
IP Routing

Tecnologie e Protocolli per Internet II
rev 1

Andrea Detti

University of Roma “Tor Vergata”

Electronic Engineering dept.

E-mail: andrea.detti@uniroma2.it

Some sources:

Cisco CCNA Routing and Switching ICND1 and ICND2

Routing TCP/IP Vol. II by Jeff Doyle and Jennifer Carroll

<http://faculty.valenciacollege.edu/wyousif/CCNP/Semester5/Presentations/bgp1.pp>

<http://faculty.valenciacollege.edu/wyousif/CCNP/Semester5/Presentations/bgp2.ppt>

<https://www.cs.princeton.edu/courses/archive/fall06/cos561/slides/14bgp-policy.ppt>

Routing Goal

- + **Setup IP routing tables** according to configurable administrative rules
- + Routing protocol creates the **Routing Table**
 - Possibly more routes for as same destination prefix
- + An internal algorithm choses best routes per prefix and inject this choice in the **Forwarding Table** used during packet forwarding operation

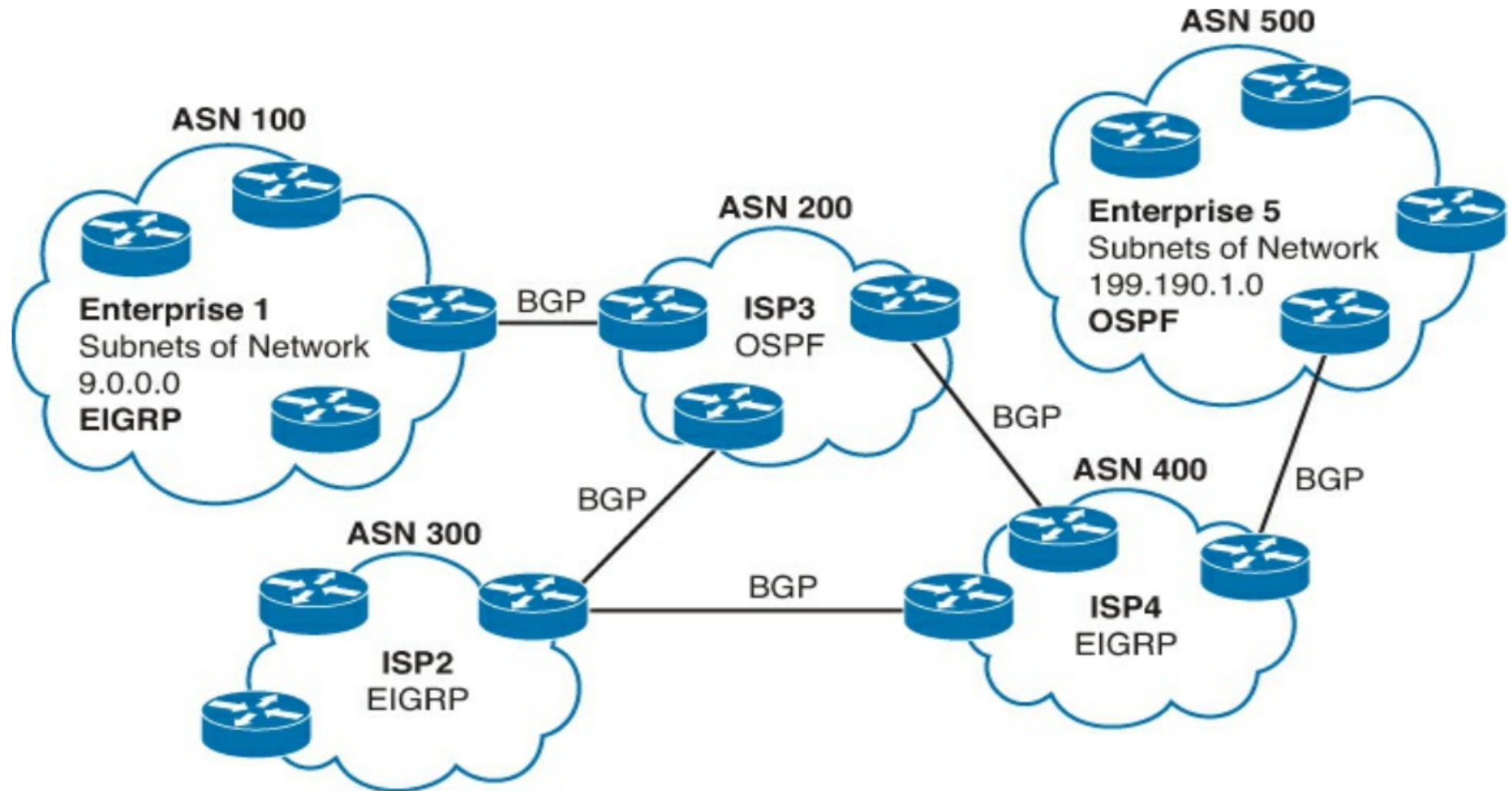
Interior and Exterior Routing Protocols

- + IP routing protocols fall into one of two major categories: **interior gateway protocols (IGP)** or **exterior gateway protocols (EGP)**
 - IGP: A routing protocol that was designed and intended for use inside a single autonomous system (AS)
 - EGP: A routing protocol that was designed and intended for use between different autonomous systems
- + An AS is a network under the administrative control of single organization
- + Each AS can be assigned a number called AS number (ASN).

Autonomous System

- + An AS is a network under the administrative control of single organization
- + Each AS is identified by a number called **AS number** (ASN)
- + Each AS has a public IP address range

ASN, EGP (BGP), IGP(EIGRP/OSPF)



IGP

- + Organizations have several options when choosing an IGP for their enterprise network (but most companies today use OSPF)
- + Two main branches of **routing protocol algorithms** exist for IGP routing protocols:
 - Distance vector (sometimes called Bellman-Ford after its creators)
 - Link-state

IGP

- + Historically, **distance vector protocols were invented first.**
- + Routing Information Protocol (RIP) was the first popularly used protocol
 - RFC 1058, June 1988
- + By the early 1990s, **distance vector protocols** showed **slow convergence** and **potential for routing loops**
- + **Link-state protocols**—Open Shortest Path First (OSPF) and Integrated Intermediate System to Intermediate System (IS-IS)— **solve these issues.**
- + Albeit they require **extra CPU and memory** on routers

Routing Protocols Compared

Feature	RIP-1	RIP-2	EIGRP	OSPF	IS-IS
Classless/sends mask in updates/ supports VLSM	No	Yes	Yes	Yes	Yes
Algorithm (DV, advanced DV, LS)	DV	DV	advanced DV	LS	LS
Supports manual summarization	No	Yes	Yes	Yes	Yes
Cisco-proprietary	No	No	Yes ¹	No	No
Routing updates are sent to a multicast IP address	No	Yes	Yes	Yes	—
Convergence	Slow	Slow	Fast	Fast	Fast

Administrative distance

- + Multiple routing protocols in a same domain
- + routing protocols might learn routes to the same subnets
- + When IOS must choose between routes learned using different routing protocols, IOS uses a concept called administrative distance.
- + Administrative distance is a number that denotes how believable an entire routing protocol is on a single router.
- + The lower the number, the better, or more believable, the routing protocol.
- + The administrative distance values are configured on a single router and are not exchanged with other routers

Administrative distance

Route Type	Administrative Distance
Connected	0
Static	1
BGP (external routes)	20
EIGRP (internal routes)	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EIGRP (external routes)	170
BGP (internal routes)	200
Unusable	255

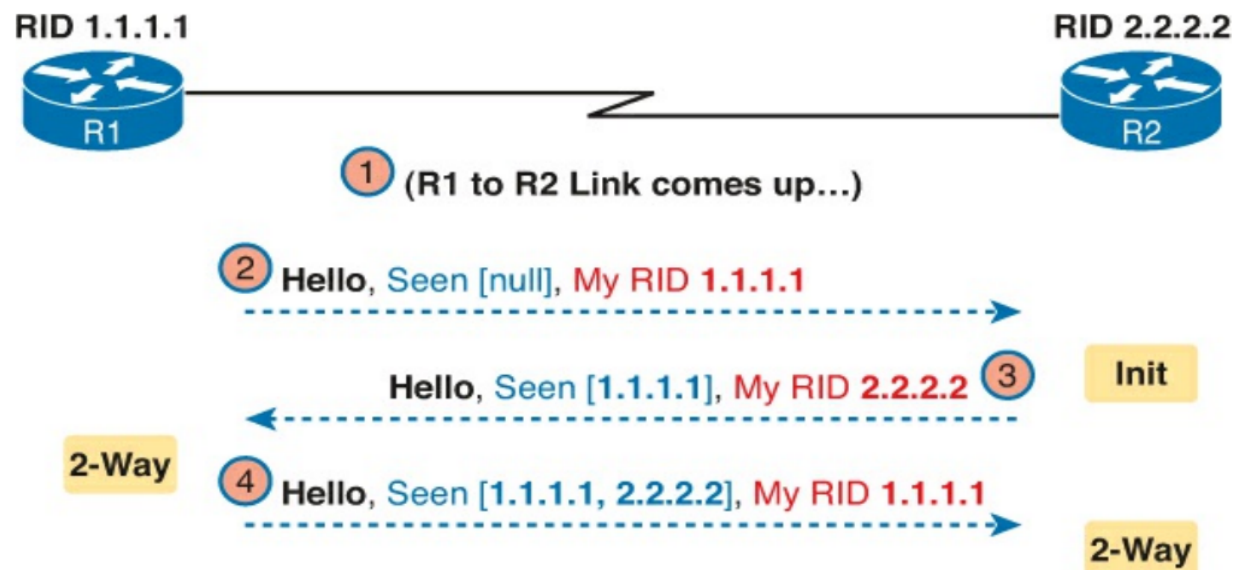
Open Short Path First

OSPF Principles

- + Build information about neighbors through **Hello** messages exchange
- + Flood this information to all other routers of the routing area
 - **Link State Announcement (LSA)**
 - Announcements are sent ONCE, and only updated if there's a change
 - Or every 30min
- + Build a topology database
 - **Link State Data Base (LSDB)**
 - DB sync problem
- + Use a "shortest path" algorithm to find best route
- + Use **multicast IP addresses** (224.0.0.5 or .6) to distribute OSPF messages

OSPF Hello and Neighbors

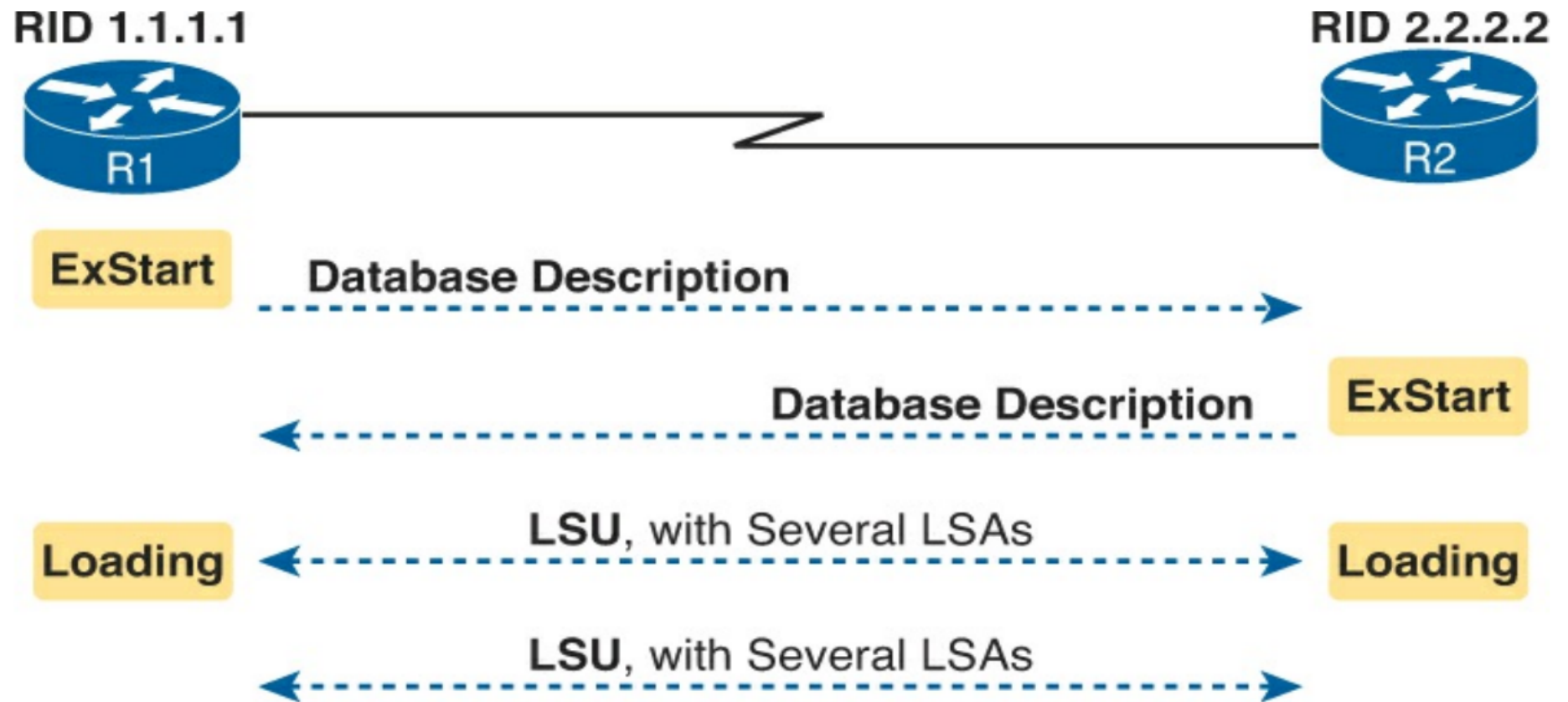
- + OSPF neighbors are routers that both use OSPF and both sit on the same data link (e.g. point-to-point link or LAN).
- + To agree to become neighbors by sending OSPF Hello messages, which include OSPF parameters
- + If the compatible parameters (2-way state), they form a neighbor relationship and begin to exchange their LSDBs
- + Router ID: A 32-bit number assigned to each router running the OSPF protocol. This number uniquely identifies the router within an Autonomous System.



Initial LSDB Sync

- + After two routers decide to exchange databases, first, they tell each other a list of LSAs in their respective databases (not all the details of the LSAs) by using **OSPF Database Description**, or DD, packet
- + Then each router requests to the other the details of missing LSA through **Link-State Request**, or LSR, packets
- + The neighbor provides missing LSAs within a **Link-State Update (LSU) packet**
- + The link-state advertisements (LSA) are not packets, but rather data structures, which sit inside the LSDB and describe the topology.

Initial LSDB Sync

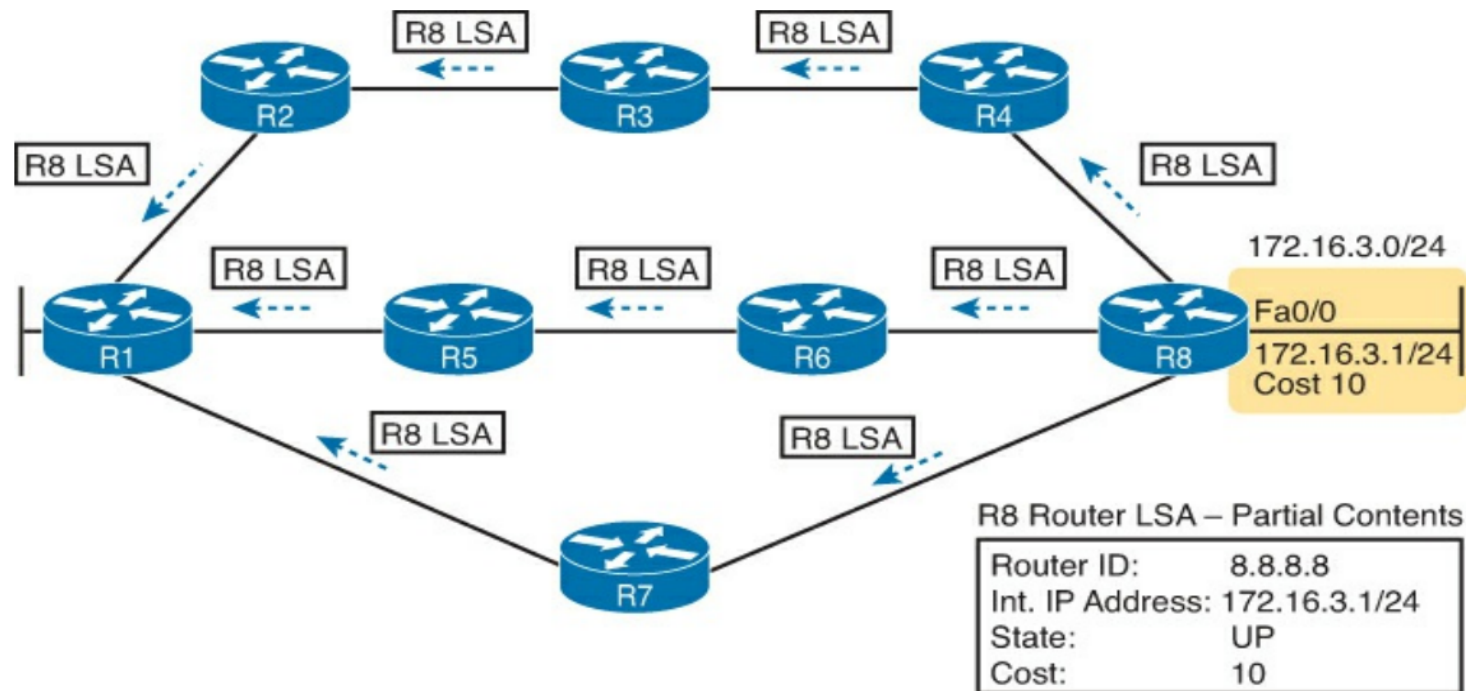


Maintain the neighbor relationship

- + Once neighbors reach a **full state**, they have done all the initial work to exchange OSPF information
- + However, neighbors still have to do some small ongoing tasks to maintain the neighbor relationship.
- + Routers monitor each neighbor relationship using Hello messages and two related timers.
 - Routers send Hellos every **Hello Interval** to each neighbor.
 - Each router expects to receive a Hello from each neighbor based on the Hello Interval, so if a neighbor is silent for the length of the **Dead Interval** (by default 4 times as long as the Hello Interval), the loss of Hellos means that the neighbor has failed.

