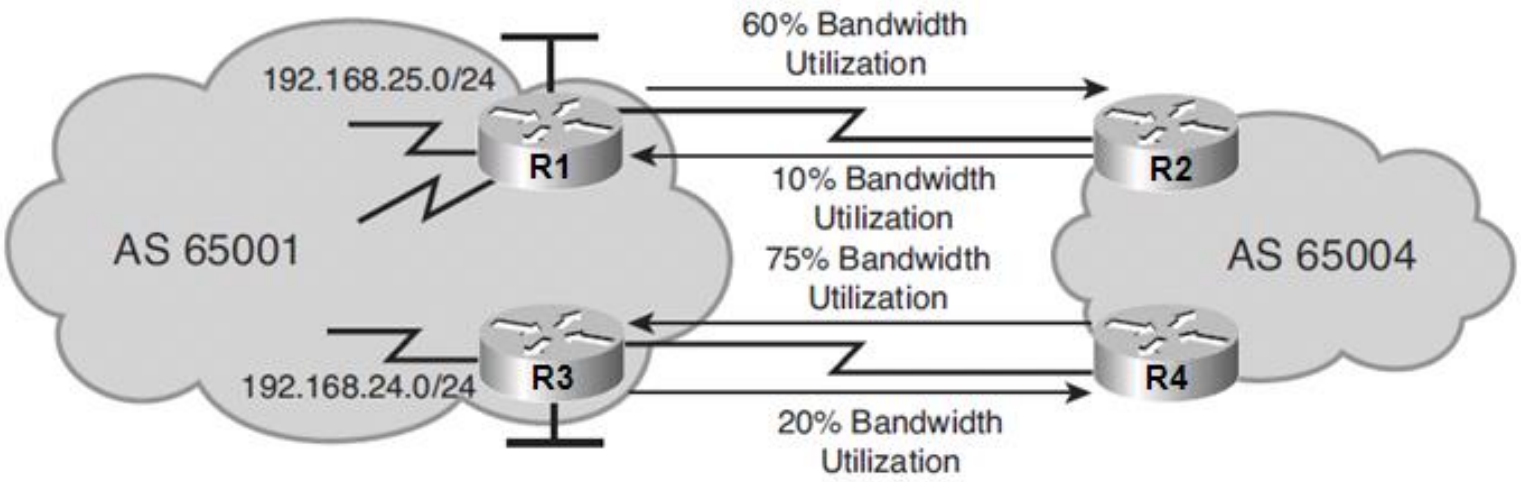
- In this example consider that:
  - R1 is using 60% of its outbound bandwidth to AS 65004.
  - R3 is using 20% of its outbound bandwidth to AS 65004.
  - R2 is using 10% of its outbound bandwidth to AS 65001.
  - R4 is using 75% of its outbound bandwidth to AS 65001.
- Traffic should be diverted using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

Which traffic should be re-routed?



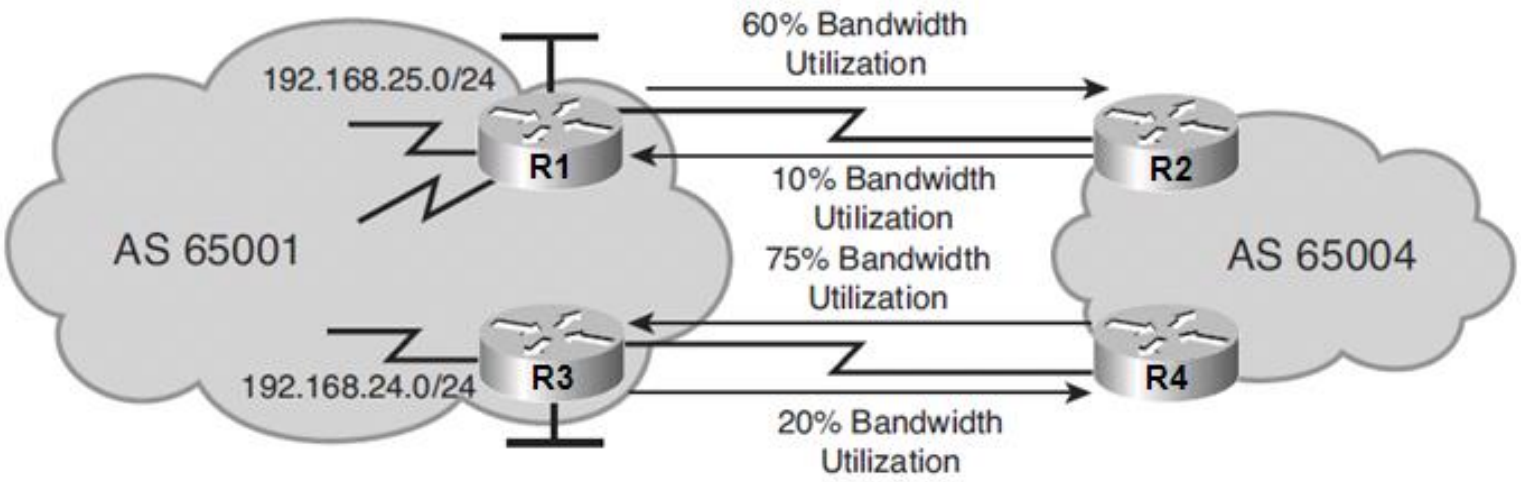
- To determine which path to manipulate, perform a traffic analysis on Internet-bound traffic by examining the most heavily visited addresses, web pages, or domain names.
  - Examine network management records or accounting information.
- If a heavily accessed traffic pattern is identified, a route map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

BGP With Routing Policy Example #1



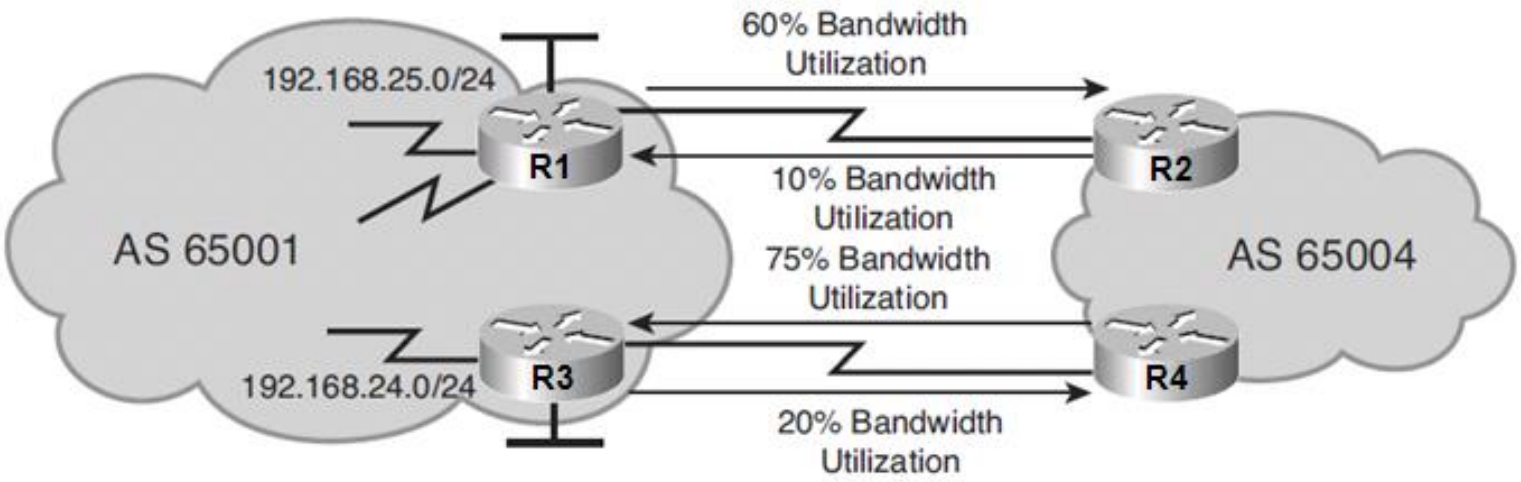
- For example, assume that 35% of all traffic from AS 65001 has been going to http://www.cisco.com.
  - The administrator does a reverse DNS lookup and obtains the Cisco IP address and AS number.
- A route map can be used to change the local preference to manipulate packets destined to Cisco's network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

BGP Routing Policy Example #2



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

BGP Routing Policy Example #2



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

BGP Route Selection Process

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

Change the Weight

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

BGP Route Selection Process

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

Changing the Default Weight Example

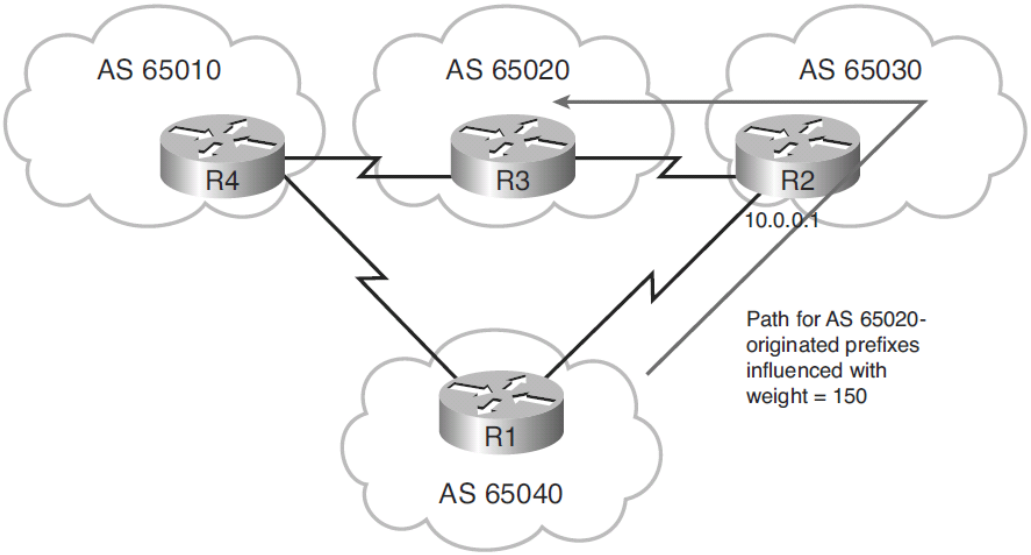
- Assign a default weight to all routes from a peer.

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

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

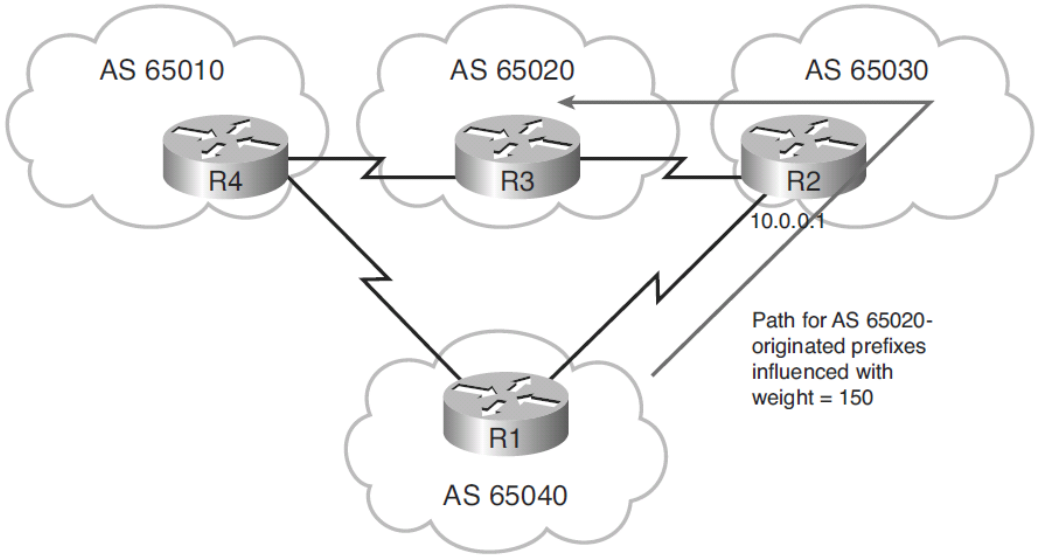
- Routes learned through another BGP peer have a default weight of 0 and routes sourced by the local router have a default weight of 32768.
- The *number* is the weight to assign.
  - Acceptable values are from 0 to 65535.
- The route with the highest weight will be chosen as the preferred route when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

Changing Weight with Route Map Example



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

Changing Weight with Route Map Example




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

Configure an Autonomous System ACL

- Configure an autonomous system path filter.

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

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

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

Regular Expression Syntax

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

Regular Expression Examples

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

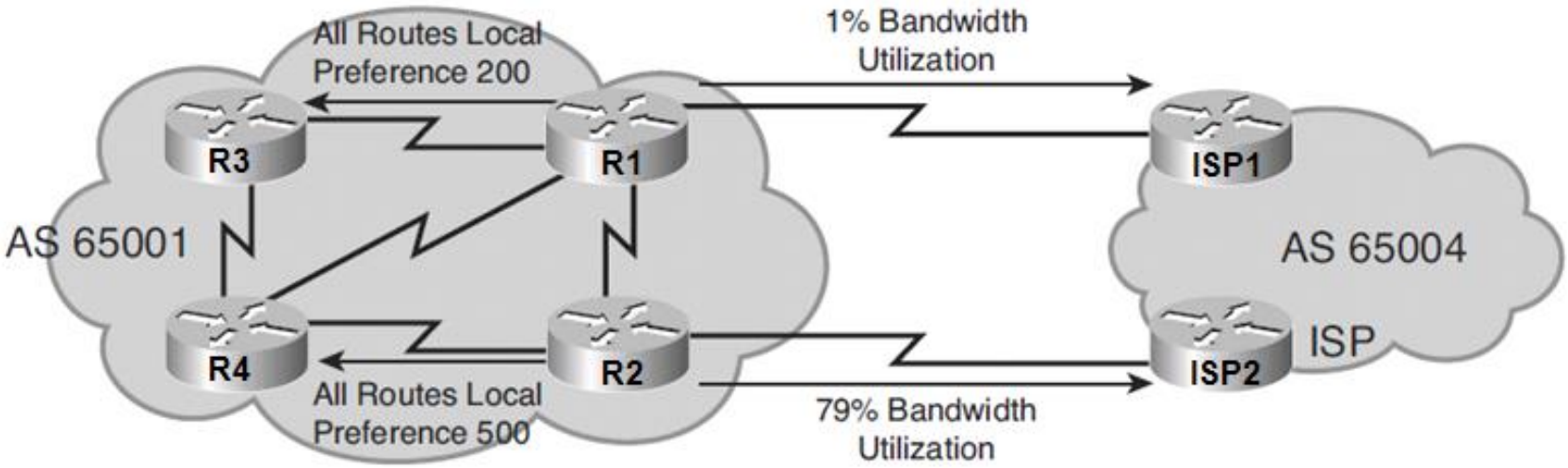
- Change the default local preference for outgoing routes.

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

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

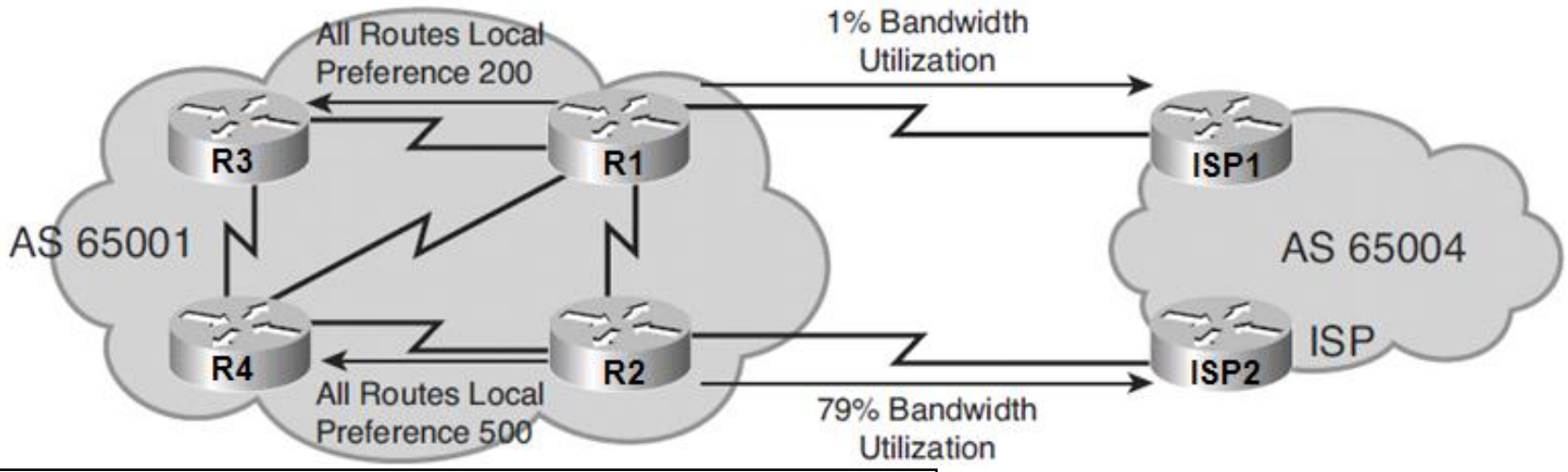
- The local preference attribute applies a degree of preference to a route during the BGP best path selection process.
  - The attribute is exchanged only between iBGP peers.
  - The route with the highest local preference is preferred.
- The *`number`* is the local preference value from 0 to 4294967295.
  - 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

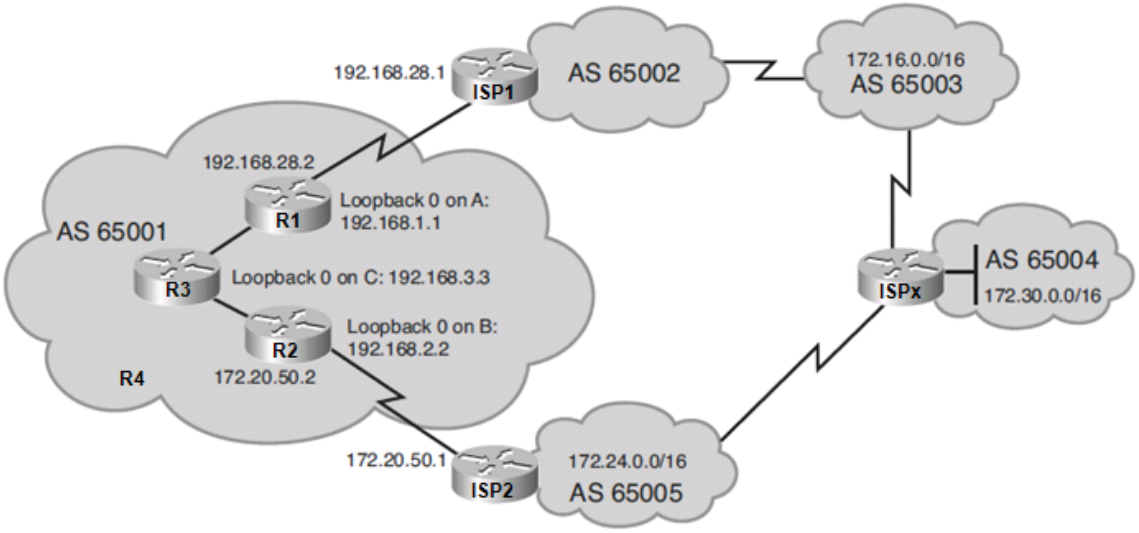



```
R1(config)# router bgp 65001
R1(config-router)# bgp default local-preference 200
R1(config-router)#
```