LSA update

- + When something changes the routers must flood the changed LSAs to each neighbor so that the neighbor can change its LSDB
- + Reflood unchanged LSAs as their lifetime expires (default 30 minutes)

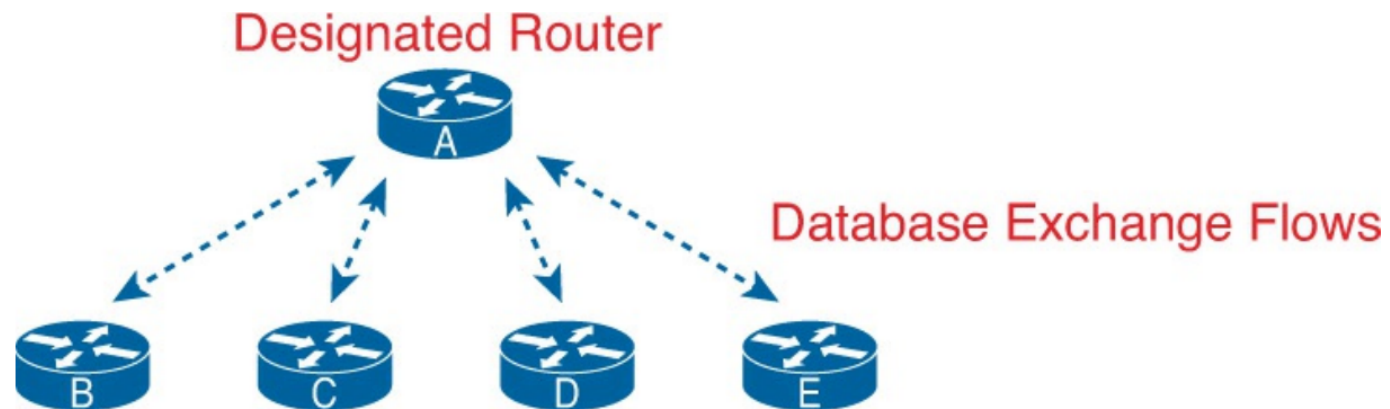


LSA types

- + One **router LSA** for each router in the area
- + One **network LSA** for each network that has a DR plus one neighbor of the DR
- + One **summary LSA** (prefix/mask) for each subnet ID that exists in a different area

OSPF on Ethernet links

- + OSPF behaves differently on some types of interfaces, particularly comparing point-to-point and Ethernet links.
- + On Ethernet links, OSPF elects one of the routers on the same subnet to act as the **designated router** (DR).
- + The database exchange process on an Ethernet link happens between the DR and each of the other routers, with the DR making sure that all the other routers get a copy of each LSA through LSA Ack
- + A Backup Designated Router (BRD) is also elected
- + Routers that not are neither DR or BDR are called DROthers



OSPF on Ethernet links

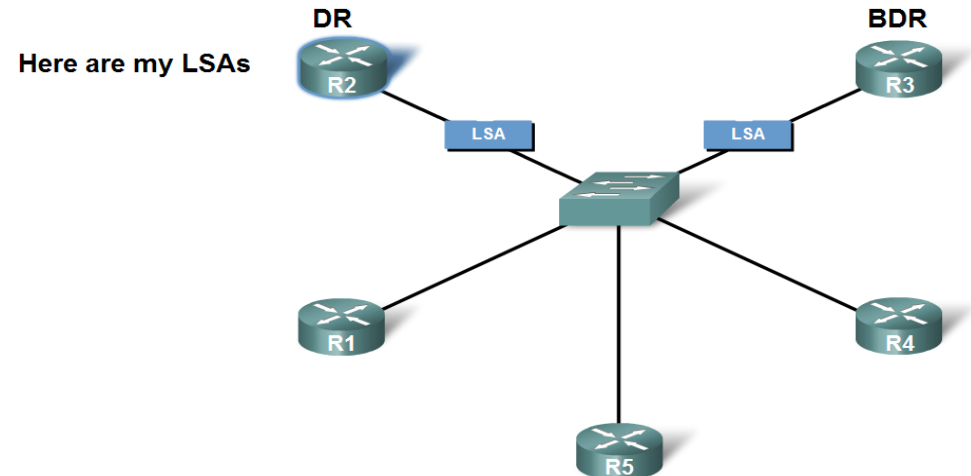
+ Use of DR devised to reduce traffic

- the Designated Router multicasts its Link State Update Packets to the address AllSPFRouters (224.0.0.5), rather than sending separate packets over each adjacency
- consequence: lots of bandwidth consumed and chaotic traffic in a broadcast means

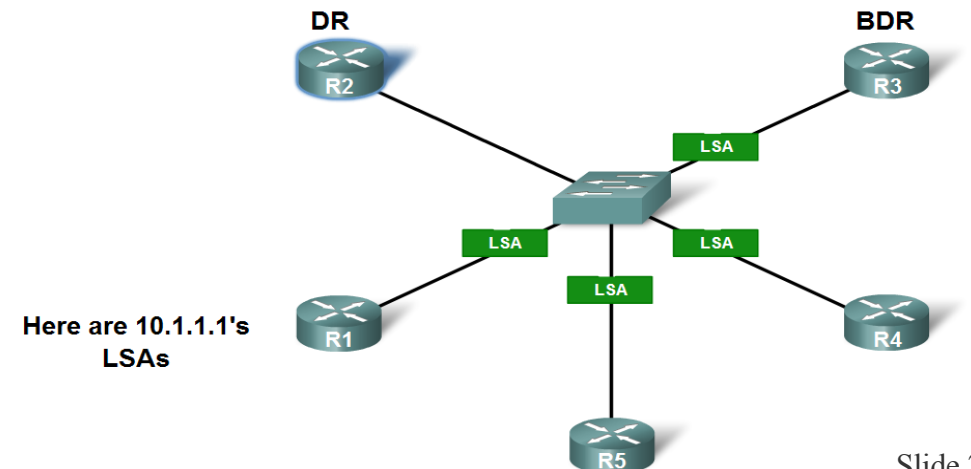
+ Sending & Receiving LSA

- DRothers send LSAs via multicast 224.0.0.6 to DR & BDR
- DR forward LSA via multicast address 224.0.0.5 to all other routers
- LSack sent by receiving DRothers only to DR, by BDR to the originating DRothers, implicit LSack from DR to originating DRothers since it relays the same LSA

Adjacencies are formed with DR and BDR only.
LSAs are sent to the DR. BDR listens.

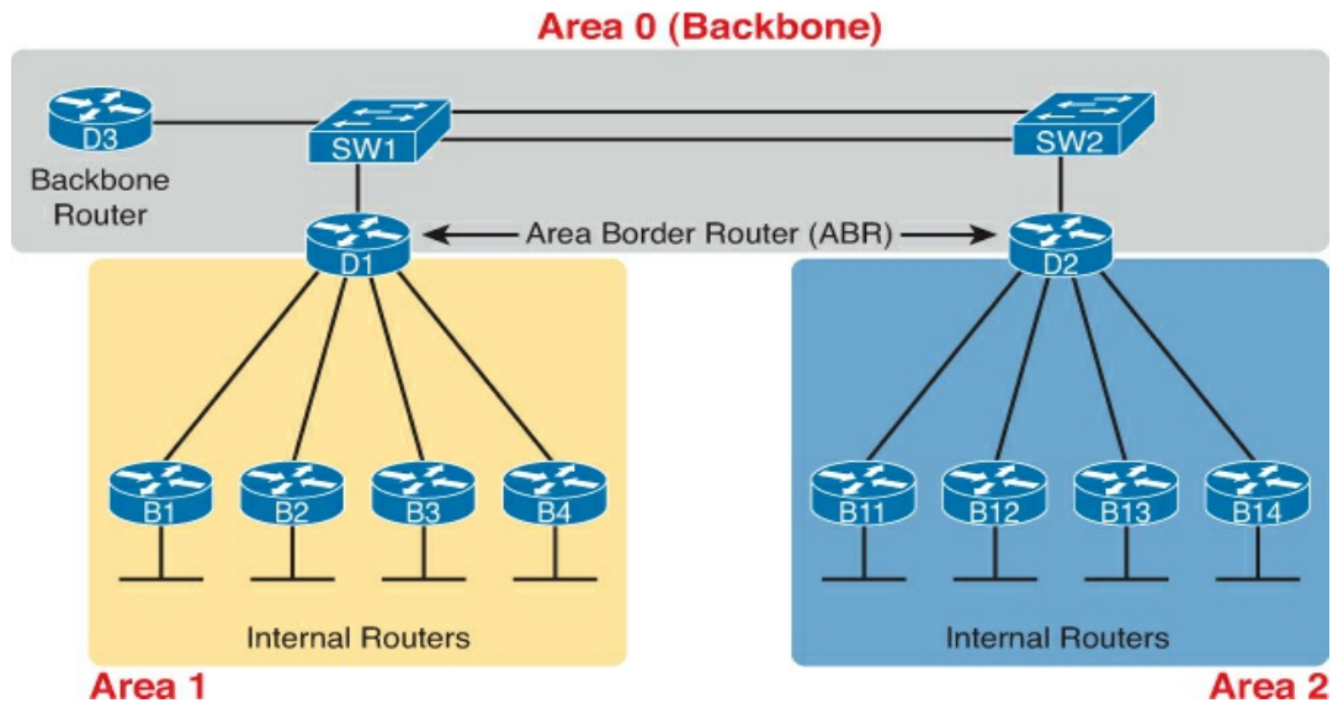


DR sends out any LSAs to all other routers.



Scaling OSPF: OSPF Areas

- + Group of contiguous nodes/networks
- + Per area topology DB
 - Invisible outside the area
 - Reduces routing traffic
- + Backbone Area is contiguous
 - All others areas must connect to the backbone



Scaling OSPF: OSPF Areas

- + Simple design rules
- + Put all interfaces connected to the same subnet inside the same area.
- + Some routers may be internal to an area, with all interfaces assigned to that single area.
- + Some routers may be ABRs, because some interfaces connect to the backbone area, and some connect to non-backbone areas.
- + All non-backbone areas must connect to the backbone area (area 0) by having at least one ABR connected to both the backbone area and the non-backbone area.

Scaling OSPF: OSPF Areas

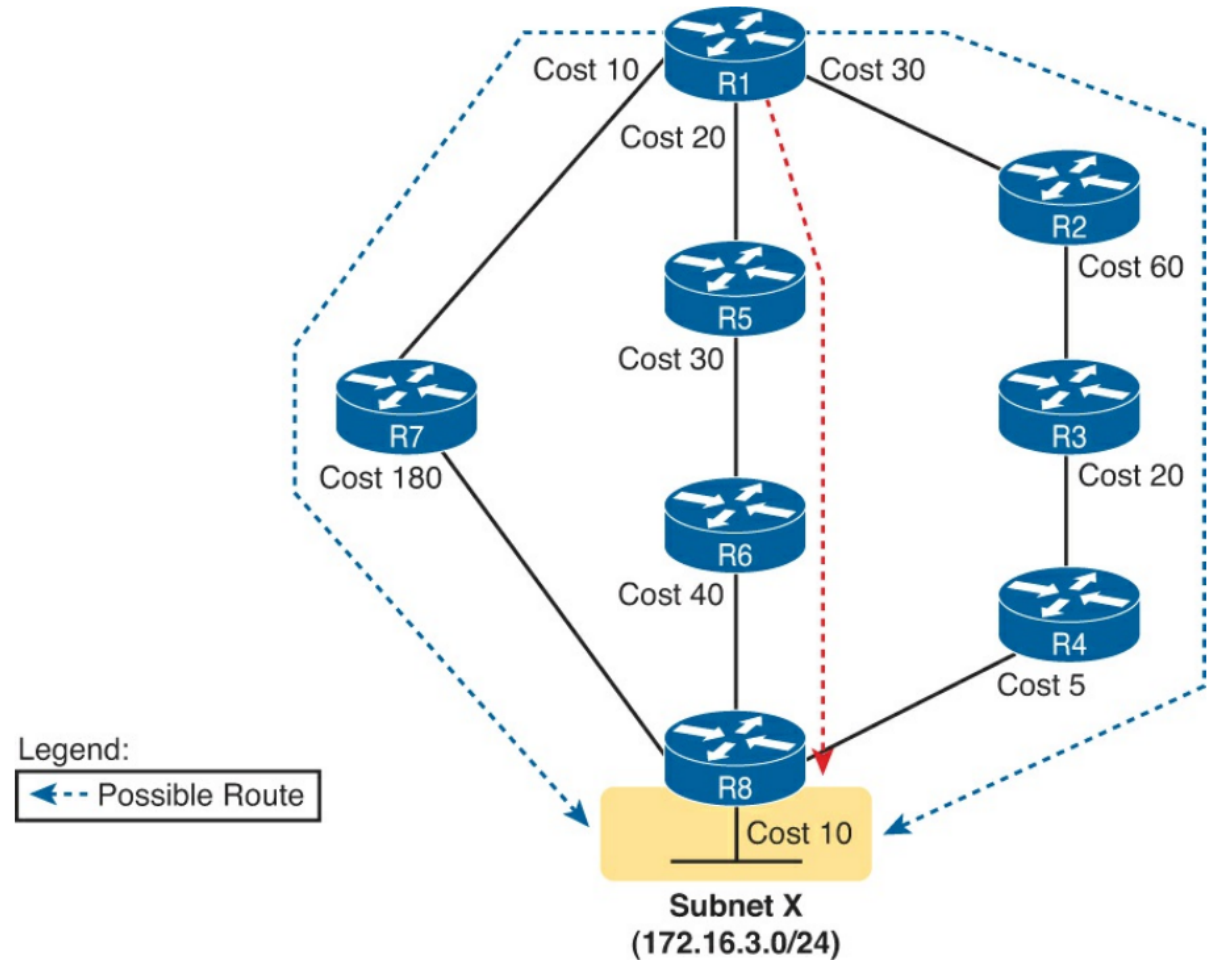
Term	Description
Area Border Router (ABR)	An OSPF router with interfaces connected to the backbone area and to at least one other area
Backbone router	A router in one area (the backbone area)
Internal router	A router in one area (not the backbone area)
Area	A set of routers and links that share the same detailed LSDB information, but not with routers in other areas, for better efficiency
Backbone area	A special OSPF area to which all other areas must connect—area 0
Intra-area route	A route to a subnet inside the same area as the router
Interarea route	A route to a subnet in an area of which the router is not a part

LSA types

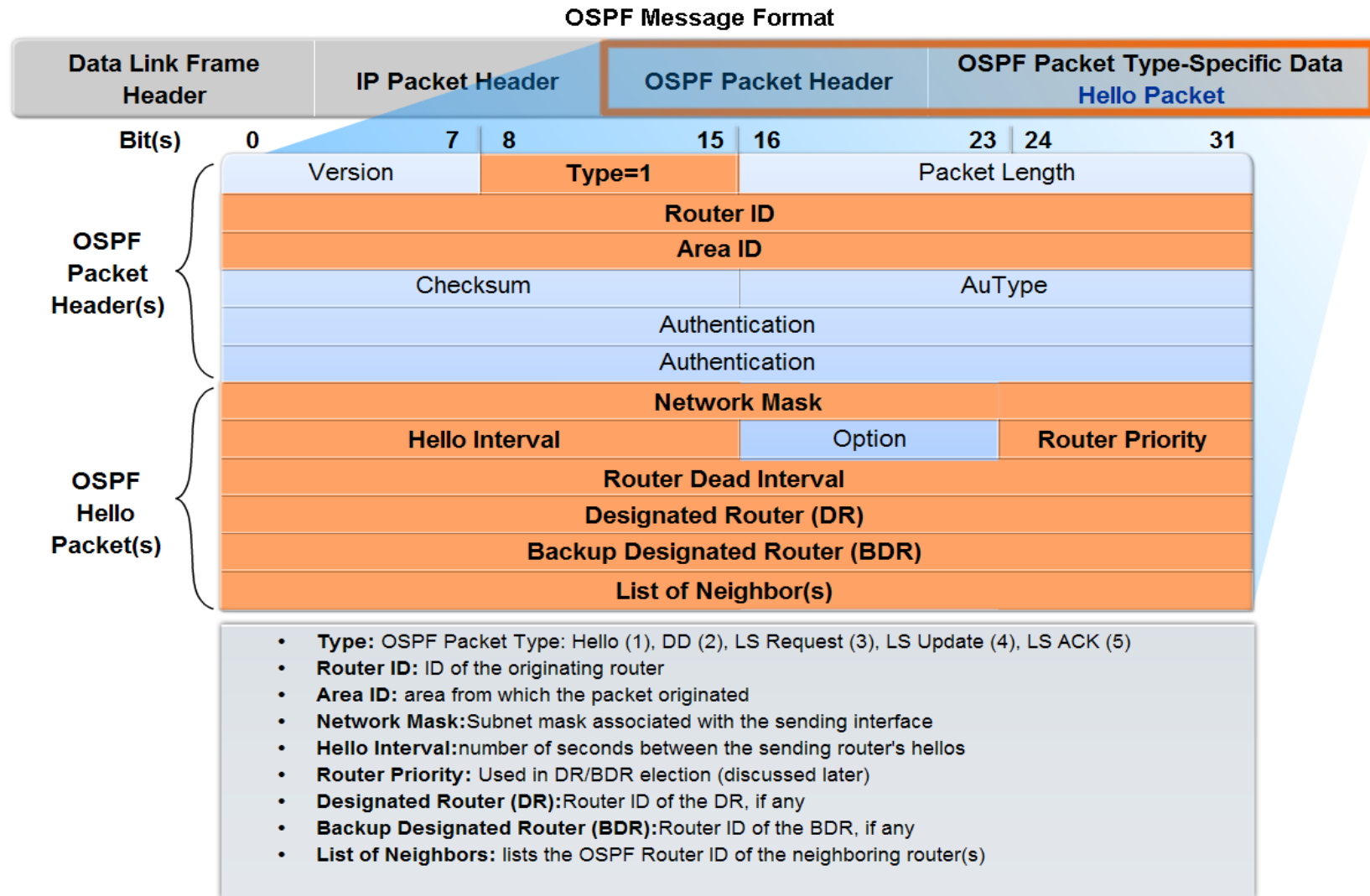
Name	Number	Primary Purpose	Contents of LSA
Router LSA	1	Describe Router	RID, interfaces, IP address/mask, current interface state (status)
Network	2	Describe network that has a DR	DR and BDR IP addresses, subnet ID, mask
Summary	3	Describe a subnet in another area	Subnet ID, mask, RID of ABR that advertises the LSA