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

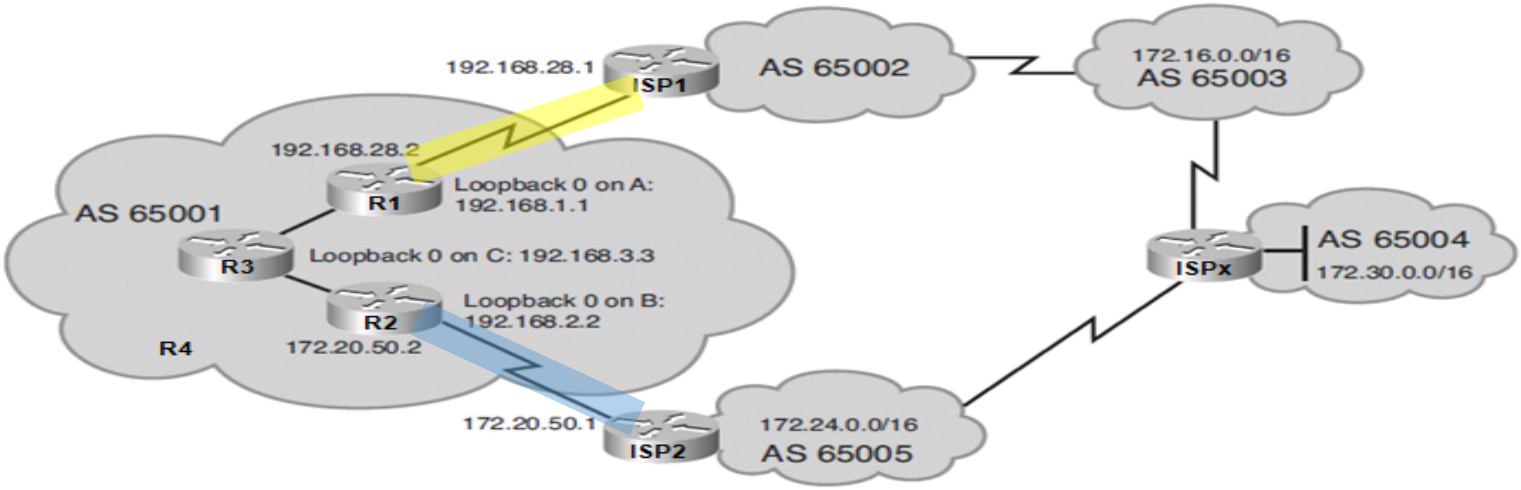
- The resulting configuration makes the IBGP routers in AS 65001 send all Internet bound traffic to R2, but the R1 to ISP1 link is underutilized.
  - Route maps could be configured to select specific routes to have a higher local preference.

Local Preference and Route Map Example



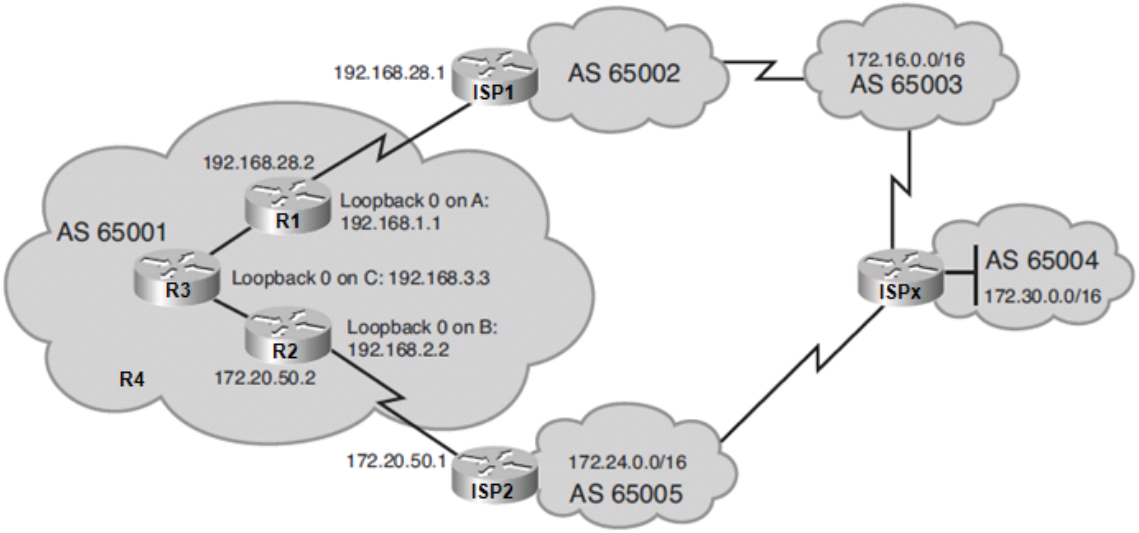
- The BGP routing policy results in the following:
  - All routes have a weight of 0 and a default local preference of 100.
  - BGP uses the shortest AS-path to select the best routes as follows:
    - For network 172.16.0.0, the shortest AS-path is through ISP1.
    - For network 172.24.0.0, the shortest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

Local Preference and Route Map Example



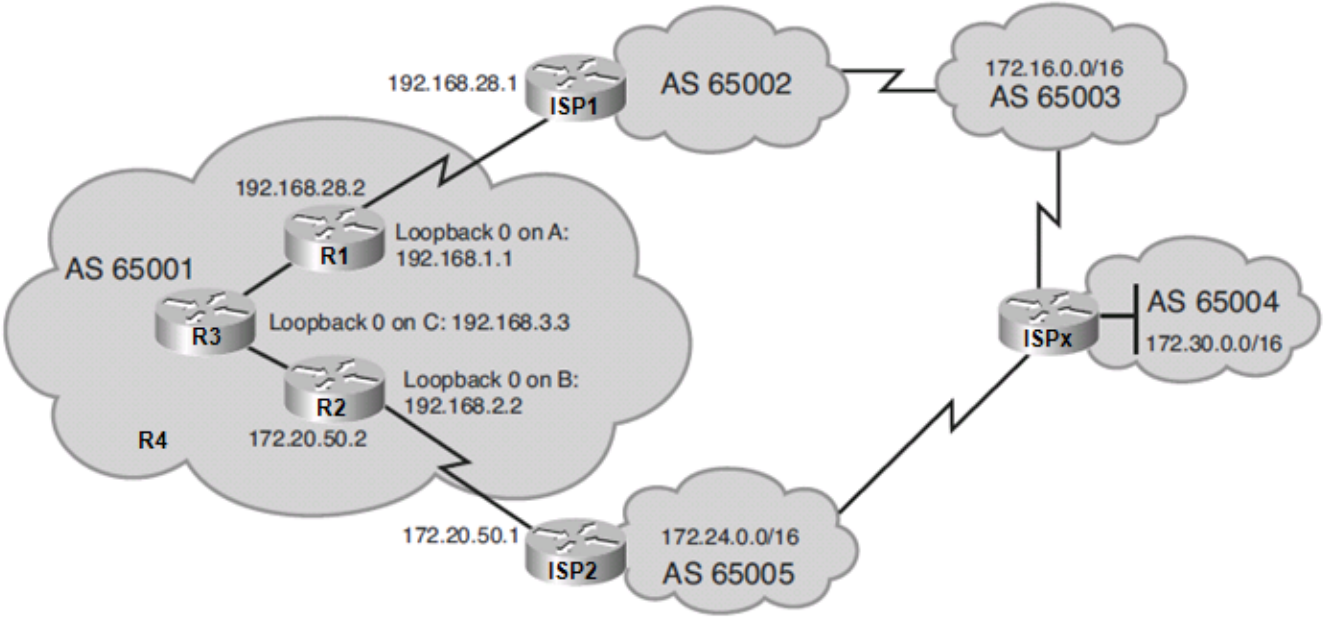

```
R3# show ip bgp
BGP table version is 7, local router ID is 192.168.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,  r
   RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
Network             Next Hop      Metric LocPrf  Weight Path
* i172.16.0.0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



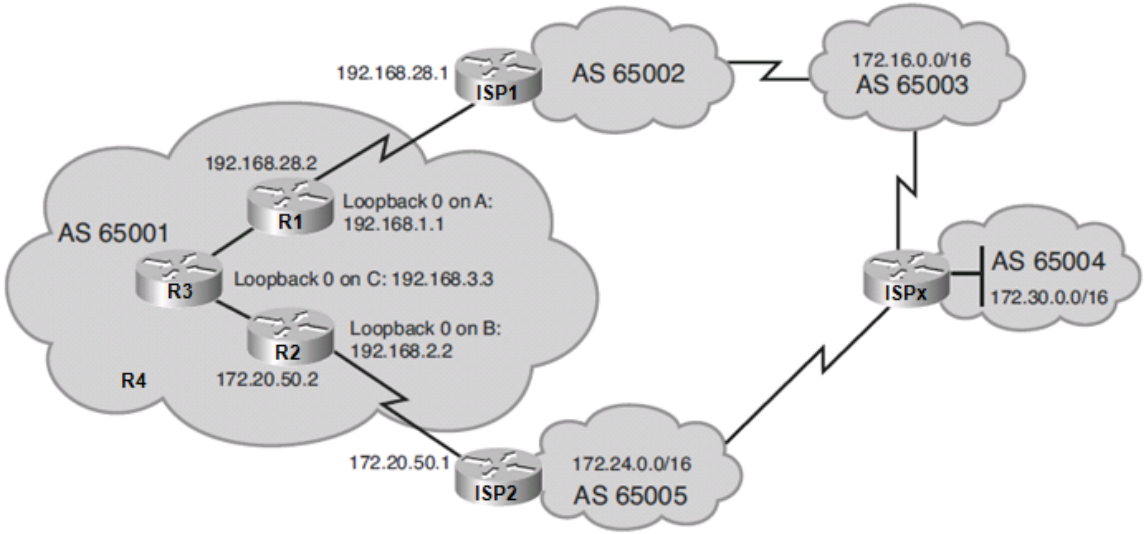
- A traffic analysis reveals the following traffic patterns:
  - 10% of traffic flows from R1 to ISP1 to network 172.16.0.0.
  - 50% of Internet traffic flow from R2 to ISP2 to networks network 172.24.0.0 and network 172.30.0.0.
  - The remaining 40 percent is going to other destinations.
- A solution is to use route maps to divert traffic to 17230.0.0 through R1.

Local Preference and Route Map Example



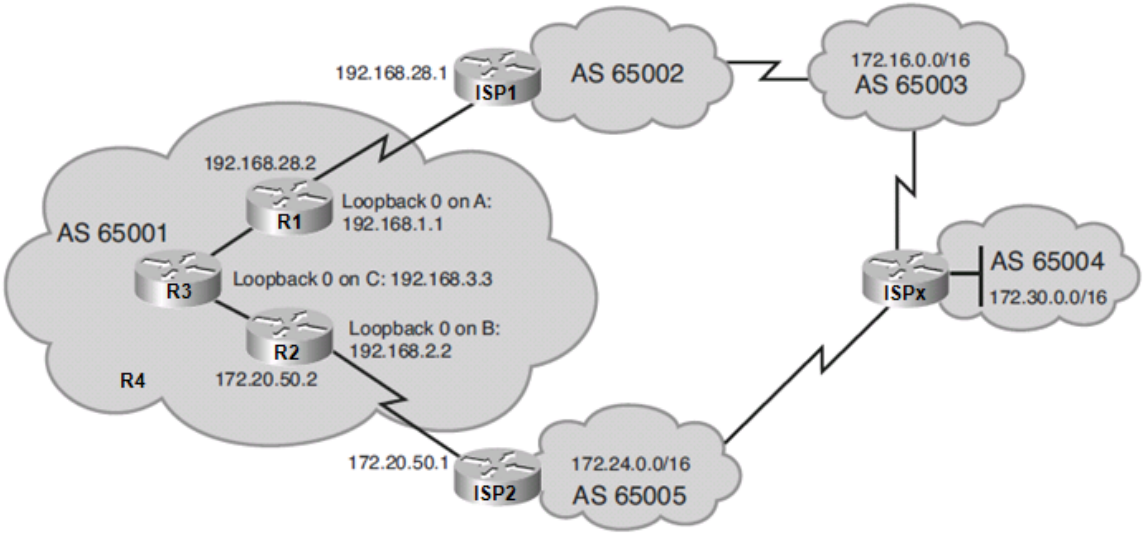

```
R1(config)# access-list 65 permit 172.30.0.0 0.0.255.255
R1(config)#
R1(config)# route-map LOCAL_PREF permit 10
R1(config-route-map)# match ip address 65
R1(config-route-map)# set local-preference 400
R1(config-route-map)#
R1(config-route-map)# route-map LOCAL_PREF permit 20
R1(config-route-map)#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

Local Preference and Route Map Example




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

Local Preference and Route Map Example




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

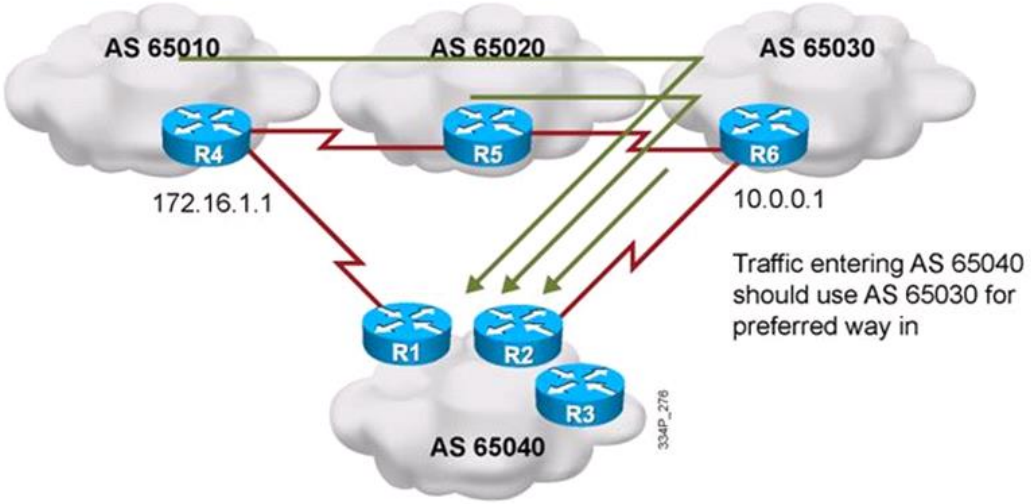
Modifying the AS Path

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

BGP Route Selection Process

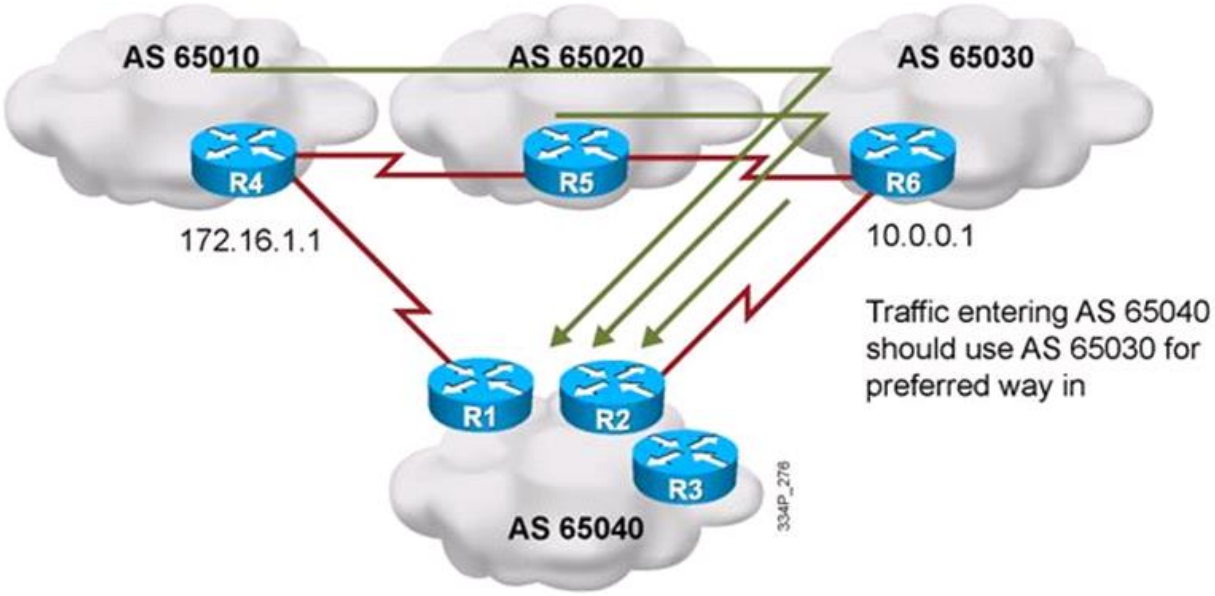
1. Prefer highest Weight
2. Prefer highest LOCAL_PREF
3. Prefer locally generated routes
4. Prefer shortest AS_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID
10.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