Shortest Path First

- + SPF select the path with lowest cost
- + Cost is associated to an interface
- + Equal cost == minimum hops



OSPF message format



OSPF Packet types

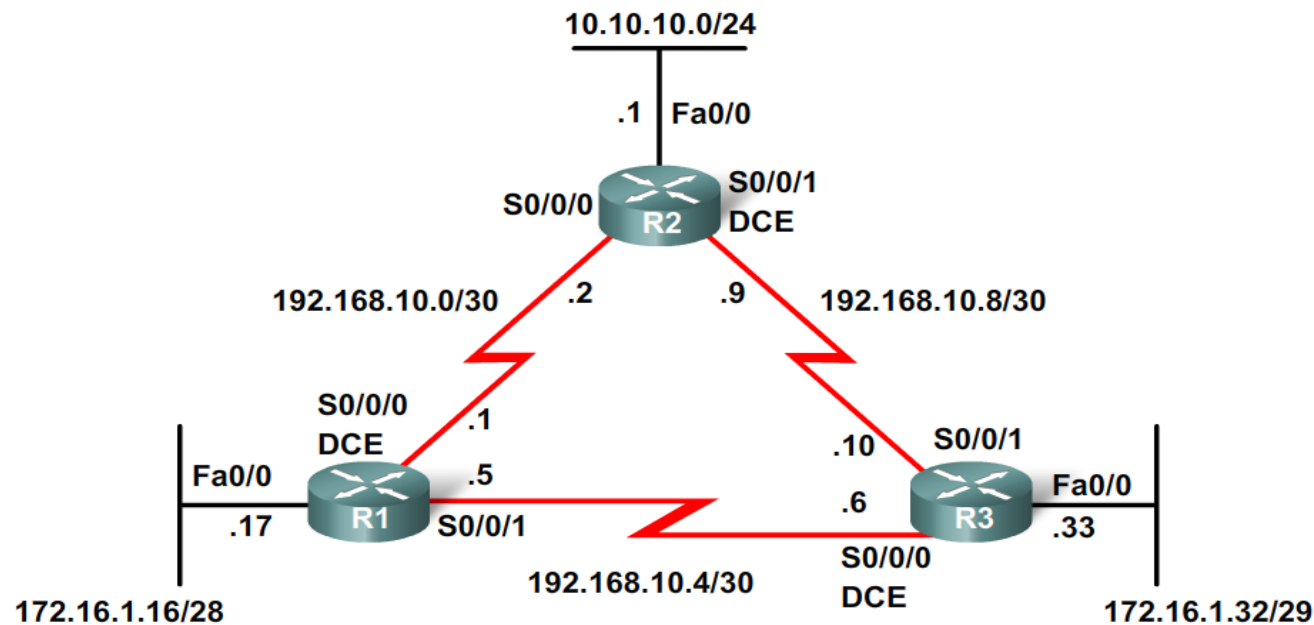
Type	Packet Name	Description
1	Hello	Discovers neighbors and builds adjacencies between them
2	Database Description (DBD)	Checks for database synchronization between routers
3	Link-State Request (LSR)	Requests specific link-state records from router to router
4	Link-State Update (LSU)	Sends specifically requested link-state records
5	Link-State Acknowledgement (LSAck)	Acknowledges the other packet types

BASIC OSPF CISCO CONFIGURATION

Lab Topology

+ Topology used

- Discontiguous IP addressing scheme
- Since OSPF is a classless routing protocol the subnet mask is configured in



The router ospf command

- + To enable OSPF on a router use the following command
 - R1(config)#**router ospf** *process-id*
 - Process id
 - A locally significant number between **1** and **65535**

```
R1 (config) #router ospf 1  
R1 (config-router) #
```

```
R2 (config) #router ospf 1  
R2 (config-router) #
```

```
R3 (config) #router ospf 1  
R3 (config-router) #
```

OSPF network command

- + Used to identify the interface **where** run OSPF
- + *network address + wildcard mask*
 - **wildcard mask** - the inverse of the subnet mask
- + area-id refers to the OSPF area.

```
R1 (config) #router ospf 1  
R1 (config-router) #network 172.16.1.16 0.0.0.15 area 0  
R1 (config-router) #network 192.168.10.0 0.0.0.3 area 0  
R1 (config-router) #network 192.168.10.4 0.0.0.3 area 0
```

```
R2 (config) #router ospf 1  
R2 (config-router) #network 10.10.10.0 0.0.0.255 area 0  
R2 (config-router) #network 192.168.10.0 0.0.0.3 area 0  
R2 (config-router) #network 192.168.10.8 0.0.0.3 area 0
```

OSPF Router ID

- This is an IP address used to identify a router
- 3 criteria for deriving the router ID
 - Use IP address configured with OSPF *router-id* command
 - Takes precedence over loopback and physical interface addresses
 - If router-id command not used then router chooses highest IP address of any loopback interfaces
 - If no loopback interfaces are configured then the highest IP address on any active interface is used

OSPF Router ID

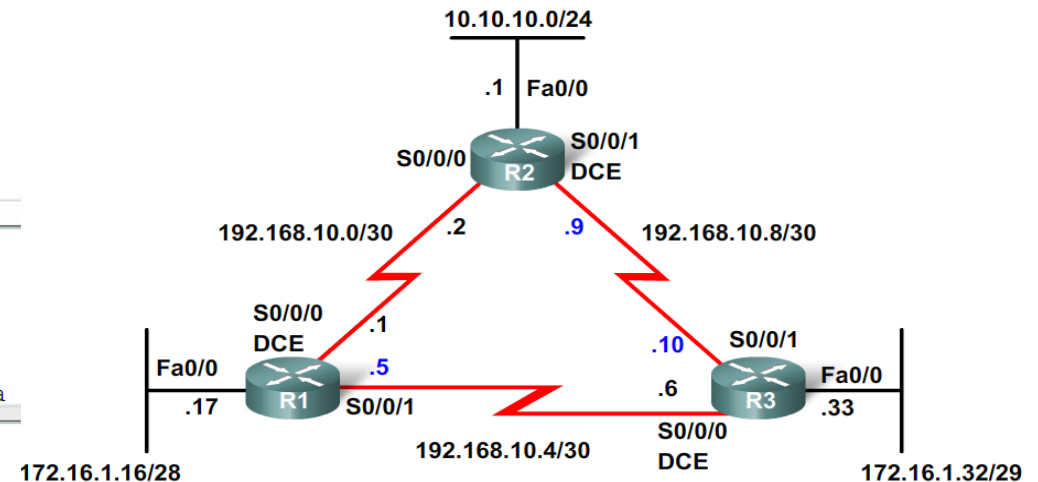
+ Commands used to verify current router ID

- Show ip protocols
- Show ip ospf
- Show ip ospf interface

```
R1#show ip protocols
Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 192.168.10.5
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa

R2#show ip protocols
Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 192.168.10.9
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa

R3#show ip protocols
Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 192.168.10.10
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
```



Router ID & Loopback addresses

```
R1 (config) #interface loopback 0  
R1 (config-if) #ip add 10.1.1.1 255.255.255.255
```

+ Router ID & Loopback addresses

- Highest loopback address will be used as router ID if router-id command isn't used
- Advantage of using loopback address
- The loopback interface cannot fail → OSPF stability

+ The OSPF router-id command

- Introduced in IOS 12.0
- Command syntax
 - Router(config)#router ospf process-id
 - Router(config-router)#router-id ip-address

+ Modifying the Router ID

- Use the command Router#clear ip ospf process

Verifying OSPF

- + Use the **show ip ospf** command to verify & trouble shoot OSPF networks
 - Command will display the following:
 - Neighbor adjacency
 - No adjacency indicated by -
 - **Neighboring router's Router ID is not displayed**
 - **A state of full is not displayed**
 - Consequence of no adjacency-
 - **No link state information exchanged**
 - **Inaccurate SPF trees & routing tables**

```
R1#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.3.3.3	1	FULL/ -	00:00:30	192.168.10.6	Serial0/0/1
10.2.2.2	1	FULL/ -	00:00:33	192.168.10.2	Serial0/0/0

Verifying OSPF - Additional Commands

Command	Description
Show ip protocols	Displays OSPF process ID, router ID, networks router is advertising & administrative distance
Show ip ospf	Displays OSPF process ID, router ID, OSPF area information & the last time SPF algorithm calculated
Show ip ospf interface	Displays hello interval and dead interval

Examining the routing table

- + Use the show ip route command to display the routing table
 - -An “O” at the beginning of a route indicates that the router source is OSPF
 - -Note OSPF does not automatically summarize at major network boundaries

```
R1#show ip route

Codes: <some code output omitted>
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

Gateway of last resort is not set

    192.168.10.0/30 is subnetted, 3 subnets
C       192.168.10.0 is directly connected, Serial0/0/0
C       192.168.10.4 is directly connected, Serial0/0/1
O       192.168.10.8 [110/128] via 192.168.10.2, 14:27:57, Serial0/0/0
    172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O       172.16.1.32/29 [110/65] via 192.168.10.6, 14:27:57, Serial0/0/1
C       172.16.1.16/28 is directly connected, FastEthernet0/0
    10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
O       10.10.10.0/24 [110/65] via 192.168.10.2, 14:27:57, Serial0/0/0
C       10.1.1.1/32 is directly connected, Loopback0
```

OSPF Metric

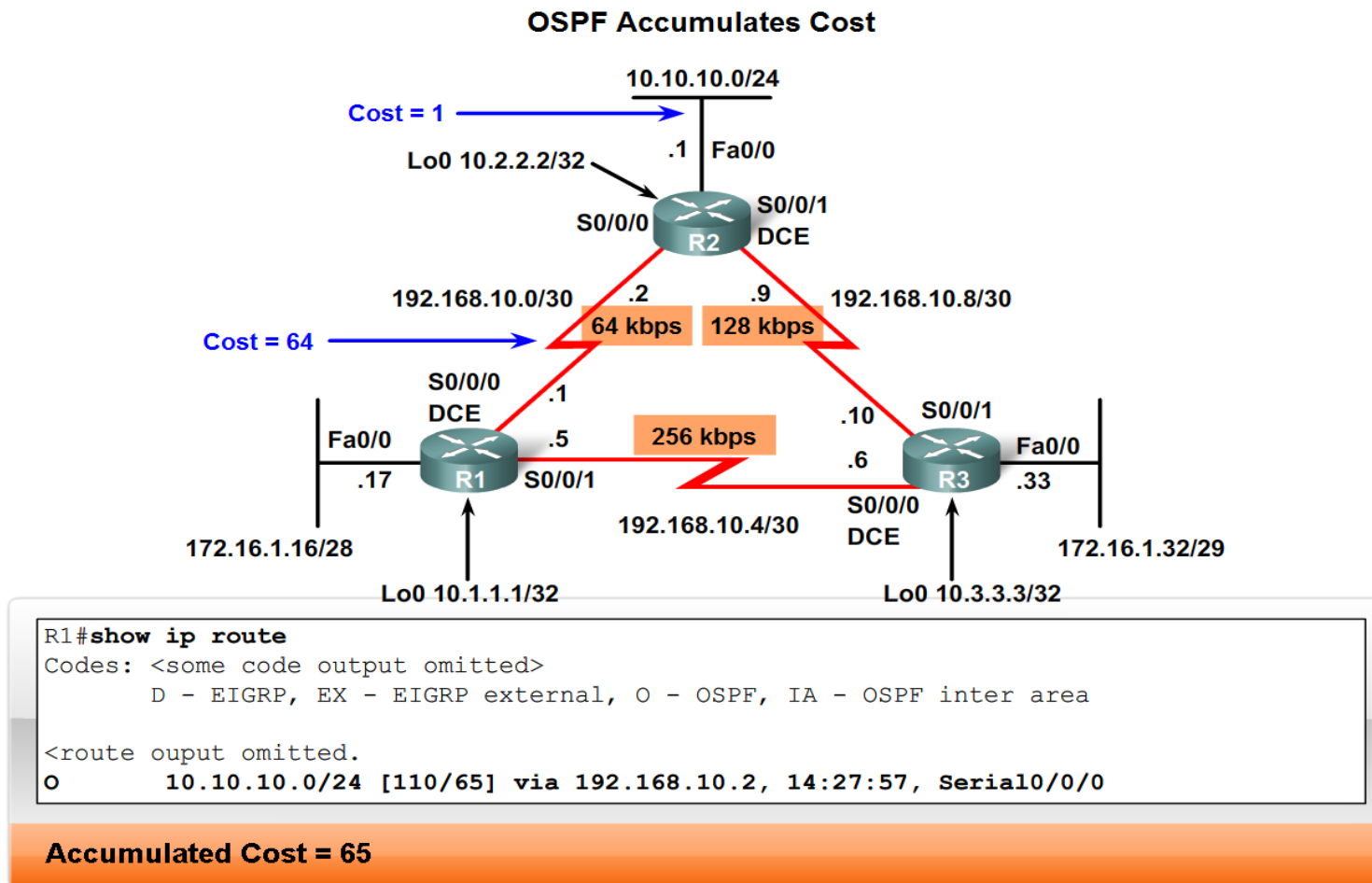
- + OSPF uses cost as the metric for determining the best route
 - The best route will have the lowest cost
 - Cost is based on bandwidth of an interface
 - Cost is calculated using the formula $10^8 / \text{bandwidth}$
 - Reference bandwidth
 - defaults to 100Mbps
 - can be modified using
 - auto-cost reference-bandwidth command

Interface Type	$10^8 / \text{bps} = \text{Cost}$
Fast Ethernet and faster	$10^8 / 100,000,000 \text{ bps} = 1$
Ethernet	$10^8 / 10,000,000 \text{ bps} = 10$
E1	$10^8 / 2,048,000 \text{ bps} = 48$
T1	$10^8 / 1,544,000 \text{ bps} = 64$
128 kbps	$10^8 / 128,000 \text{ bps} = 781$
64 kbps	$10^8 / 64,000 \text{ bps} = 1562$
56 kbps	$10^8 / 56,000 \text{ bps} = 1785$

OSPF Metric

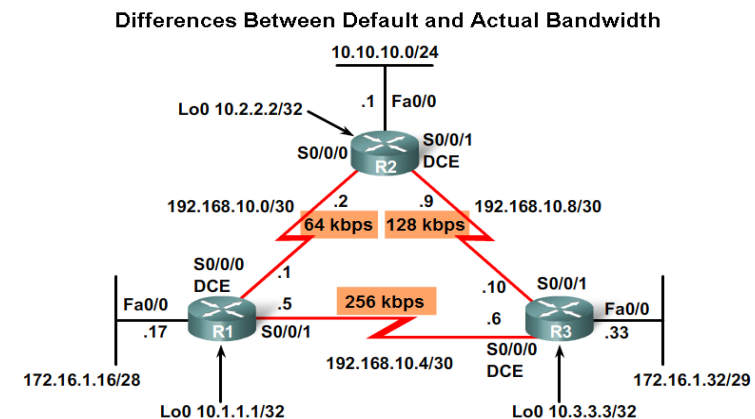
+ COST of an OSPF route

- Is the accumulated value from one router to the next



OSPF Metric

- + Usually the actual speed of a link is **different** than the default bandwidth
 - This makes it imperative that the bandwidth value reflects link's actual speed
 - Reason: so routing table has best path information
- + The **show interface** command will display interface's bandwidth
 - -Most serial link default to 1.544Mbps



```
R1#show interface serial 0/0/0
Serial0/0/0 is up, line protocol is up
Hardware is GT96K Serial
Description: Link to R2
Internet address is 192.168.10.1/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
reliability 255/255, txload 1/255, rxload 1/255
```

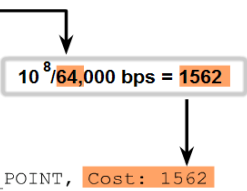
Default Bandwidth = 1544 kbps
Actual Bandwidth = 64 kbps

Basic OSPF Configuration

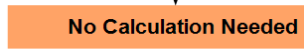
Modifying the Cost of a link

- + Both sides of a serial link should be configured with the same bandwidth
 - Commands used to modify bandwidth value
 - **Bandwidth** command
 - Example: Router(config-if)#bandwidthbandwidth-kbps
 - **ip ospf cost** command – allows you to directly specify interface cost
 - R1(config)#interface serial 0/0/0
 - R1(config-if)#ip ospf cost 1562

```
R1(config)#inter serial 0/0/0
R1(config-if)#bandwidth 64
R1(config-if)#inter serial 0/0/1
R1(config-if)#bandwidth 256
R1(config-if)#end
R1#show ip ospf interface serial 0/0/0
Serial0/0 is up, line protocol is up
Internet Address 192.168.10.1/30, Area 0
Process ID 1, Router ID 10.1.1.1, Network Type POINT_TO_POINT, Cost: 1562
Transmit Delay is 1 sec, State POINT_TO_POINT,
<output omitted>
```



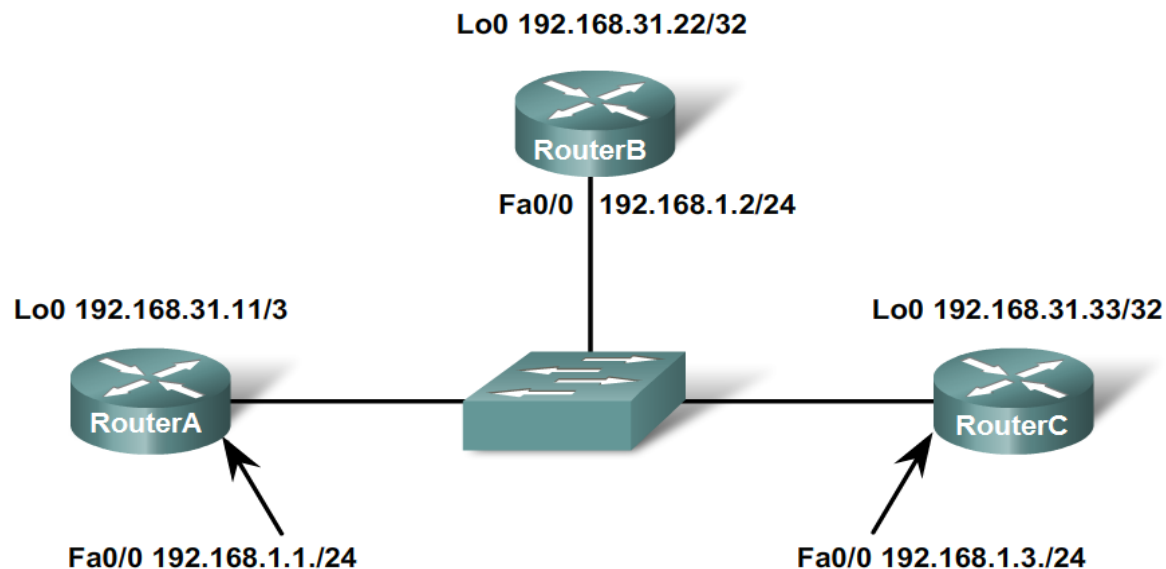
```
R1(config)#inter serial 0/0/0
R1(config-if)#ip ospf cost 1562
R1(config-if)#end
R1#show ip ospf interface serial 0/0/0
Serial0/0 is up, line protocol is up
Internet Address 192.168.10.1/30, Area 0
Process ID 1, Router ID 10.1.1.1, Network Type POINT_TO_POINT, Cost: 1562
Transmit Delay is 1 sec, State POINT_TO_POINT,
<output omitted>
```



OSPF in Ethernet

- + DR/BDR elections **will take place on multiaccess networks** as shown below

Multiaccess Three Router Topology

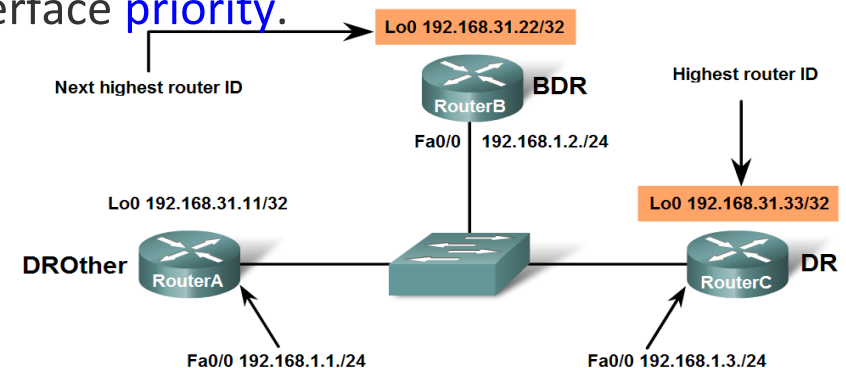


Notice that routers are now communicating via LAN interfaces.

OSPF in Ethernet

+ Criteria for getting elected DR/BDR

1. **DR:** Router with the **highest** OSPF interface **priority**.
- 2. **BDR:** Router with the **second highest** OSPF interface **priority**.
- 3. **If** OSPF interface **priorities are equal**, the **highest router ID** is used to break the tie.



```
RouterA#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.31.33	1	FULL/DR	00:00:39	192.168.1.3	FastEthernet0/0
192.168.31.22	1	FULL/BDR	00:00:36	192.168.1.2	FastEthernet0/0

```
RouterB#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.31.33	1	FULL/DR	00:00:34	192.168.1.3	FastEthernet0/0
192.168.31.11	1	FULL/DROTHER	00:00:38	192.168.1.1	FastEthernet0/0

```
RouterC#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.31.22	1	FULL/BDR	00:00:35	192.168.1.2	FastEthernet0
192.168.31.11	1	FULL/DROTHER	00:00:32	192.168.1.1	FastEthernet0

Priority is equal at the default value of 1.

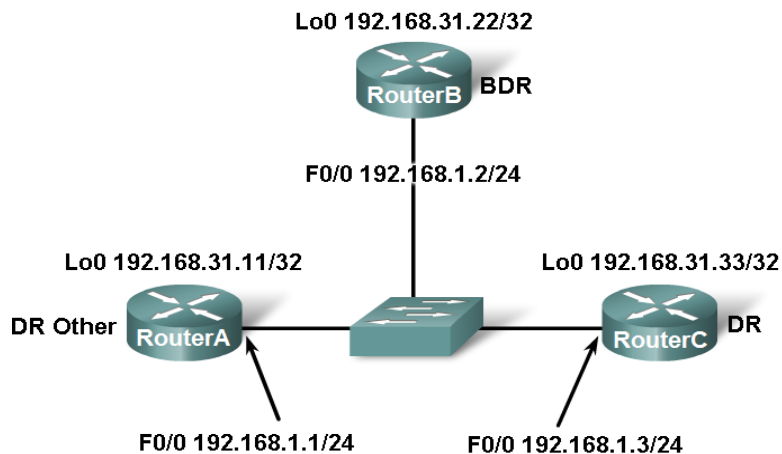
```
RouterA#show ip ospf interface fastethernet 0/0
```

```
FastEthernet0/0 is up, line protocol is up
Internet Address 192.168.1.1/24, Area 0
Process ID 1, Router ID 192.168.31.11, Network Type BROADCAST, Cost: 1
Transmit Delay is 1 sec, State DROTHER, Priority 1
Designated Router (ID) 192.168.31.33, Interface address 192.168.1.3
Backup Designated router (ID) 192.168.31.22, Interface address 192.168.1.2
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  oob-resync timeout 40
  Hello due in 00:00:06
Supports Link-local Signaling (LLS)
Index 1/1, flood queue length 0
Next 0x0(0)/0x0(0)
Last flood scan length is 0, maximum is 1
Last flood scan time is 0 msec, maximum is 0 msec
Neighbor Count is 2, Adjacent neighbor count is 2
  Adjacent with neighbor 192.168.31.22 (Backup Designated Router)
  Adjacent with neighbor 192.168.31.33 (Designated Router)
Suppress hello for 0 neighbor(s)
```

OSPF Interface Priority

+ Manipulating the DR/BDR election process continued

- Use the `ip ospf priority interface` command.
- Example: `Router(config-if)#ip ospf priority {0 - 255}`
 - Priority number range 0 to 255
 - 0 means the router cannot become the DR or BDR
 - 1 is the default priority value



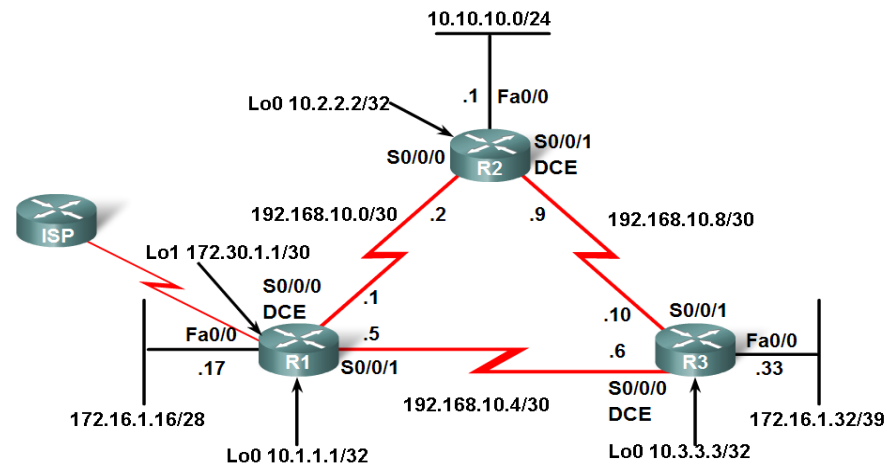
```
RouterA#show ip ospf interface fastethernet 0/0
FastEthernet0/0 is up, line protocol is up
Internet Address 192.168.1.1/24, Area 0
Process ID 1, Router ID 192.168.31.11, Network Type BROADCAST, Cost: 1
Transmit Delay is 1 sec, State DROTHER, Priority 1
Designated Router (ID) 192.168.31.33, Interface address 192.168.1.3
Backup Designated router (ID) 192.168.31.22, Interface address 192.168.1.2
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
oob-resync timeout 40
..
..
```

Redistributing an OSPF Default Route

+ Topology includes a link to ISP

– Router connected to ISP

- Called an autonomous system border router
- Used to propagate a default route
 - Example of static default route
 - **R1(config)#ip route 0.0.0.0 0.0.0.0 loopback 1**
 - Requires the use of the default-information originate command
 - Example of default-information originate command
 - **R1(config-router)#default-information originate**



Redistributing learned routes

+ The use of a routing protocol to advertise routes that are learned by some other means, such as by another routing protocol, static routes, or directly connected routes, is called **redistribution**.

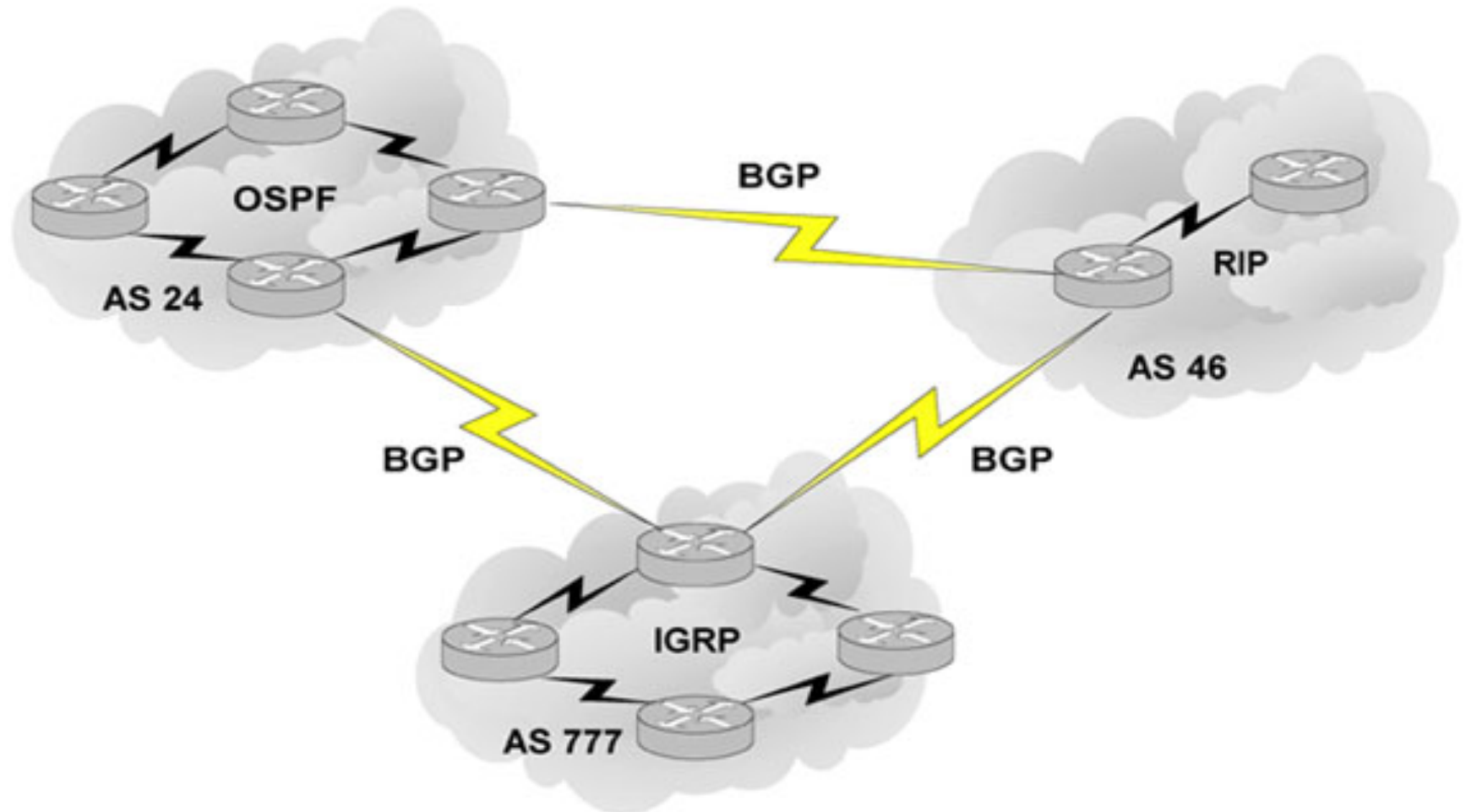
- iBGP → OSPF, OSPF → iBGP, etc.

+ Some possible OSPF redistribution

- bgp Border Gateway Protocol (BGP)
- connected Connected
- metric Metric for redistributed routes
- static Static routes

Border Gateway Protocol (BGP)

BGP – an Exterior Gateway Protocol



When to use BGP versus a static route to your provider

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 not 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

EGPs – Exterior Gateway Protocols

- + Typically, EGPs are used to exchange routing information between ISPs, or in some cases between a customer's AS and the provider's network.
- + **Border Gateway Protocol (BGP)**, version 4 (BGP4) is the most common EGP and is considered the Internet standard.

Autonomous Systems

- + An internetwork is a confederation of smaller, independent networks, called **autonomous systems**, owned and operated by a different organization: a company, university, government agency, or some other group.
- + Each AS typically represents an independent organization, and applies its own unique routing and security policies.
- + EGPs facilitate the sharing of routing information between autonomous systems.

AS Numbers

- + Each AS has an identifying number, assigned by an Internet registry or a service provider, between **1 and 65535**.
- + Private AS numbers: Between **64512 through 65535**
 - Similar to RFC 1918 IP addresses
 - Because of the finite number of available AS numbers, an organization must present justification of its need before it will be assigned an AS number.

AS Numbers

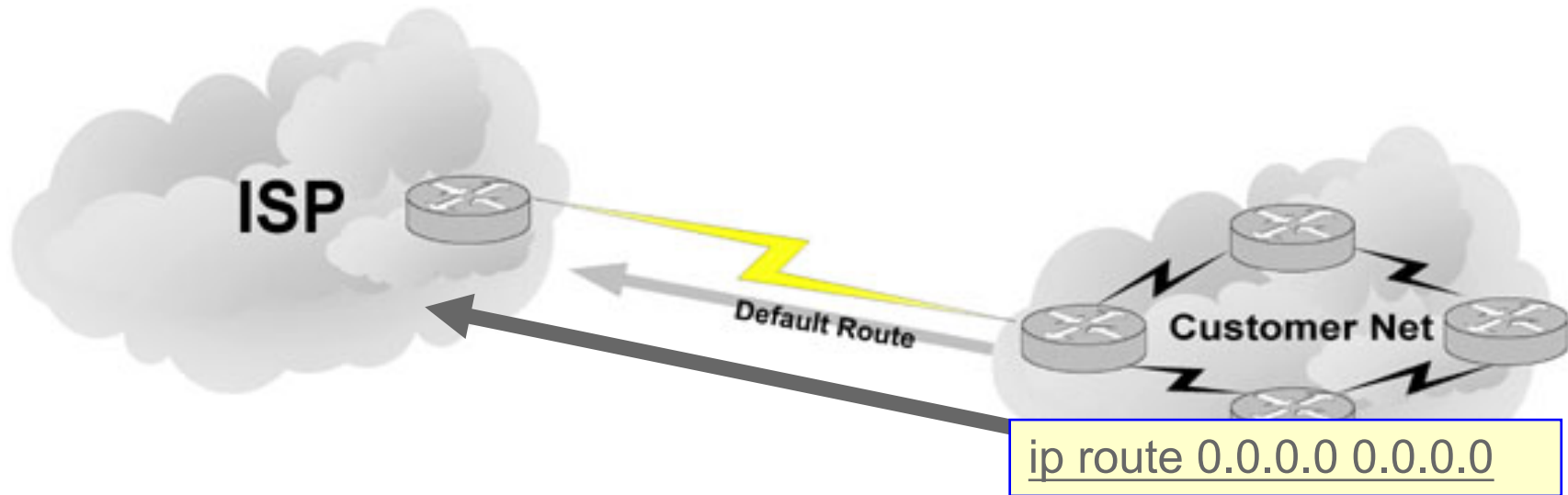
- + Organizations that connect to a single provider and share the provider's routing policies use an AS number from the private pool (64,512-65,535).
- + These private AS numbers appear only within the provider's network, and are replaced by the provider's registered number upon exiting the network.
 - Thus, to the outside world, several individual networks are advertised as part of one service provider's network.

Who Needs BGP?

- + Not as many internetworks as you may think.
- + “You should implement BGP only when a sound engineering reason compels you to do so, such as when the IGP does not provide the tools necessary to implement the required routing policies or when the size of the routing table cannot be controlled with summarization.”
- + “The majority of the cases calling for BGP involve Internet connectivity – either between a subscriber and an ISP or (more likely) between ISPs.”
- + “Yet even when interconnecting autonomous systems, BGP might be unnecessary.”