Modifying the AS Path Example



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

Modifying the AS Path Example




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

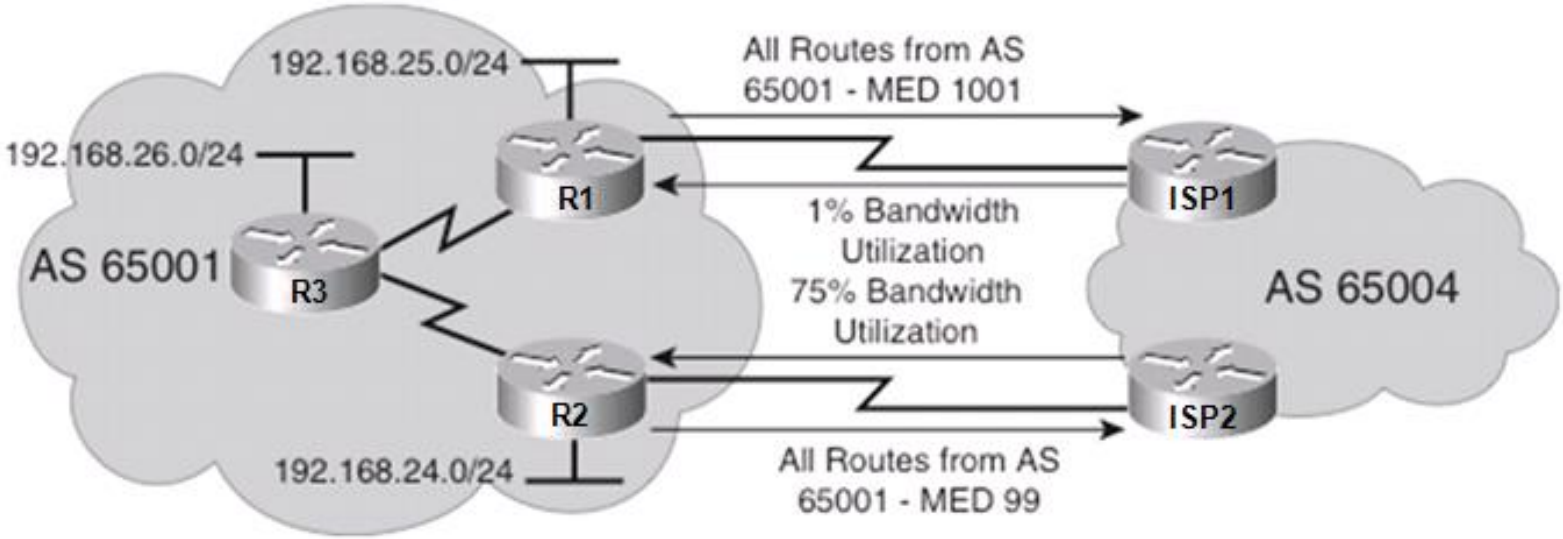
Setting the MED

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

BGP Route Selection Process

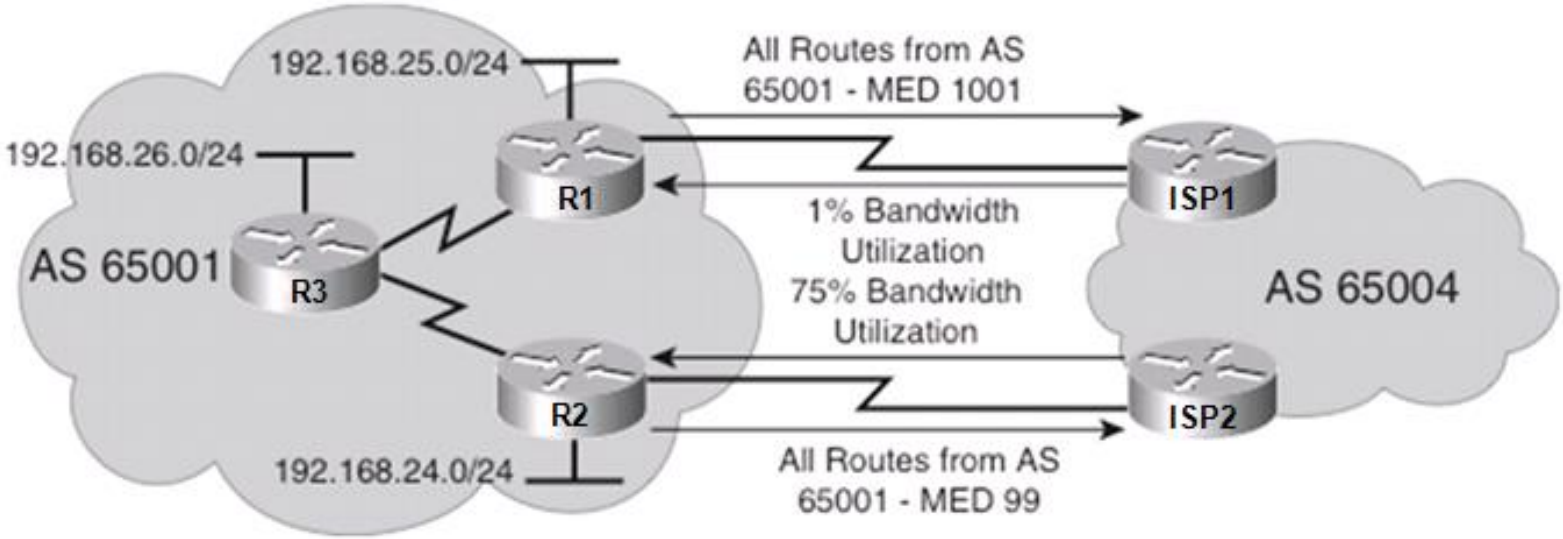
1. Prefer highest Weight
2. Prefer highest LOCAL_PREF
3. Prefer locally generated routes
4. Prefer shortest AS_PATH
5. Prefer lowest ORIGIN (IGP < EGP < incomplete)
6. Prefer lowest MED
7. Prefer EBGP over IBGP
8. Prefer routes through closest IGP neighbor
9. Prefer routes with lowest BGP router ID
10.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

Setting the Default MED Example



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

Setting the Default MED Example

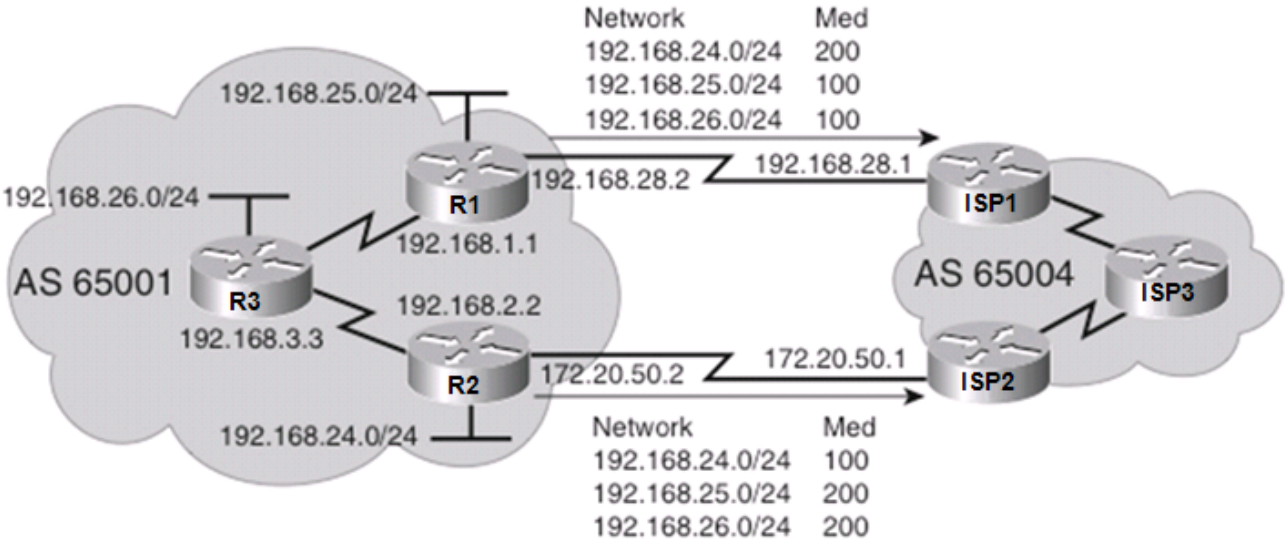



```
R1(config)# router bgp 65001
R1(config-router)# default metric 1001
R1(config-router)#
```

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

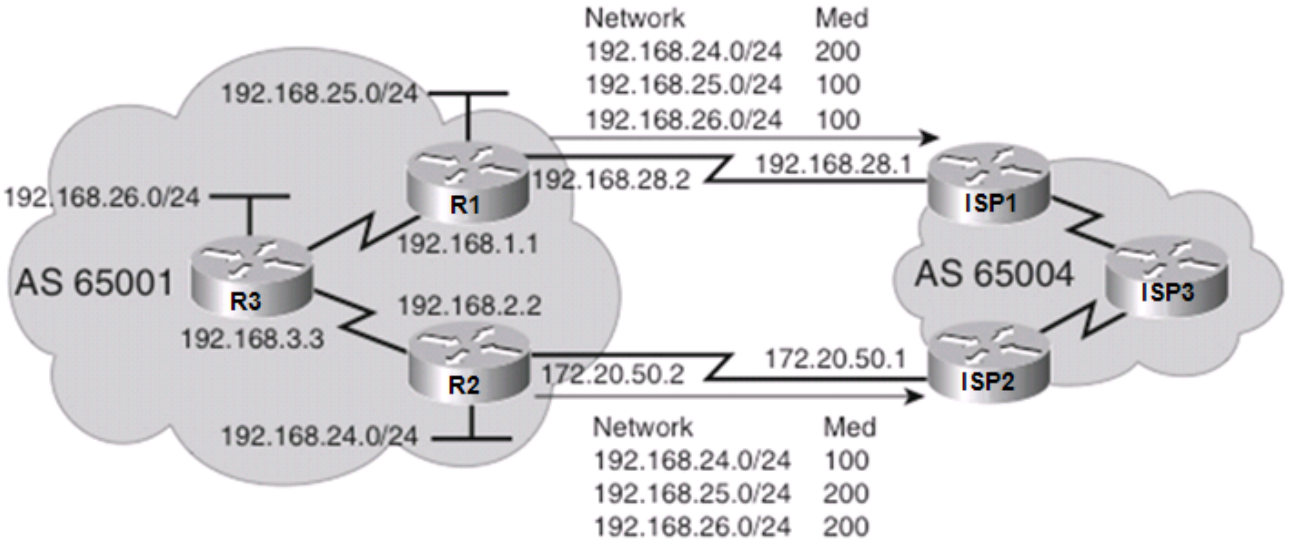
- The results are that the inbound bandwidth utilization on:
  - R1 to ISP1 link has decreased to almost nothing except for BGP routing updates.
  - R2 to ISP2 link has increased due to all returning packets from AS 65004.
  - A better solution is to have route maps configured that will make some networks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