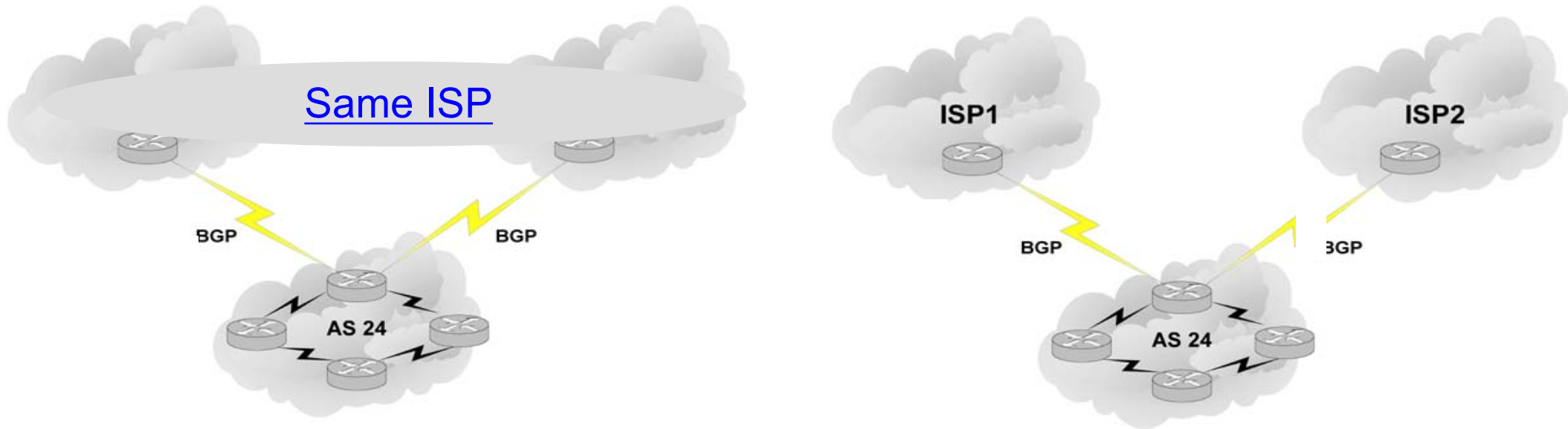
Jeff Dolye, Routing TCP/IP Vol. II

Single-Homed AS



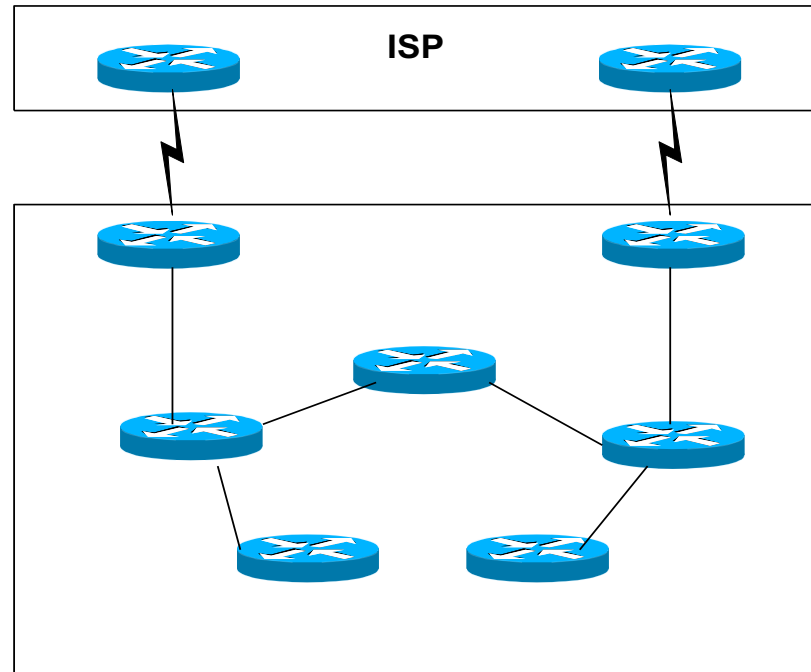
- + If an AS has only one exit point to outside networks, it is considered a **single-homed system**.
- + Single-homed autonomous systems are often referred to as **stub networks**, or *stubs*.
- + Stubs can rely on a default route to handle all traffic destined for nonlocal networks.
- + BGP is not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

Multi-homed Autonomous Systems



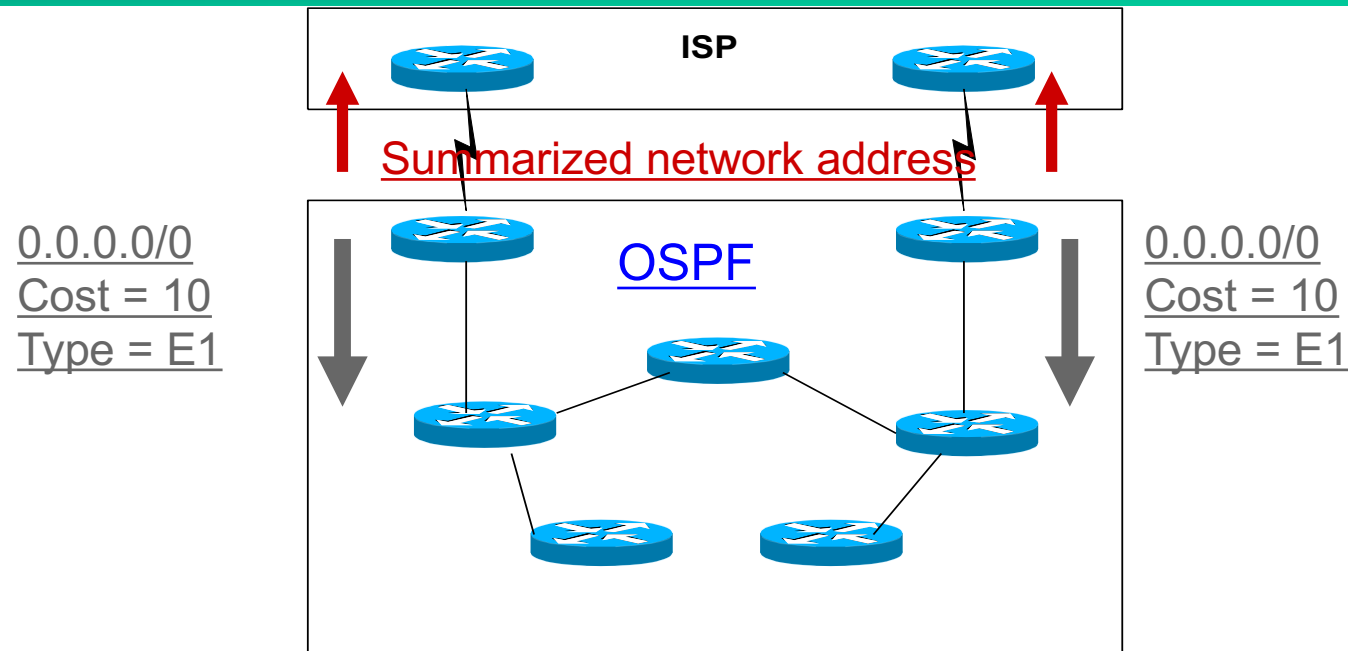
- + An AS is a **multi-homed system** if it has more than one exit point to outside networks.
- + An AS connected to the Internet can be multi-homed to:
 - a single provider (AS)
 - multiple providers (ASs)

Multi-homed to a Single Autonomous Systems



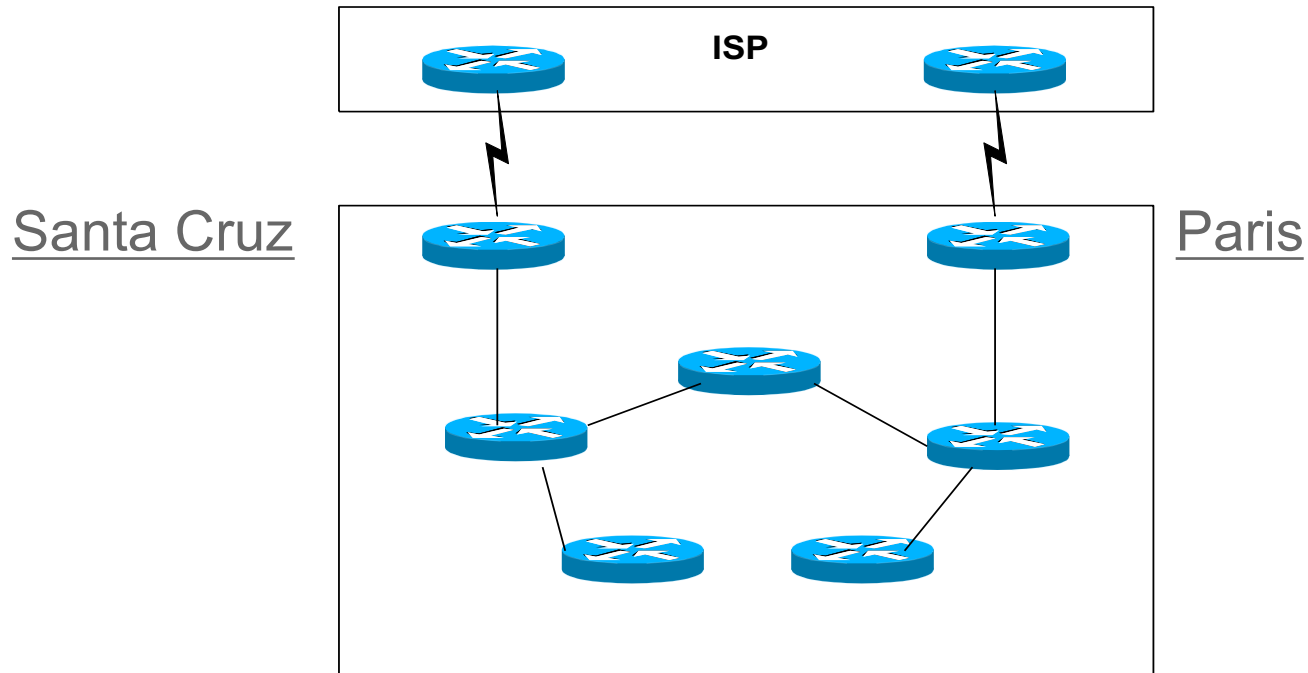
- + This is an improved topology over Single-Home AS, providing for redundancy.
- + One option may be to use one link as the **primary** link and the other as a **backup** link.

Multi-homed to a Single Autonomous Systems



- + 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.
- + One example would be to run a dynamic routing protocol like OSPF within your network, with static default routes advertised at both campus entrance routers into your network.
- + As a result, every router chooses the closest exit point, when choosing a default route.
- + In most cases this will be sufficient for good internetwork performance.

Multi-homed to a Single Autonomous Systems

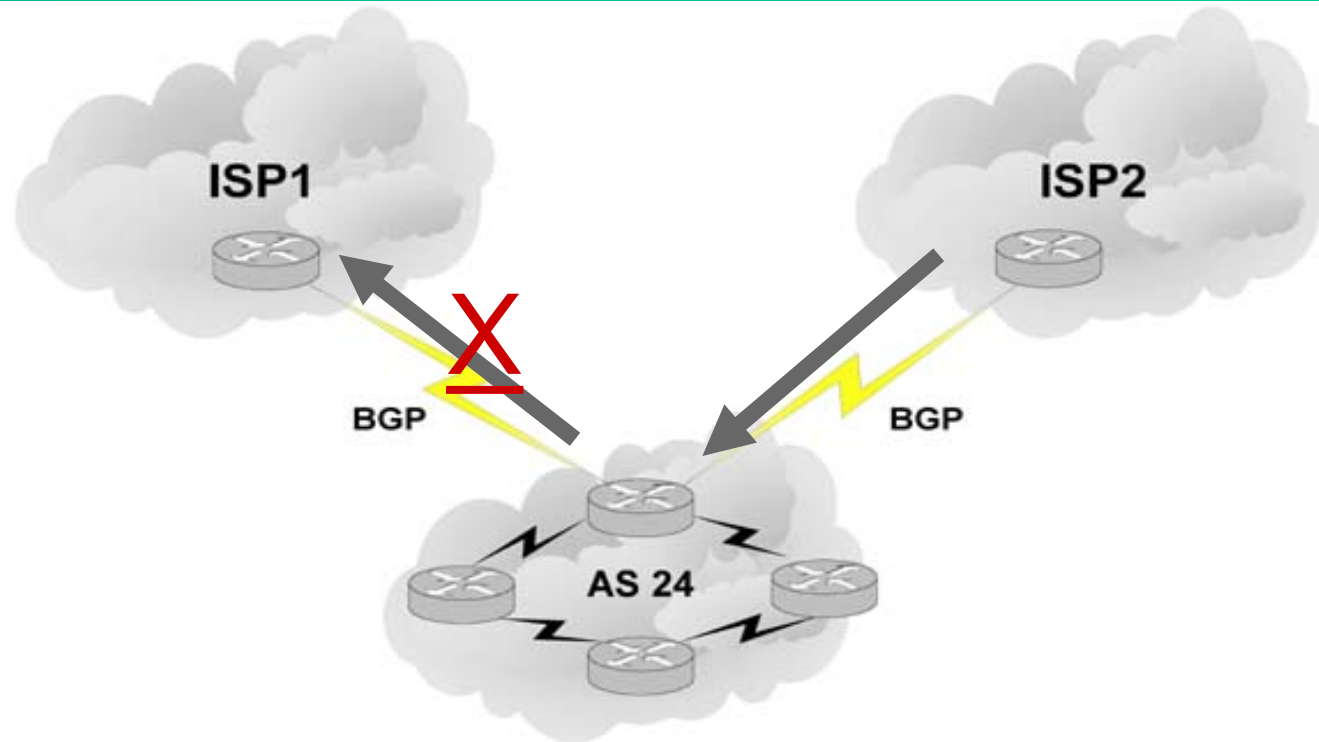


- + If the geographical separation between the two entrance routers is large enough for delay variations to become significant (one router closer to some networks, while the other router is closer to other networks), you might have a need for better control of the routing.
- + In this case BGP *might* be a consideration.

Multi-homed to a Single Autonomous Systems

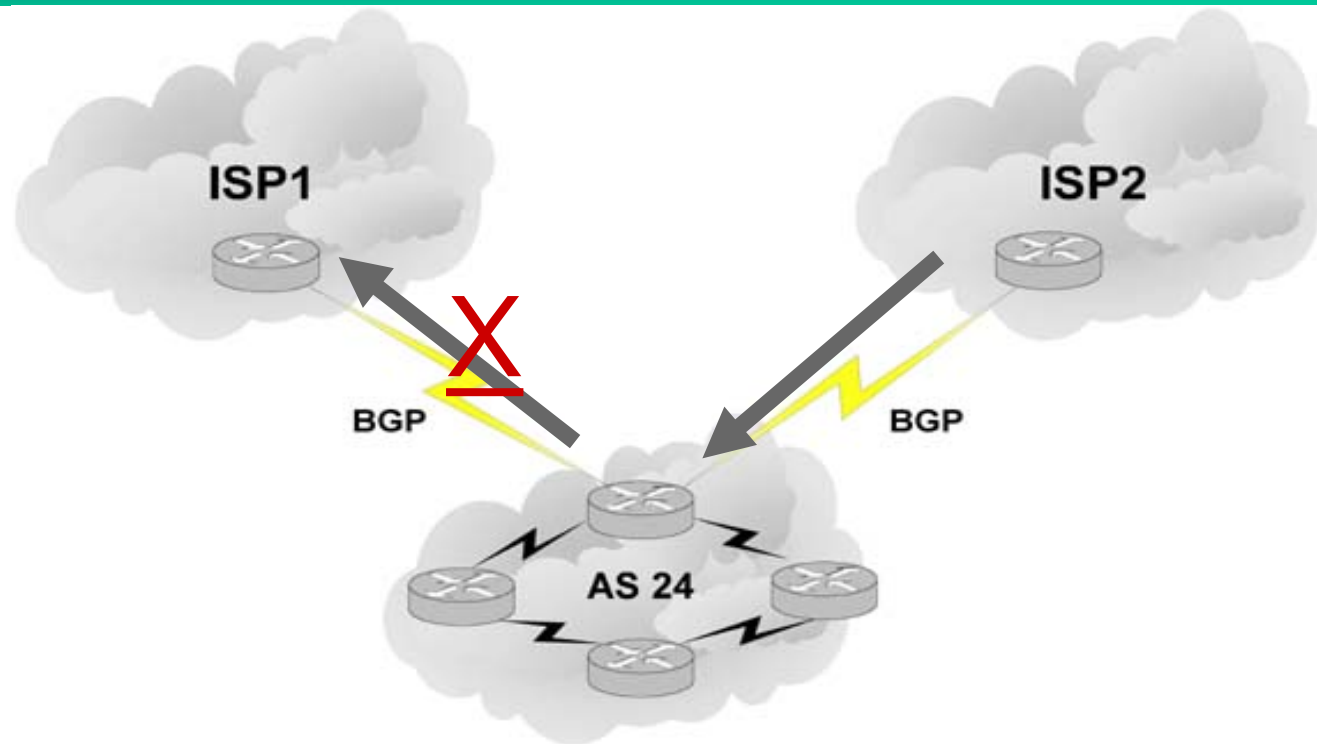
- + *Incoming route advertisements influence your outgoing traffic, and outgoing advertisements influence your incoming traffic.*
- + If the provider advertises routes into your AS via BGP, your internal routers have more accurate information about external destinations.
 - BGP also provides tools for 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 Non-transit Autonomous Systems



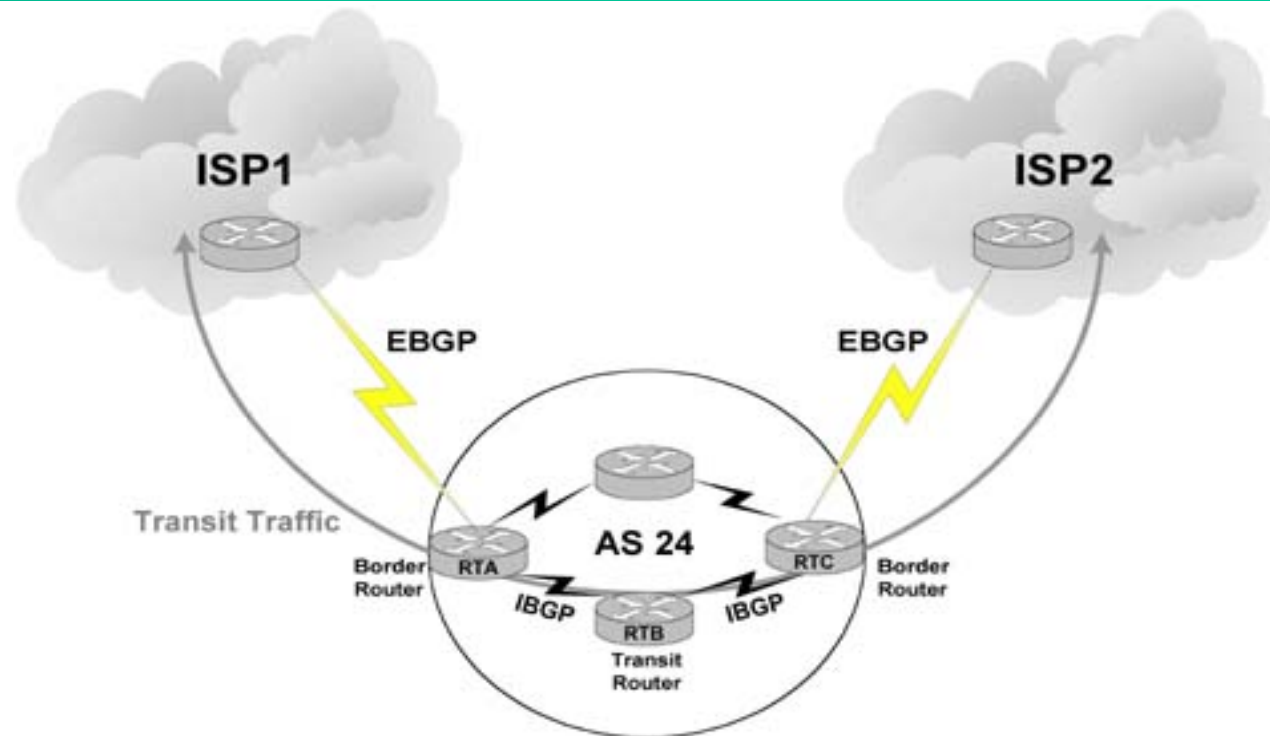
- + A **non-transit AS** does not allow transit traffic—that is, any traffic that has a source and destination outside the AS—to pass through it.
- + A **non-transit AS** would advertise only its *own* routes to both the providers it connects to—it would not advertise routes it learned from one provider to another.

Multi-homed Non-transit Autonomous Systems



- + Multi-homed **non-transit** autonomous systems don't really need to run BGP4 with their providers, although it is recommended, and often required by ISPs.
- + BGP4 offers numerous advantages, including increased control of route propagation and filtering.

Multi-homed Transit Autonomous Systems



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

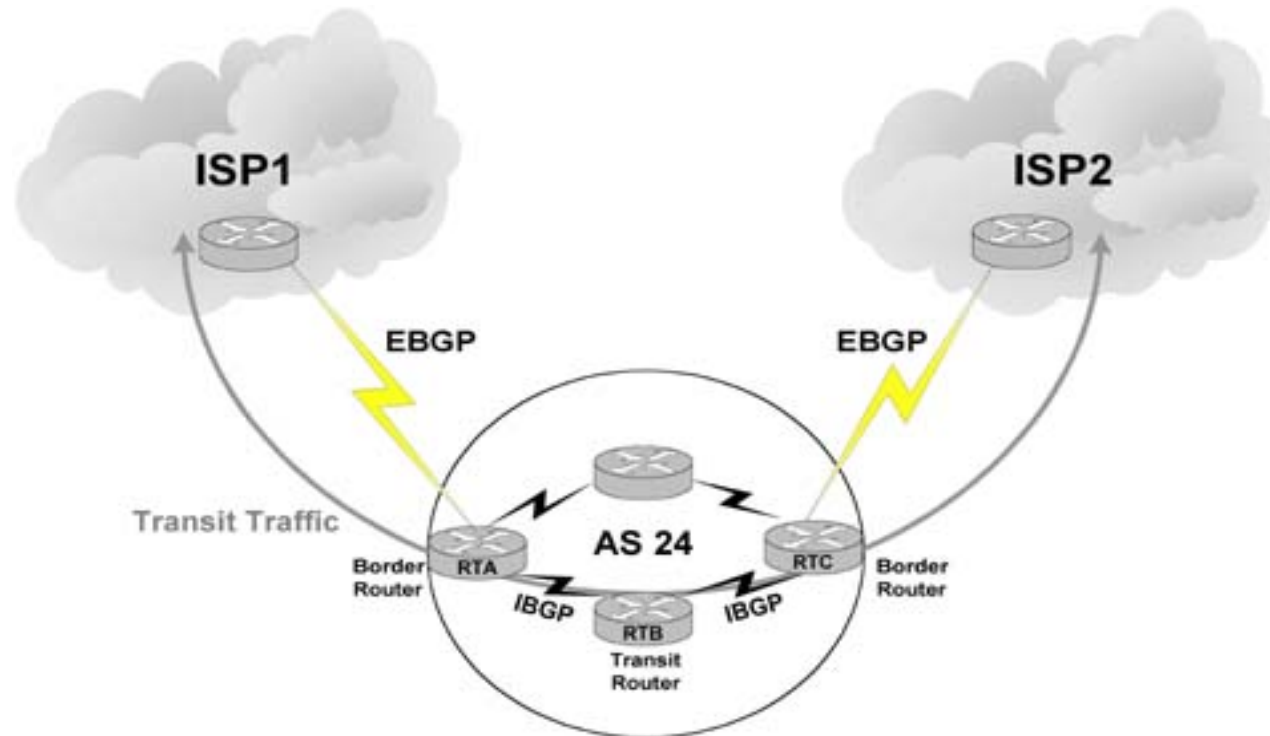
BGP Hazards

- + 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.

BGP Hazards

- + 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 Hazards: Another Example

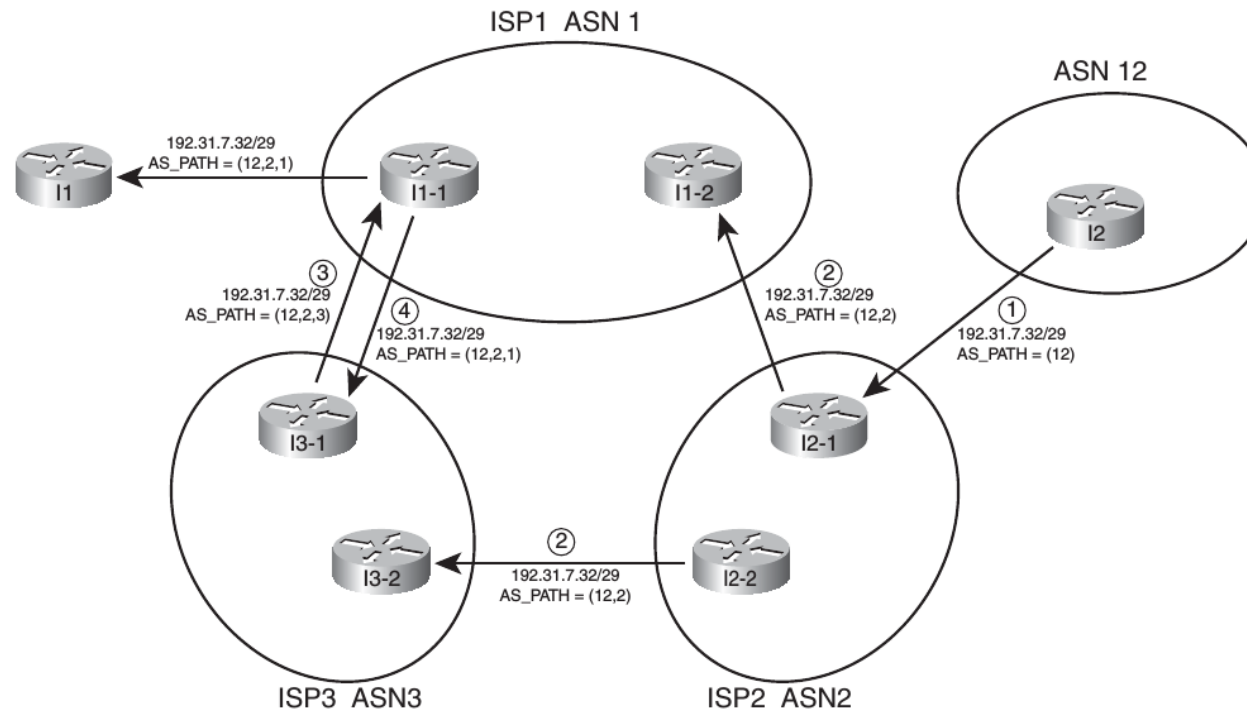


- + We inadvertently advertise routes learned from ISP2 to ISP1.
- + ISP1 customers will see our network as the best path to ISP2 customers.
- + We have become a transit domain for packets from ISP1 to ISP2.

BGP Basics

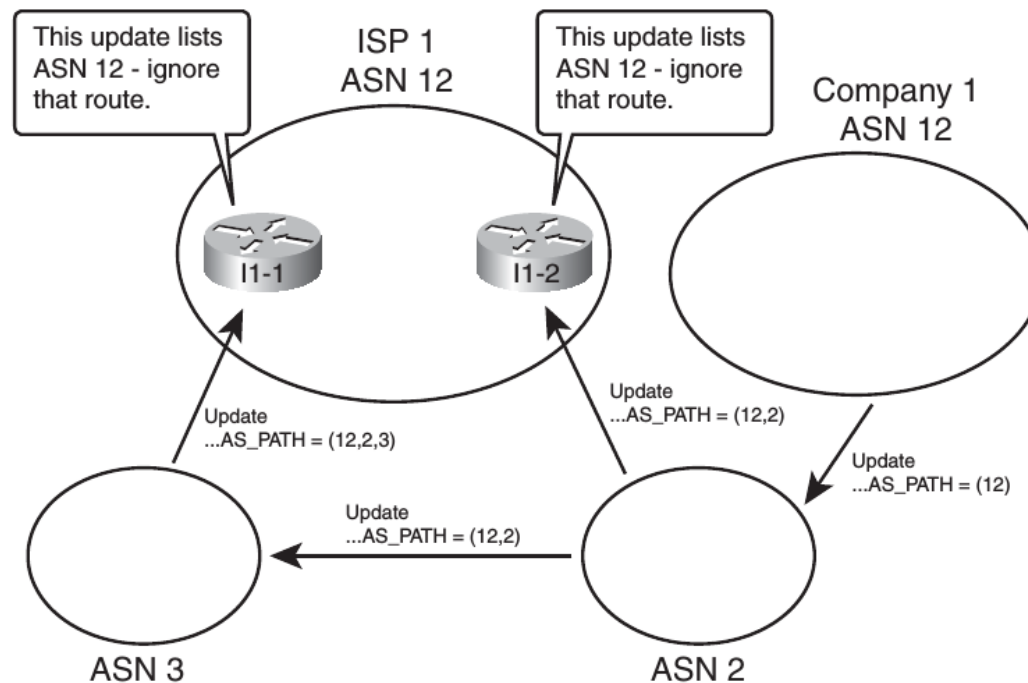
- + BGP is a path vector routing protocol – more in a moment.
- + BGP is a distance vector routing protocol, in that it relies on downstream neighbors to pass along routes from their routing table.
- + The node makes its route calculations based on those advertised routes and passes the results to upstream neighbors.
- + Other distance vector routing quantify results with a single number, (hops, shortest BW + sum of DLYs, etc.).
- + BGP uses a list of AS numbers through which a packet must pass to reach a destination.

AS 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

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 Basics

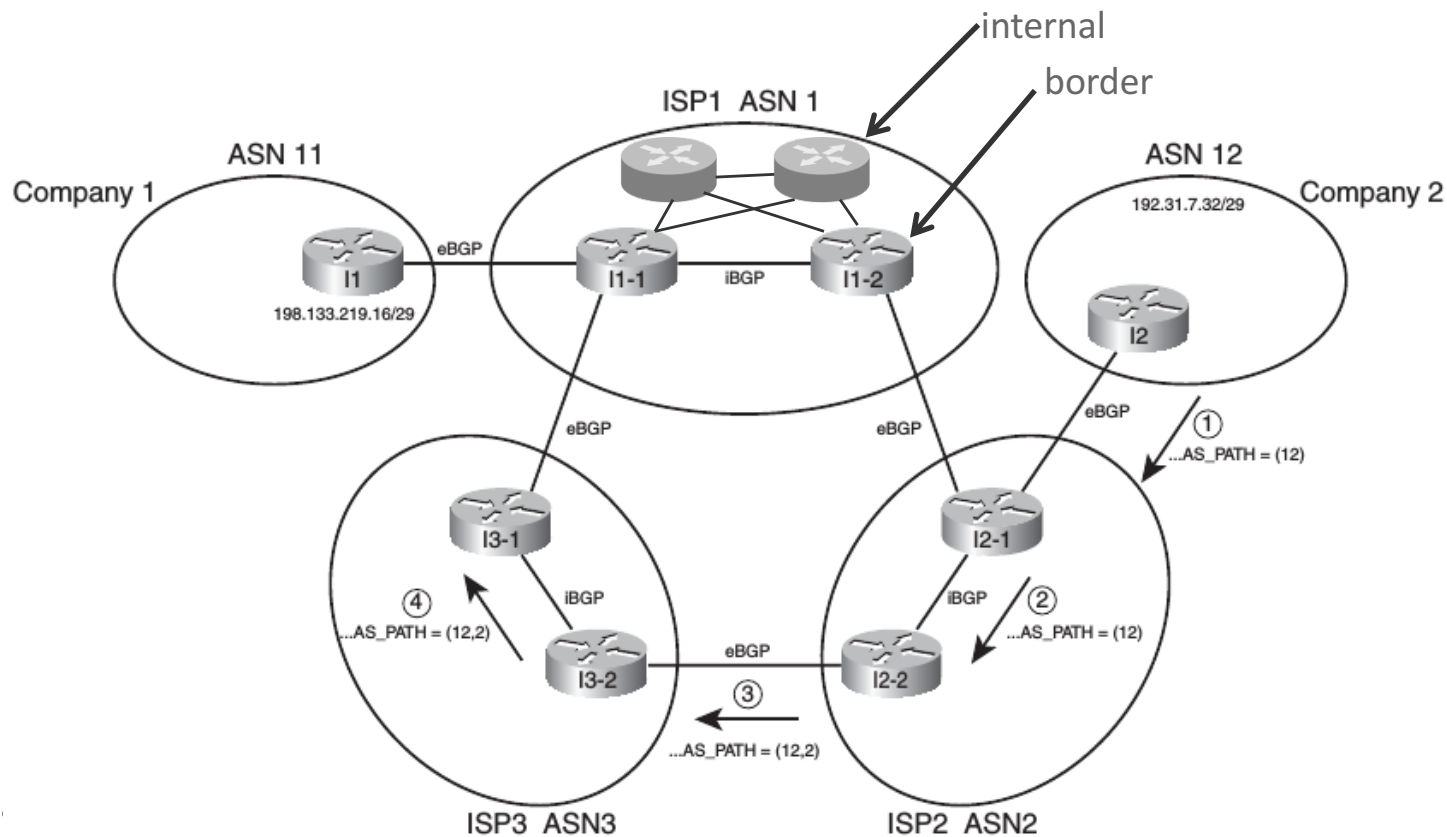
- + BGP4 is the first version of BGP that supports CIDR and route aggregation.
- + BGP does not use technical metrics, instead, BGP makes routing decisions based on network policies.
- + BGP does not show the details of topologies within each AS.
- + BGP sees only a tree of autonomous systems.

IBGP v EBGP

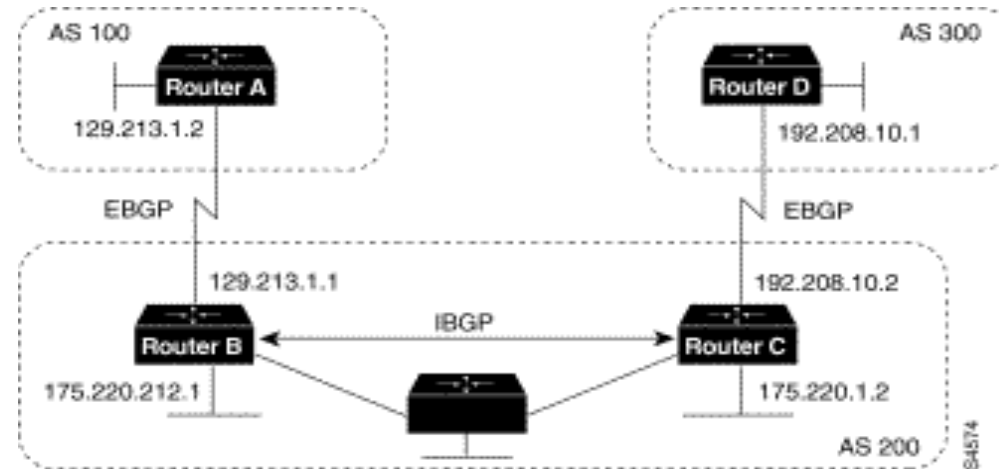
- + When BGP is running inside an AS, it is referred to as **Internal BGP (IBGP)**.
 - If a BGP router's role is to route IBGP traffic, it is called a transit router.
- + When BGP runs between autonomous systems, it is called **External BGP (EBGP)**.
 - Routers that sit on the boundary of an AS and use 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

IBGP v EBGP

- + External BGP is used to exchange prefixes with other ASes
- + Internal BGP is used to carry:
 - some/all Internet prefixes across ISP backbone
 - ISP's customer prefixes

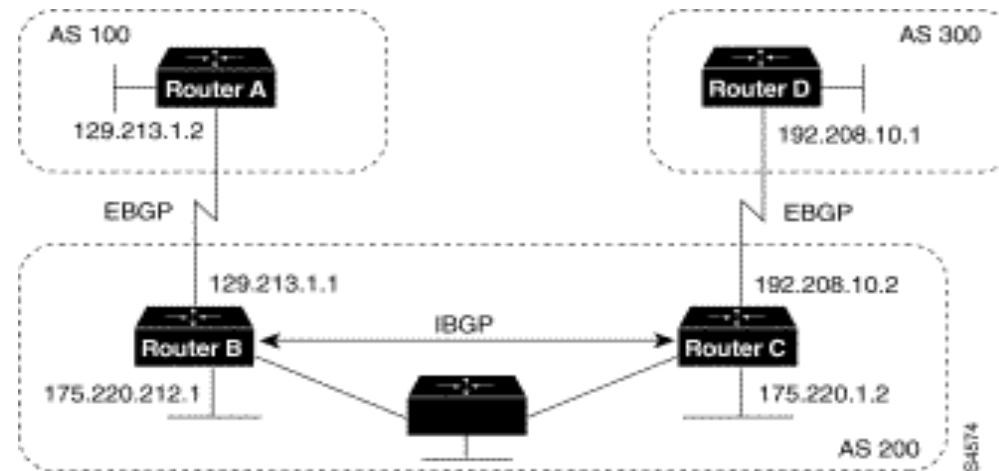


IBGP v EBGP



- + Routers A and B are running EBGP, and Routers B and C are running IBGP.
- + Note that the **EBGP** 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.

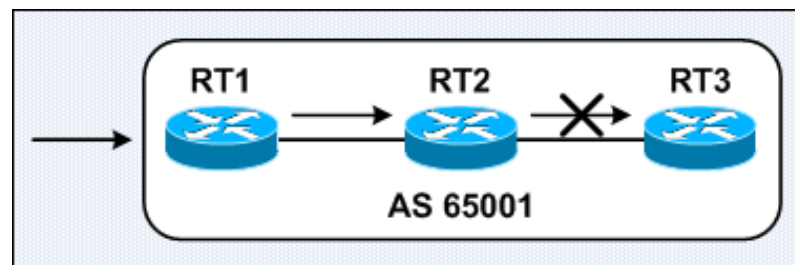
BGP uses TCP



- + BGP updates are carried using **TCP** on port 179.
- + Because BGP requires TCP, IP connectivity must exist between BGP peers and TCP connections must be negotiated between them before updates can be exchanged.
 - Thus, BGP inherits TCP's reliable, connection-oriented properties.