Setting the MED with Route Maps Example



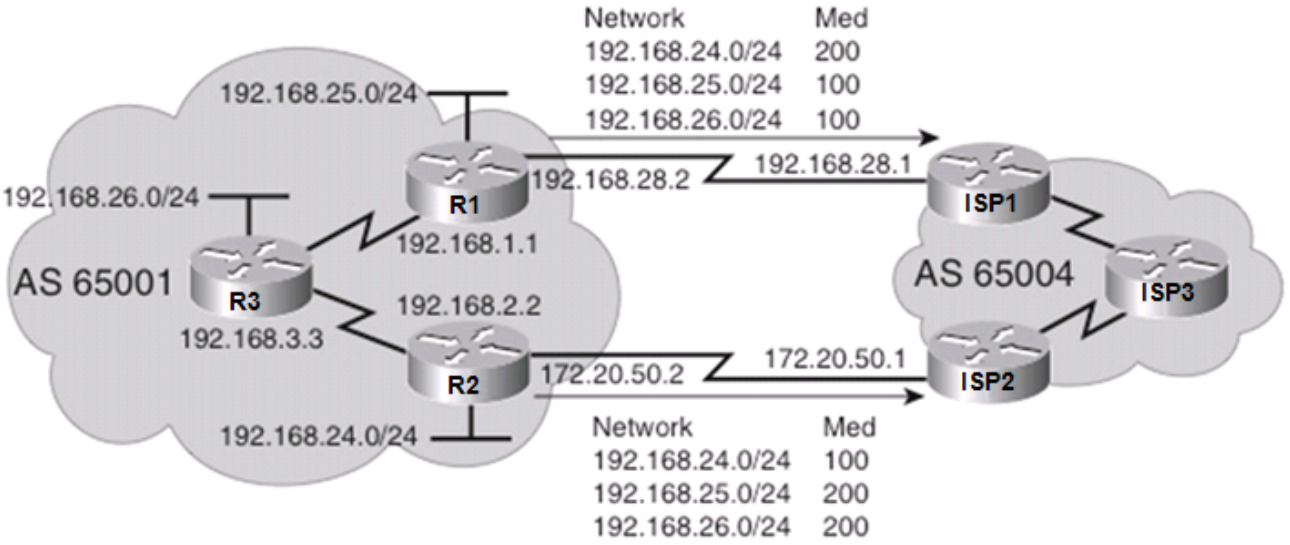

```
R1(config)# access-list 66 permit 192.168.25.0 0.0.0.255
R1(config)# access-list 66 permit 192.168.26.0 0.0.0.255
R1(config)#
R1(config)# route-map MED-65004 permit 10
R1(config-route-map)# match ip address 66
R1(config-route-map)# set metric 100
R1(config-route-map)#
R1(config-route-map)# route-map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

Setting the MED with Route Maps Example



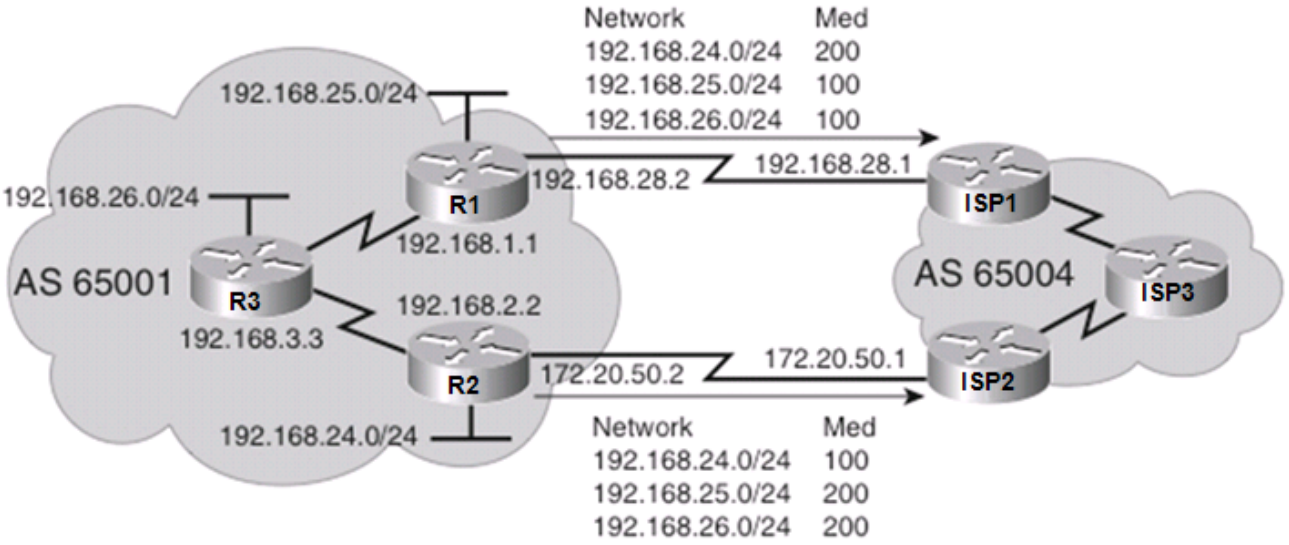

```
R1(config)# router bgp 65001
R1(config-router)# neighbor 192.168.2.2 remote-as 65001
R1(config-router)# neighbor 192.168.2.2 update-source loopback0
R1(config-router)# neighbor 192.168.3.3 remote-as 65001
R1(config-router)# neighbor 192.168.3.3 update-source loopback0
R1(config-router)# neighbor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

Setting the MED with Route Maps Example



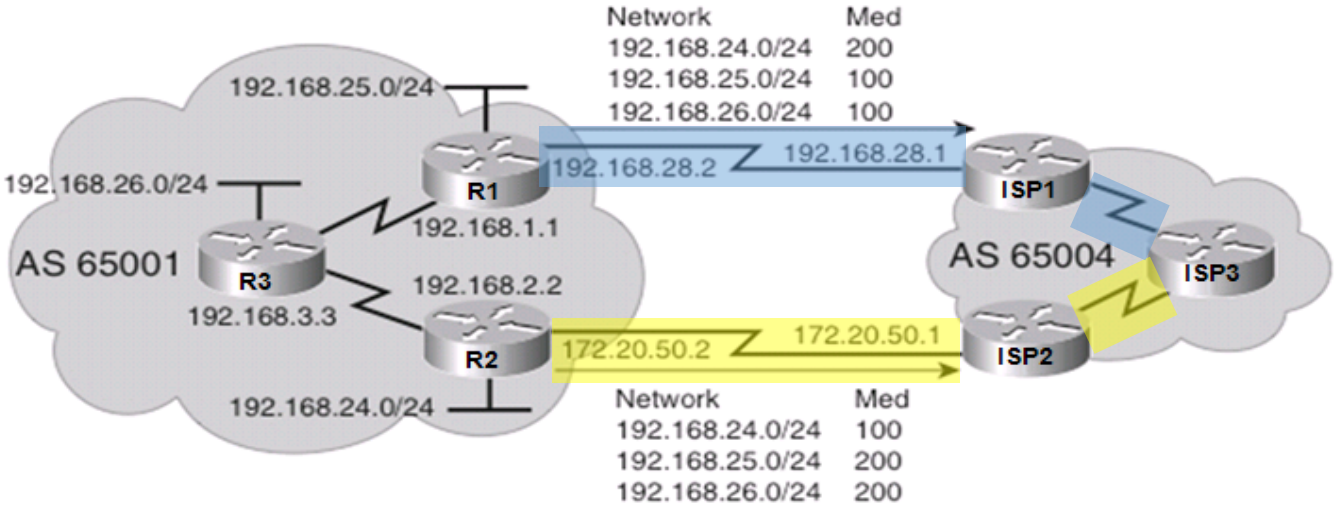

```
R2(config)# access-list 66 permit 192.168.24.0 0.0.0.255
R2(config)#
R2(config)# route-map MED-65004 permit 10
R2(config-route-map)# match ip address 66
R2(config-route-map)# set metric 100
R2(config-route-map)#
R2(config-route-map)# route-map MED-65004 permit 100
R2(config-route-map)# set metric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

Setting the MED with Route Maps Example




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

Setting the MED with Route Maps Example




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

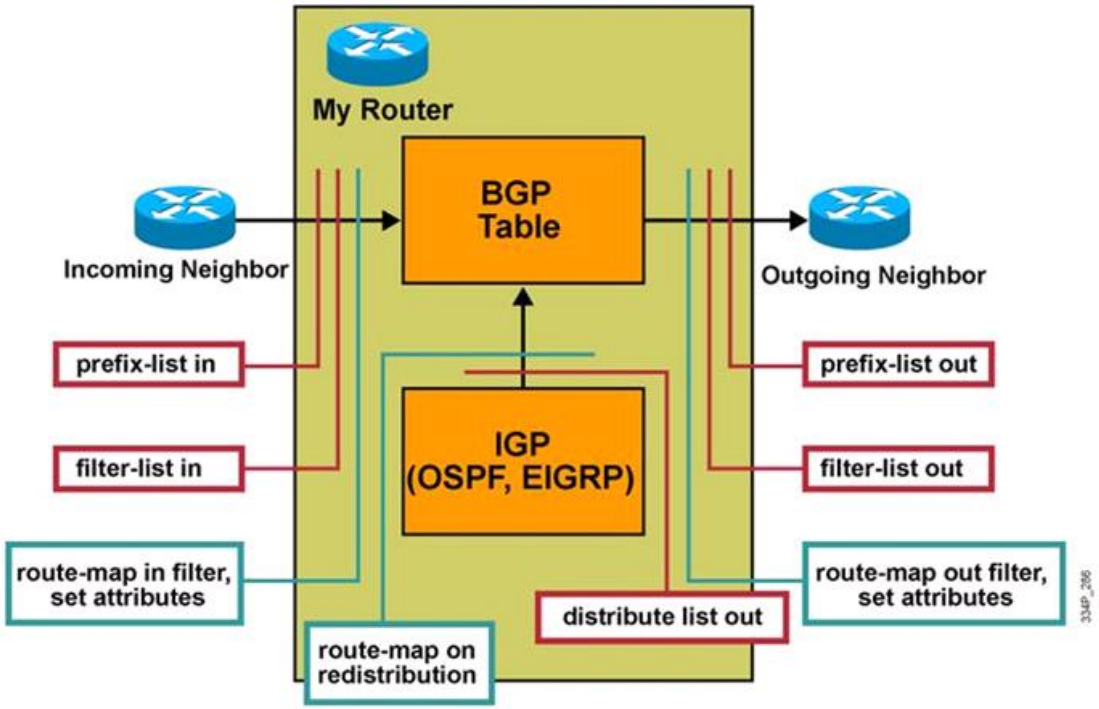
Filtering BGP Routing Updates

Filtering BGP Routing Updates

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

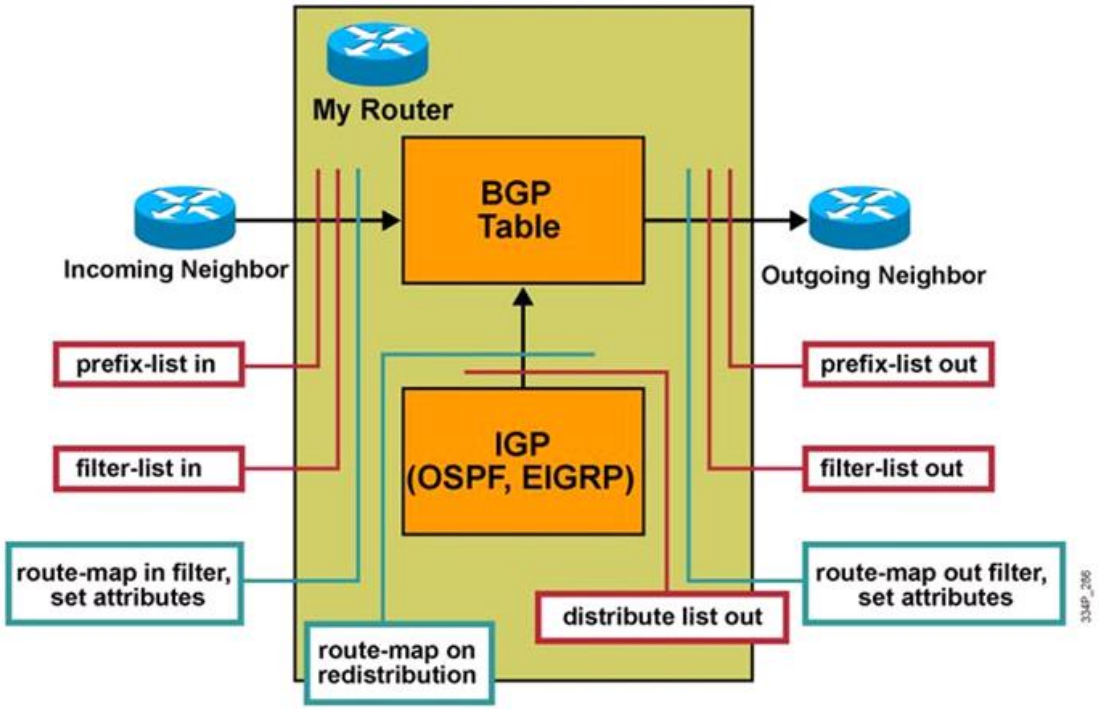
Filtering BGP Routing Updates

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

Planning BGP Filtering Using Prefix Lists

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

Configure a Prefix List

- Define a prefix list.

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

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

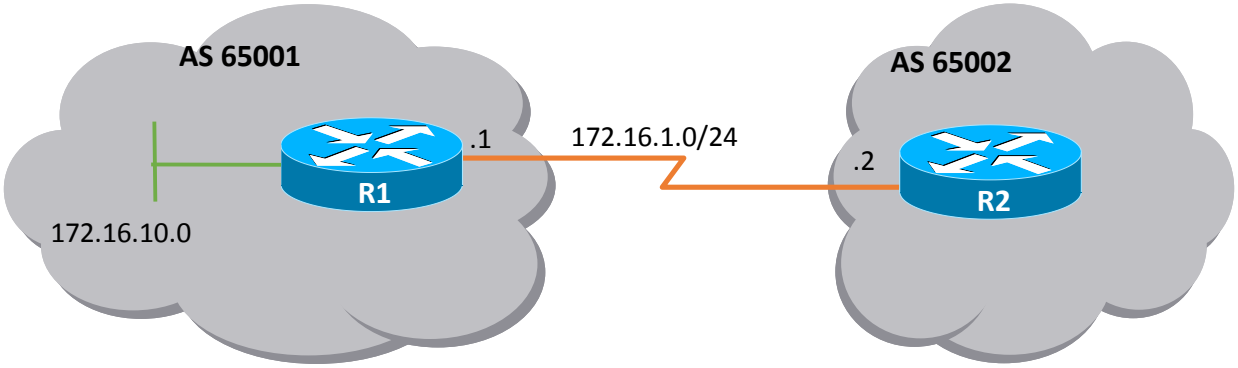
Apply a Prefix List

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

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

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

BGP Filtering Using Prefix Lists Example

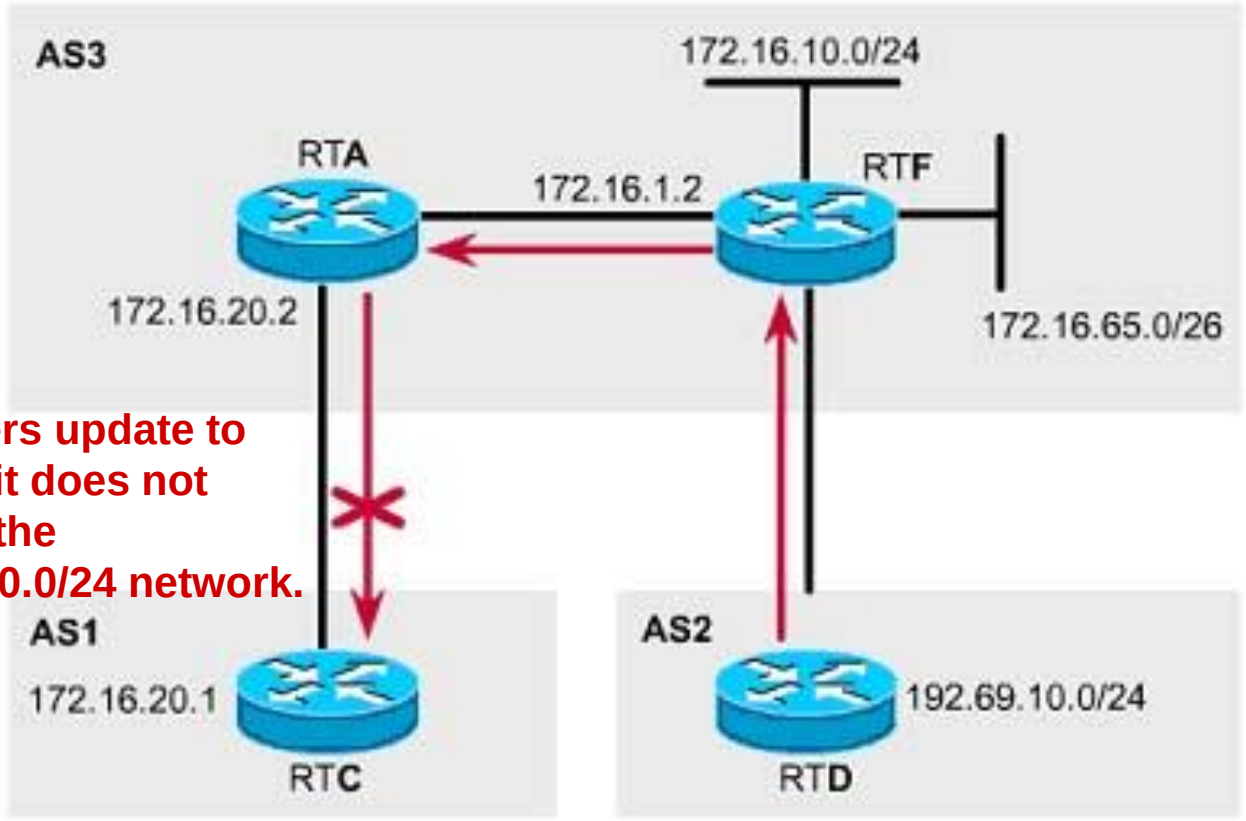