BGP Split Horizon Rule

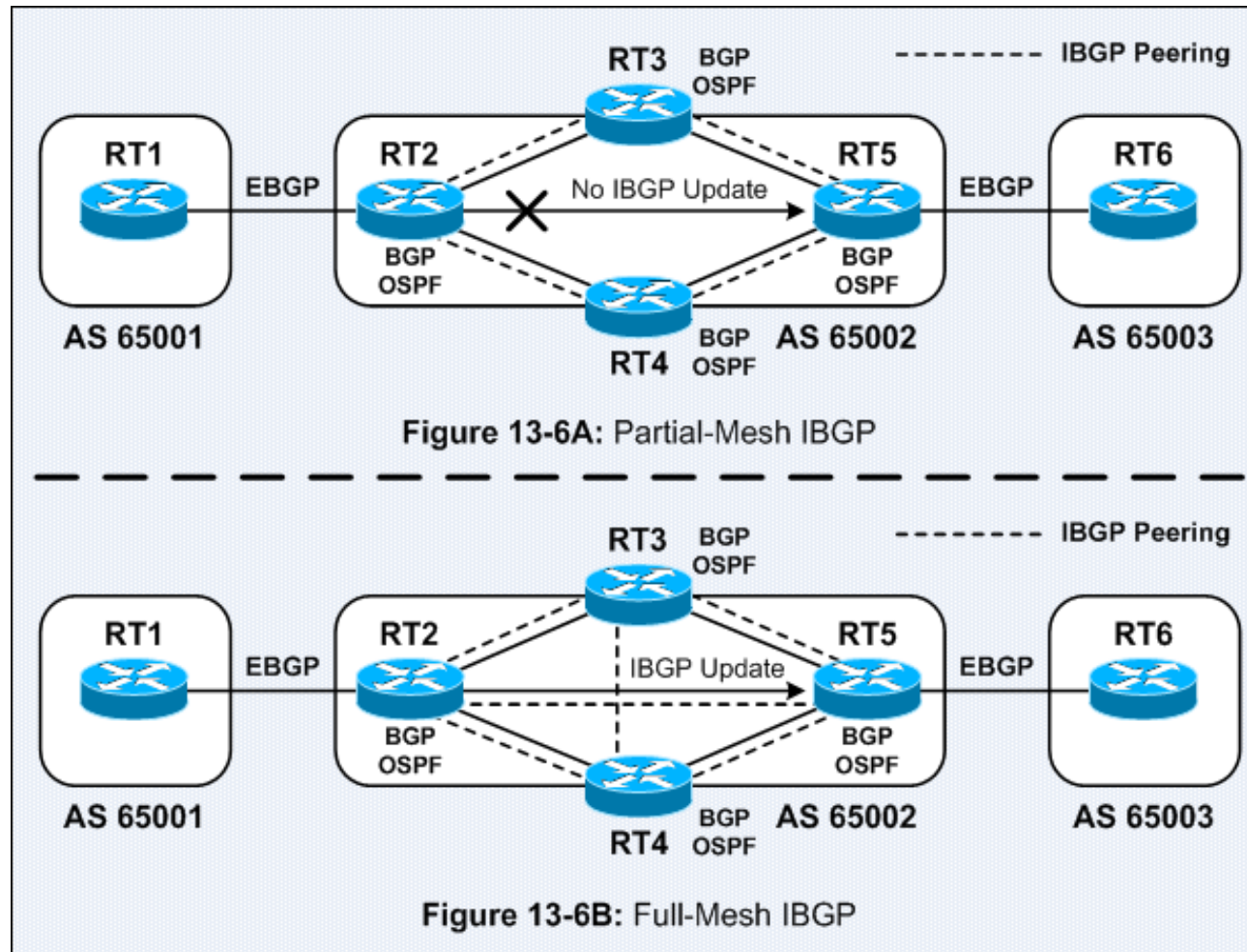
- + AS number is added to the AS path only when a BGP UPDATE goes out from the AS
- + AS path check can not be used to avoid loops within a AS
- + BGP Split Horizon Rule used to avoid loop inside the AS
 - A route learned via IBGP is never propagated to other IBGP peers.
 - A route is not advertised back to the EBGP peer from which the route was received



BGP Split Horizon Rule - consequence

- + A border router must have iBGP peering with all other border routers to relay external iBGP updates through the AS
 - Only if the AS is a transit one
- + A border router must have a iBGP peering with those internal routers that need the full routing table
- + An internal router that can inject BGP routes must have a peering with all border routers and with those internal routers that needs the full routing table
- + If all internal routers need the full routing table, there should be a **full-mesh of iBGP sessions**
 - 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.
 - An alternative to this approach: configuring a route reflector (In few slides)
- + Redistribution of BGP in IGP (e.g. OSPF) is **not recommended** since IGPs are not meant for big routing tables
 - Vice versa may be OK

BGP Split Horizon Rule - consequence



IGP-BGP Synchronization

- + The BGP synchronization rule states that a BGP router should not advertise to external neighbors destinations learned from IBGP neighbors unless those destinations are also known via an **IGP**.
- + 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
- + If the IBGP router does have an **IGP** route to this destination, the route is considered *synchronized*, and the router will announce it to other BGP peers.
- + Otherwise, the router will treat the route as not being synchronized with the **IGP** and will not advertise it.

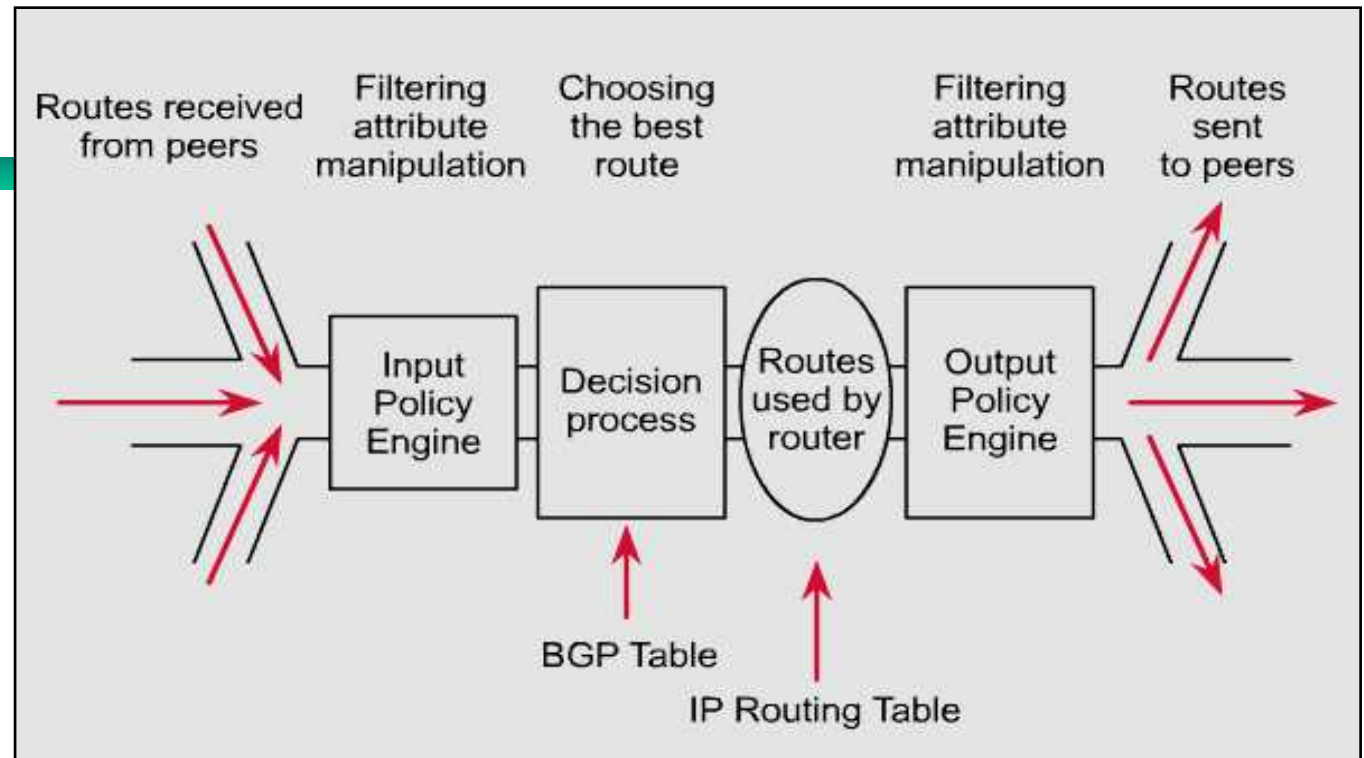
IGP-BGP Synchronization - consequence

- + 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
- + Router OS usually offers an optional command (e.g. **no synchronization**) to override the synchronization requirement, allowing the router to advertise routes learned via iBGP irrespective of an existence of an **IGP** route.

BGP NEXT_HOP

- + It is the IP address of the EBGP router advertising a prefix
- + By default, it is NOT changed while traversing IBGP sessions
- + This implies that a IBGP receiver to install an external BGP UPDATE must know how to reach the border AS node that generates the UPDATE
- + This behavior can be hacked, by forcing inbound border router to update next hop field with their address
 - E.g. Cisco: `neighbor 5.5.5.5 next-hop-self`

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 Operation

- + When two routers establish a TCP-enabled BGP connection between each other, they are called **neighbors** or **peers**.
- + Each router running BGP is called a **BGP speaker**.
- + When two neighbors first establish a BGP connection, they exchange their entire BGP routing tables.
- + After that, they exchange incremental, partial updates with only the information that has changed.

BGP Operation

- + Peers exchange keepalive messages to ensure the connection is maintained.
- + Usually the default keepalive interval is 60 seconds
- + If three keepalive intervals pass the peer declares its neighbor down.
- + These can be modified with timers bgp command.

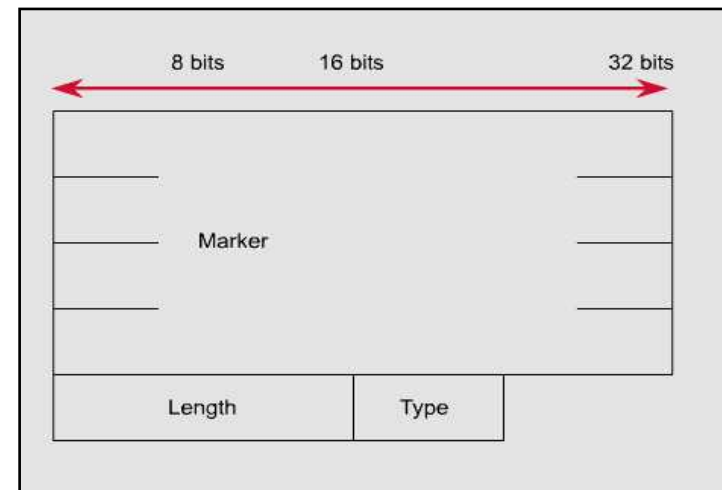
BGP Message Types

- + 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

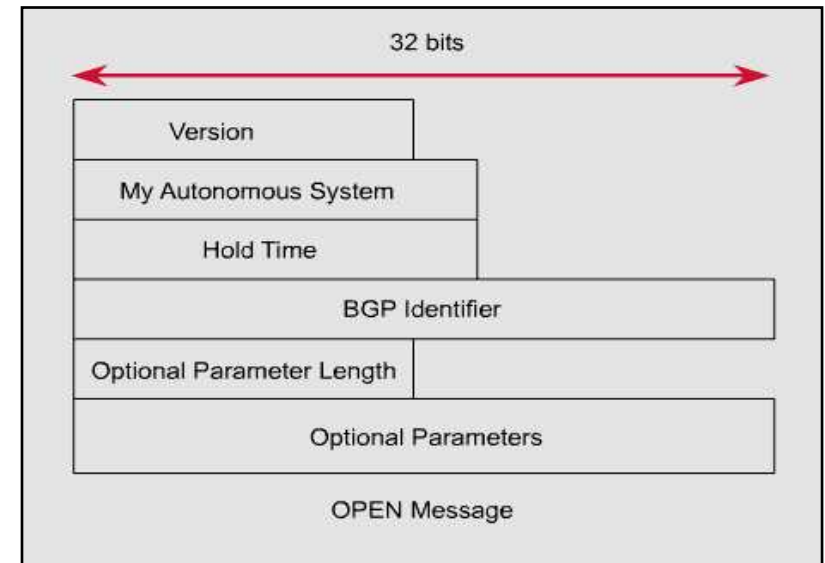
Each BGP Message contains the following header:

- + **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.



Type 1: BGP Open Message

- + After the TCP session is established, both neighbors send Open messages.
- + 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.



Type 2: BGP Keepalive Message

- + If a router accepts the parameters specified in its neighbor's Open message, it responds with a Keepalive.
- + Subsequent Keepalives are sent every (e.g.) 60 seconds or equal to one-third the agreed-upon hold time (180 seconds).

Type 3: BGP Update Message

- + The UPDATE messages contain all the information BGP uses to construct a loop-free picture of the internet network.
- + Update messages advertises feasible routes, withdrawn routes, or both.
- + The three basic components of an UPDATE message are:
 - **Network-Layer Reachability Information (NLRI)**
 - Path Attributes
 - Withdrawn Routes

Type 3: BGP Update Message

Network-Layer Reachability Information (NLRI)

- + This is one or more (Length, Prefix) tuples that advertise IP address prefixes and their lengths.
- + 192.168.160.0/19
 - Length = /19
 - Prefix = 192.168.160.0

Path Attributes

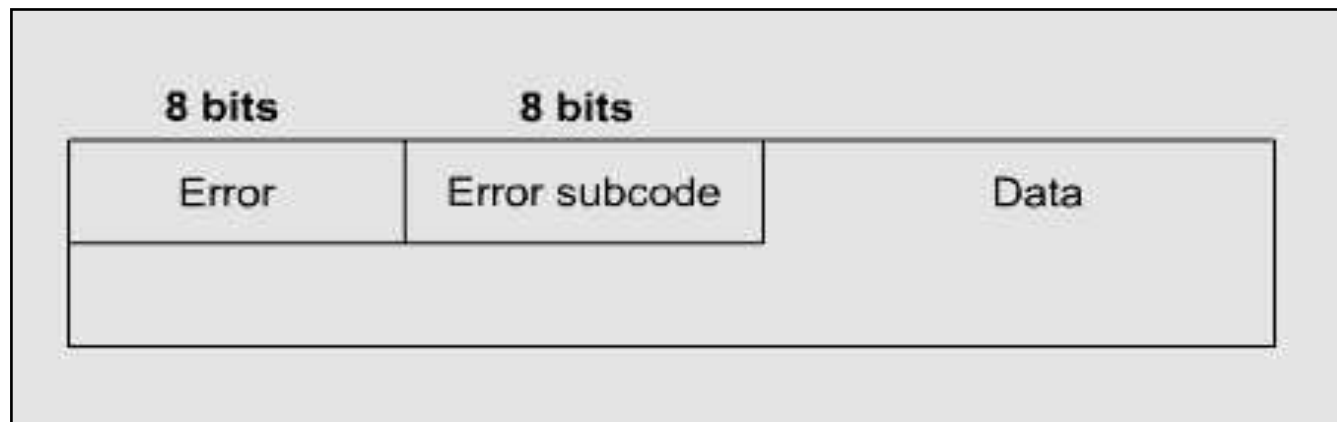
- + This is described later, providing the information that allows BGP to choose a shortest path, detect routing loops, and determine routing policy.

Withdrawn Routes

- + These are (Length, Prefix) tuples describing destination that have become unreachable and are being withdrawn from service.

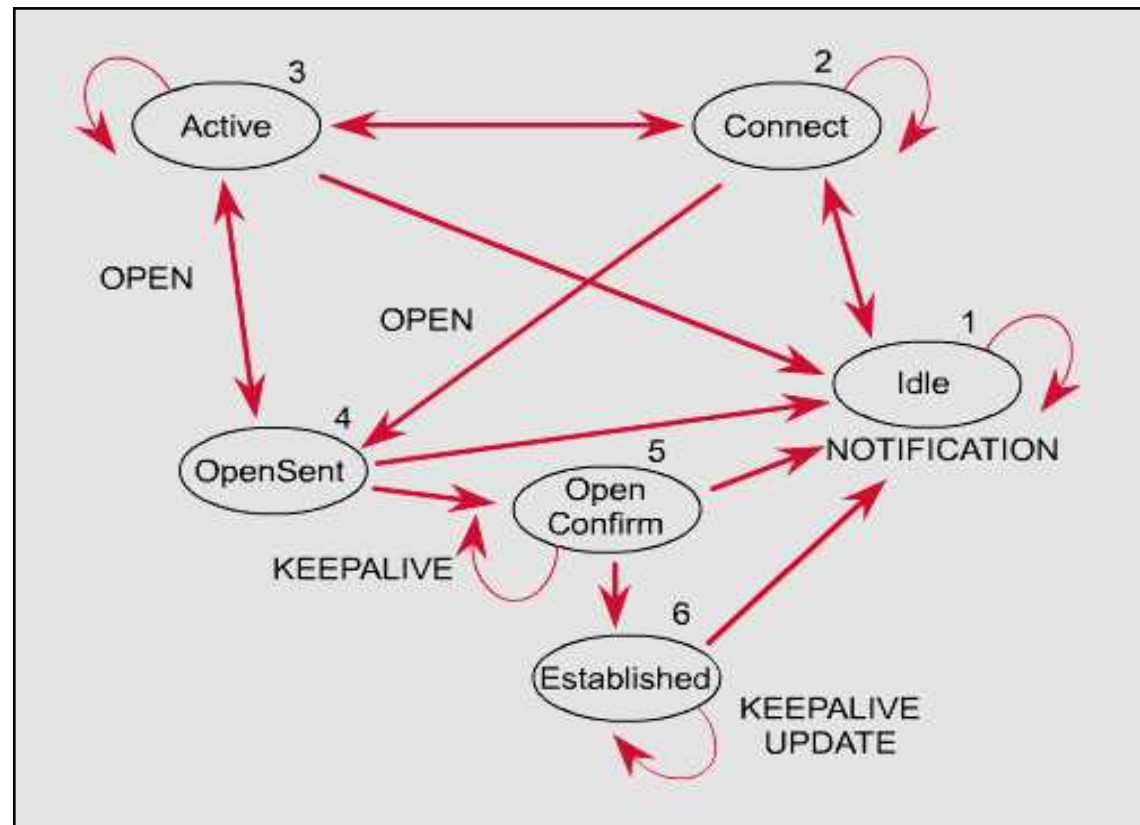
Type 4: BGP Notification Message

- + A NOTIFICATION message is sent whenever an error is detected and always causes the BGP connection to close.
- + The NOTIFICATION message is composed of the Error Code (8 bits), Error Subcode (8 bits), and a Data fields (variable length).



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 – Idle state

- + BGP always begins in the Idle state, in which it refuses all incoming connections.
- + 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

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

- + 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

- + 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**.