```
R2(config)# ip prefix-list ANY-8to24-NET permit 0.0.0.0/0 ge 8 le 24
R2(config)# router bgp 65001
R2(config-router)# neighbor 172.16.1.2 remote-as 65002
R2(config-router)# neighbor 172.16.1.2 prefix-list ANY-8to24-NET in
R2(config-router)# end
R2#
R2# show ip prefix-list detail ANY-8to24-NET
ip prefix-list 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)
```

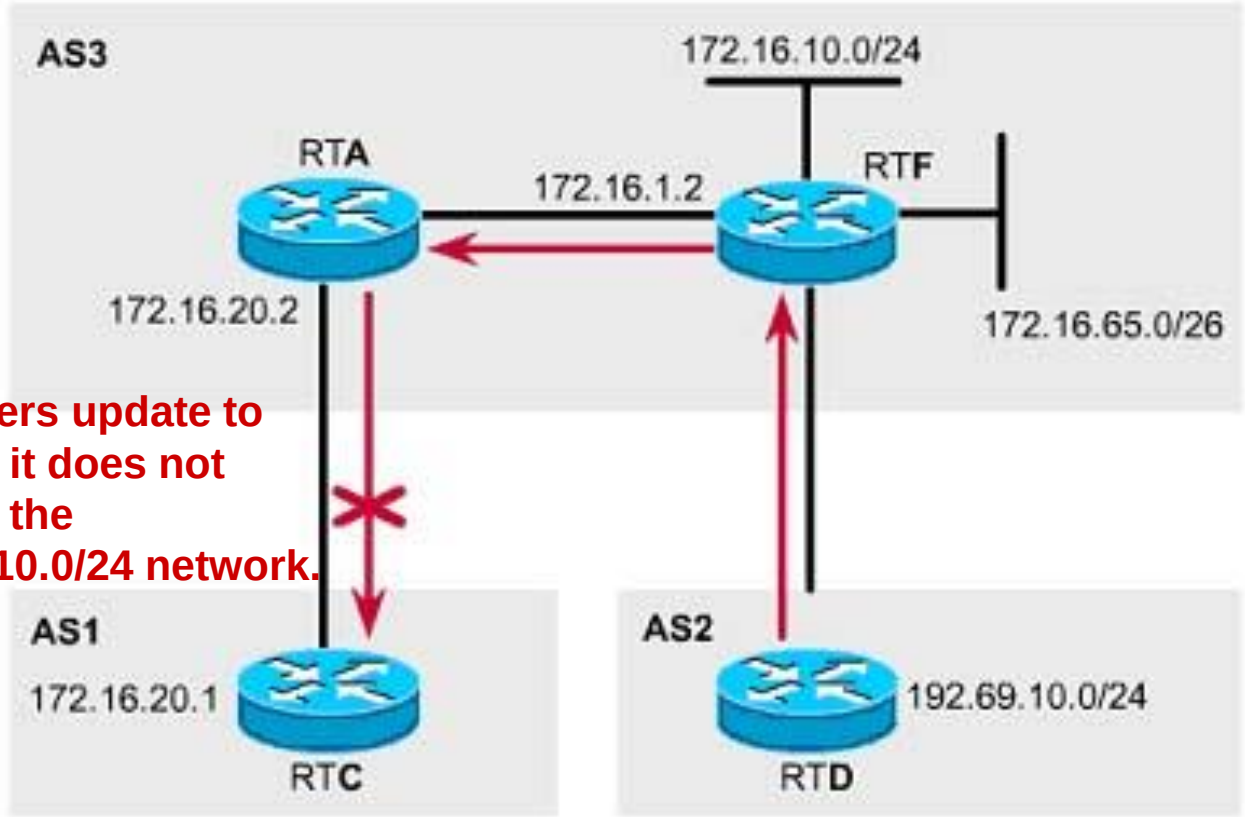
BGP Route Filtering

- 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.

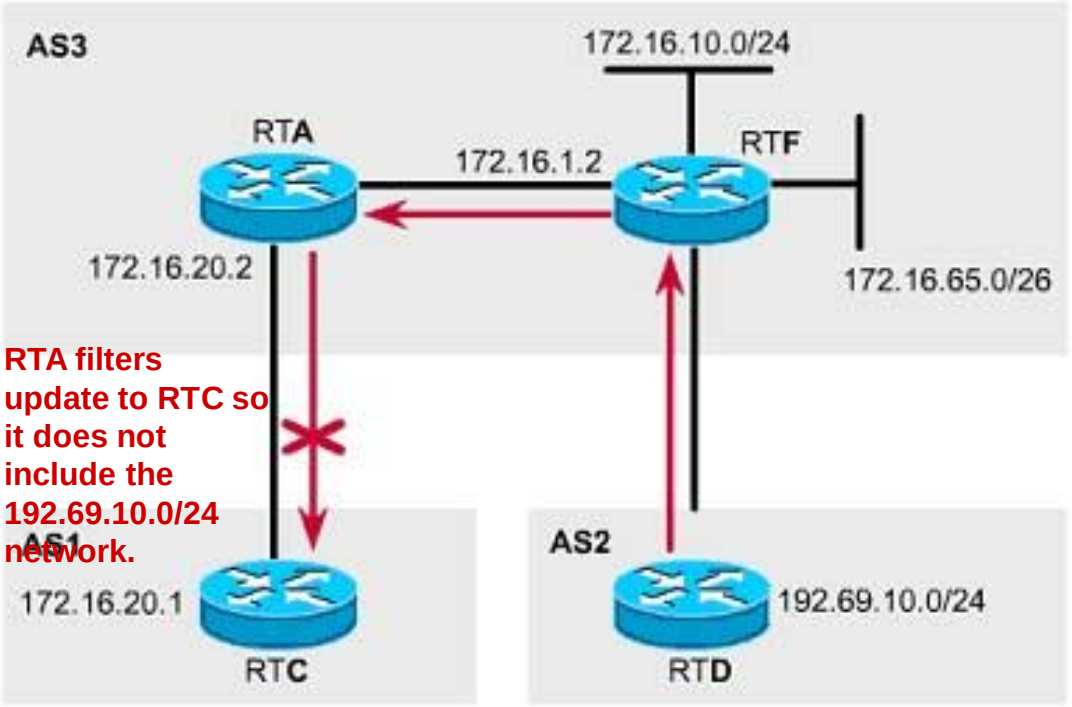
BGP Route Filtering



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

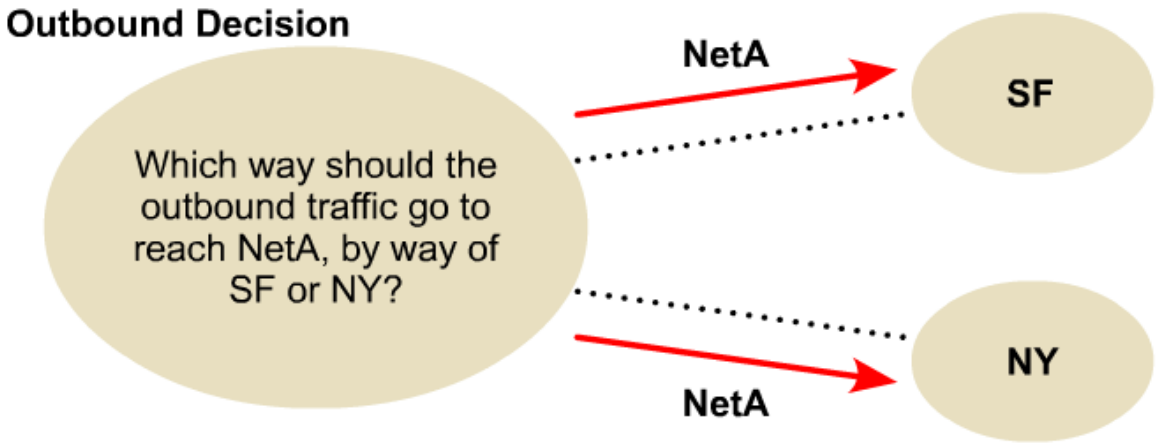
```
RTC(config)#router bgp 3
RTC(config-router)#neighbor 172.16.20.1 remote-as 1
RTC(config-router)#neighbor 172.17.1.1 remote-as 2
RTC(config-router)#network 0.0.0.0
```

- 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



- 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:

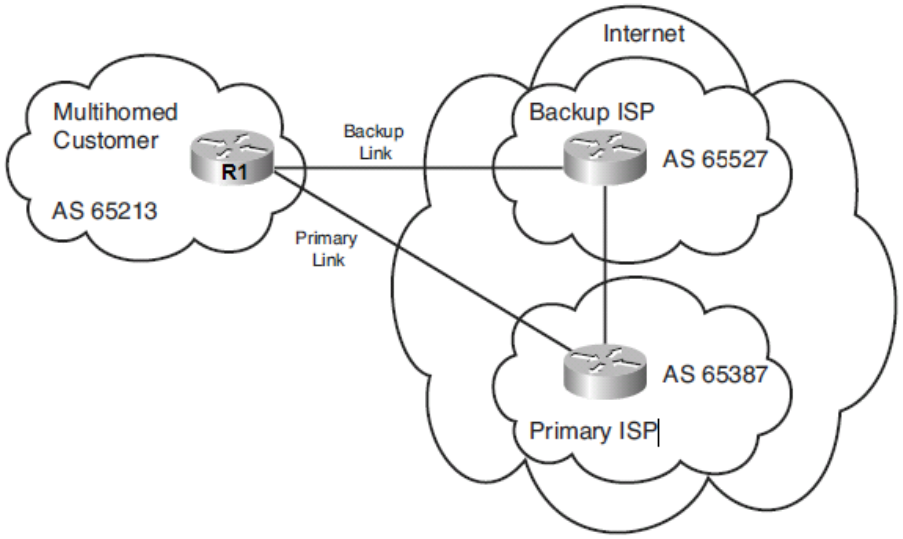
```
Router(config-router)# maximum-paths number
```

- 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

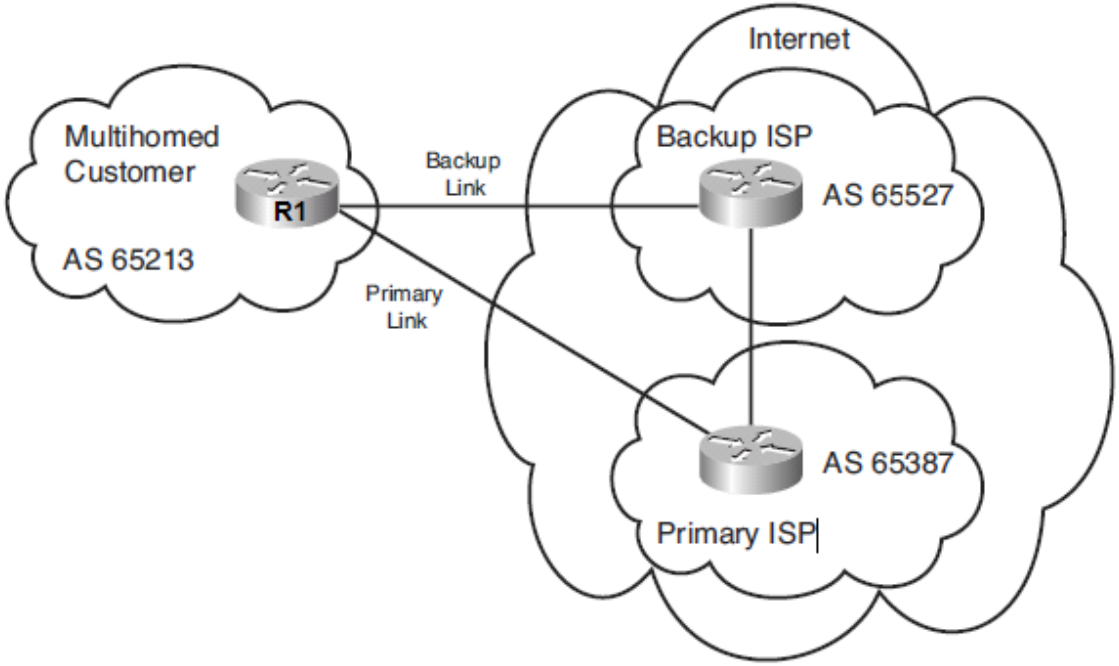
- When planning BGP filter configuration using route maps, the following steps should be documented:
  - Define the route map, including:
    - The **`match`** statements
    - The **`set`** statements
  - Configure route filtering using the route map.

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)#
```