BASIC BGP CISCO CONFIGURATION 1

AS number

- + 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.**

Manually advertise a network

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

- + The **network** command is used with **IGPs**, such as OSPF, to determine the interfaces on which to send and receive updates, as well as which directly connected networks to advertise.
- + 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 OSPF

IGP-BGP sync

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.
- + Routes learned by the BGP process are propagated by default, but are often filtered by a routing policy.

BGP peering

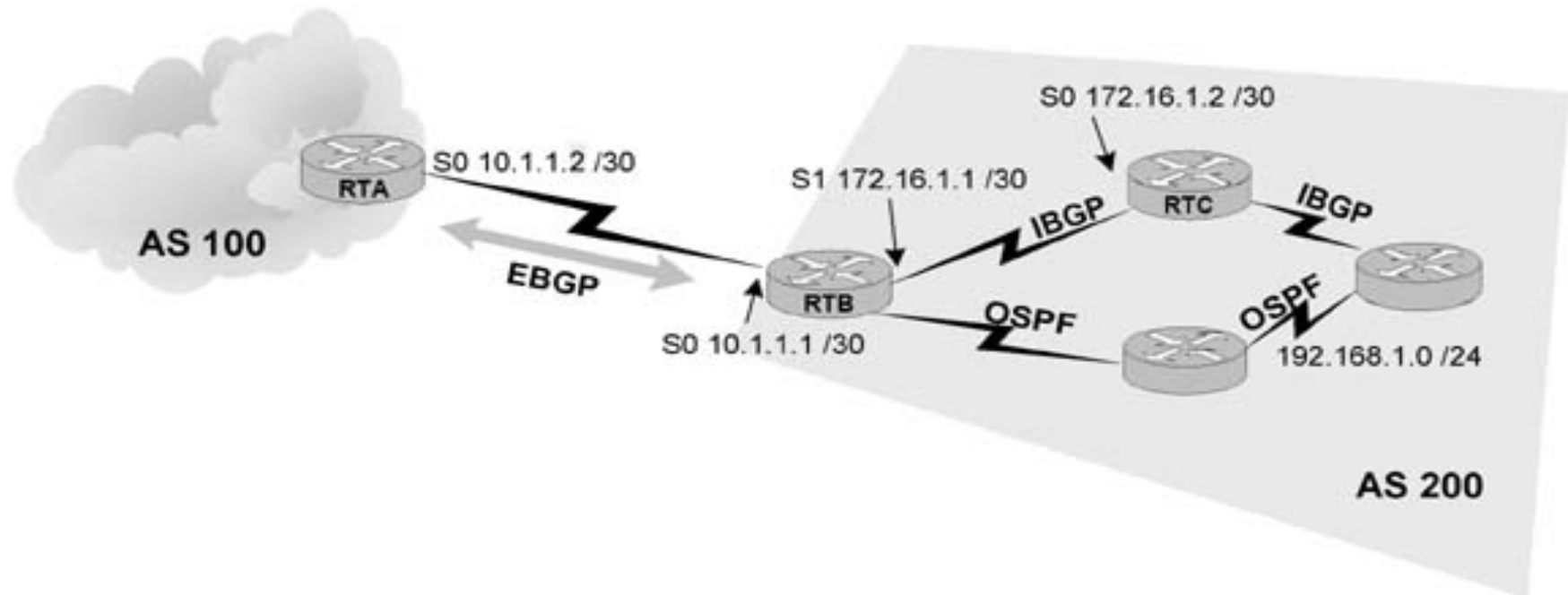
```
Router (config-router) #neighbor ip-address  
remote-as AS-number
```

- + In order for a BGP router to **establish a neighbor relationship with another BGP router**, you must issue the above 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 and Loopback interfaces

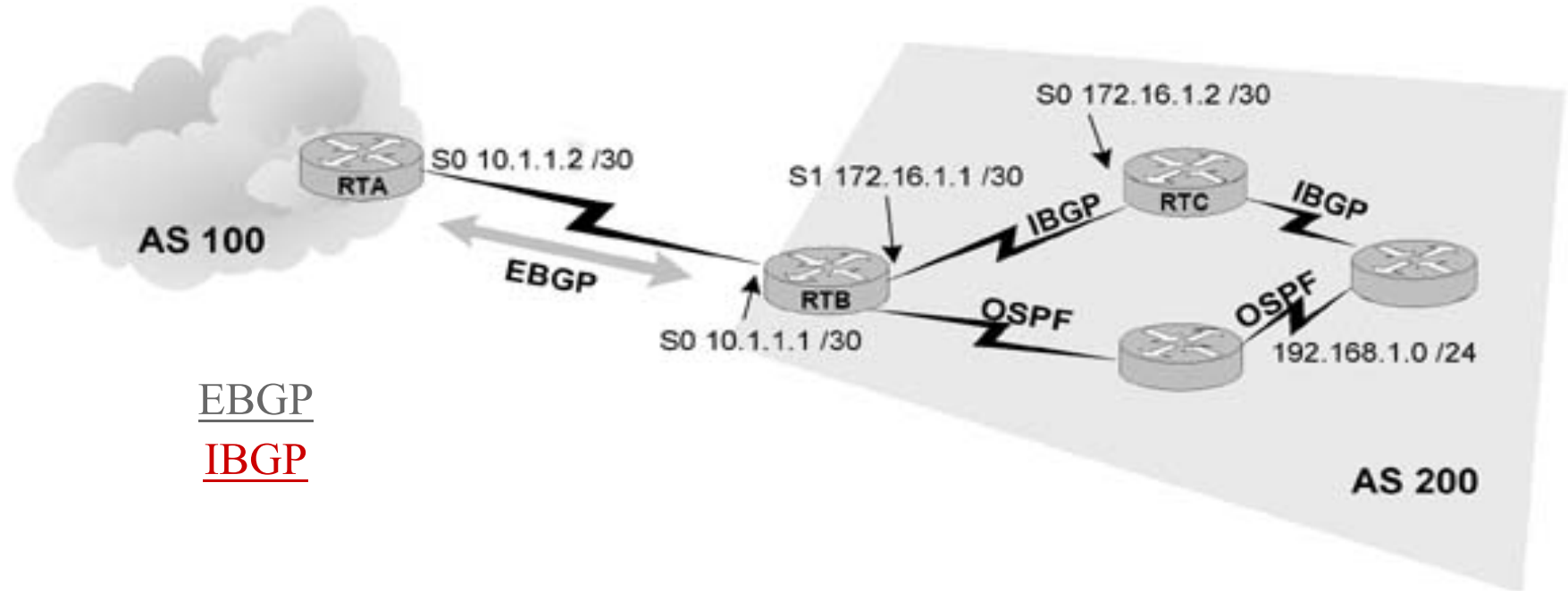
- + BGP peering session may use **loopback interfaces**
- + Loopback interface ensures that the neighbor stays up and is not affected by malfunctioning hardware of a specific interface
- + Useful when there are multiple paths between the BGP peers
 - iBGP peering are among internal routers generally connected by different IGP (OSPF) routes
- + Useful also in case of multilink, for load balancing
- + Best Practice
 - IBGP on loopback
 - EBGP on physical interface (unless multilink)

EBGP v IBGP



- + 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



EBGP

IBGP

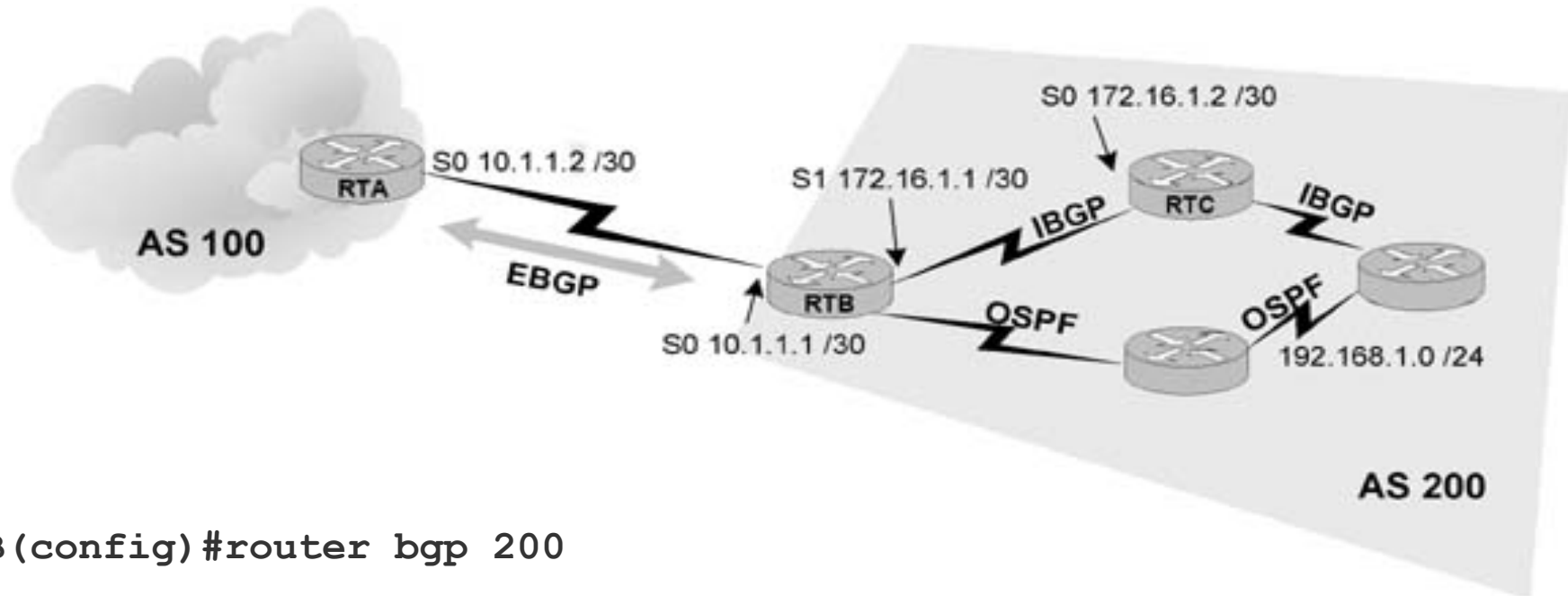
```
RTA(config)#router bgp 100
```

```
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
```

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
```

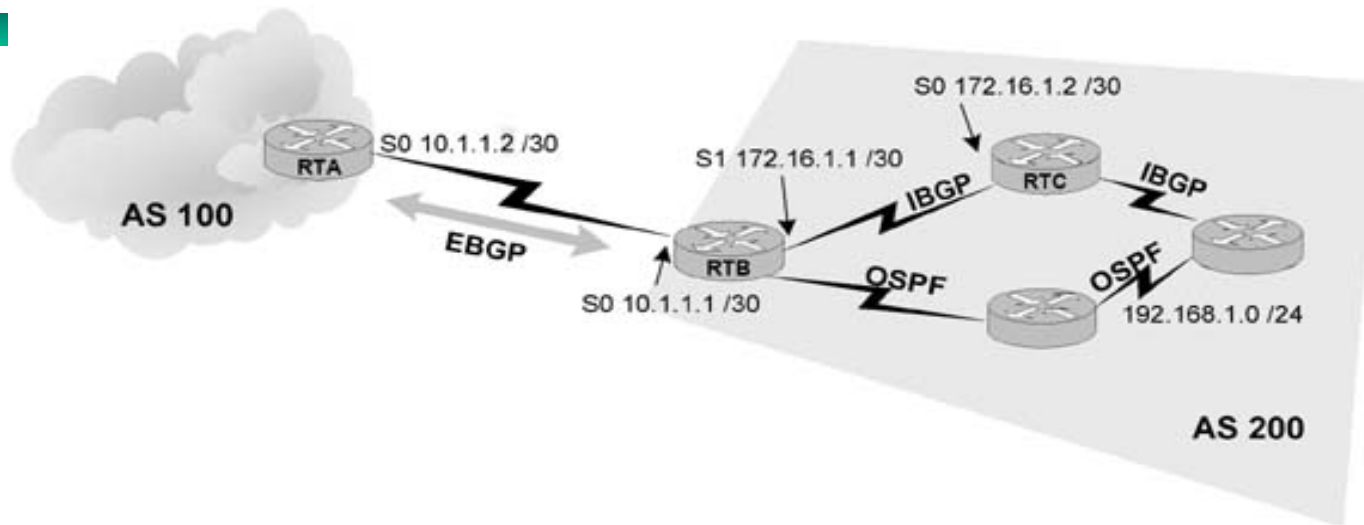
```
RTC(config-router)#neighbor 172.16.1.1 remote-as 200
```

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

Loopback source

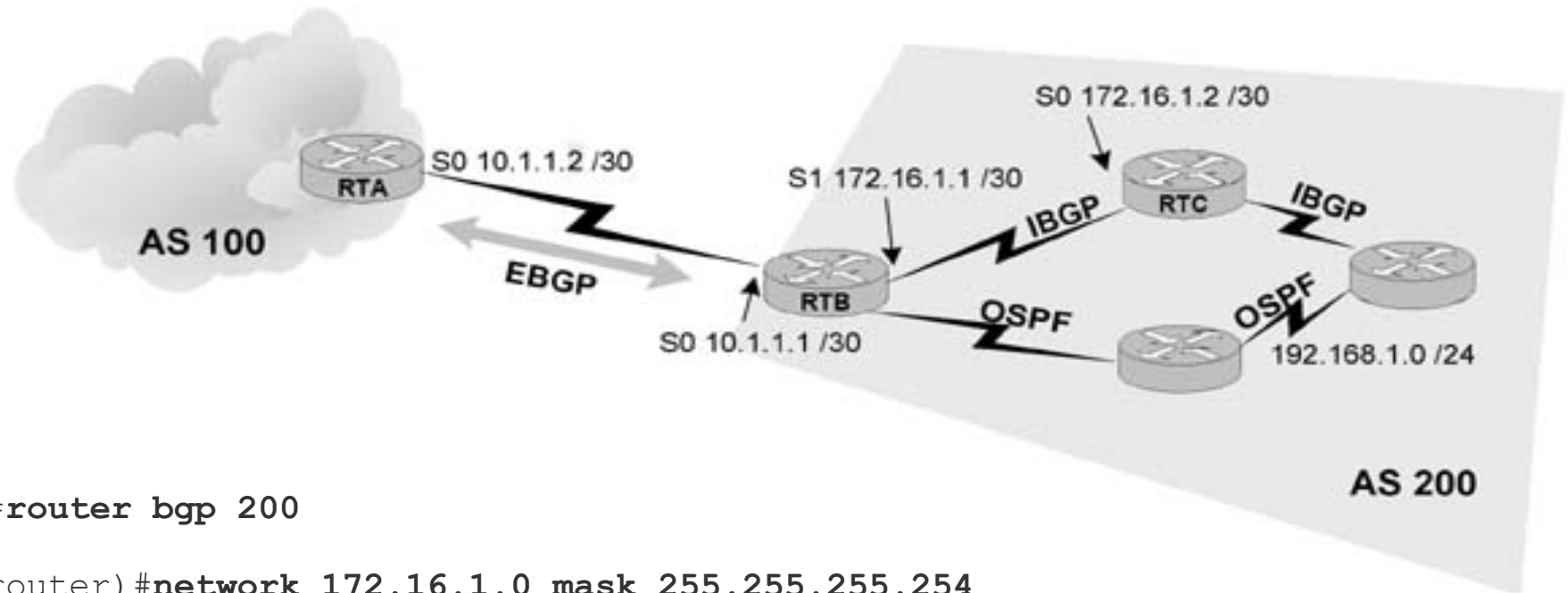
- + The **update-source loopback 0** command is used to instruct the router to use *any* operational interface for TCP connections (as long as Lo0 is up and configured with an IP address).
- + Without the **update-source loopback 0** command, BGP routers can use only the closest IP interface to the peer.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

Redistributing OSPF into BGP



- + Assume the following route appears in RTB's table:
`192.168.1.0/24 [110/74] via 10.2.2.1, 00:31:34, Serial12`
- + 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.

Redistributing OSPF into BGP



```
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
```


BGP Reset

- + Finally, whenever you are configuring BGP, you will notice that changes you make to an existing configuration may not appear immediately.
- + To force BGP to clear its table and reset BGP sessions, use the **clear ip bgp** command. The easiest way to enter this command is as follows:

```
Router#clear ip bgp *
```

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

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 ATTRIBUTES AND POLICY CISCO APPLICATION

Path Attributes

- + Much of the work you will do configuring BGP focuses on **path attributes**.
- + Each route has its own set of defined attributes, which can include path information, route preference, next-hop, and aggregation information.
- + Administrators use these values to enforce **routing policy**.
- + Based on attribute values, you can configure BGP to **filter** routing information, **prefer** certain paths, or otherwise customize its behavior.
- + Every UPDATE message has a variable-length sequence of path attributes in the form <attribute type, attribute length, attribute value>.

Path Attributes

- + Since you will use path attributes extensively when configuring routing policy, you should note that not all vendor implementations of BGP recognize the same attributes. In fact, path attributes come in four different types:
 - Well-known mandatory
 - Well-known discretionary
 - Optional transitive
 - Optional non-transitive

Path Attributes

Well-known mandatory

- + **An attribute that has to exist in the BGP UPDATE packet.**
- + It must be recognized by all BGP implementations.
- + If a well-known attribute is missing, a notification error will be generated; this ensures that all BGP implementations agree on a standard set of attributes.

Example: AS_PATH attribute.

Path Attributes

Well-known discretionary

- + An attribute that is **recognized by all BGP implementations**
- + But may or **may not be sent** in the BGP UPDATE message.

Example: LOCAL_PREF

Path Attributes

Optional transitive

- + An attribute that may or may not be, recognized by all BGP implementations (thus, **optional**).
- + Because the attribute is **transitive**, BGP **should accept and advertise** the attribute even if it isn't recognized.

Example: COMMUNITY

Path Attributes

Optional non-transitive

- + An attribute that **may or may not be, recognized** by all BGP implementations.
- + Whether or not the receiving BGP router recognizes the attribute, it is **non-transitive**, and **should not be passed along to other BGP peers**.

Example: ORIGINATOR_ID

Path Attributes

<u>Attribute Code</u>	<u>Type</u>
<u>1-ORIGIN</u>	<u>Well-known mandatory</u>
<u>2-AS_PATH</u>	<u>Well-known mandatory</u>
<u>3-NEXT_HOP</u>	<u>Well-known mandatory</u>
<u>4-MULTI_EXIT_DISC</u>	<u>Optional non-transitive</u>
<u>5-LOCAL_PREF</u>	<u>Well-known discretionary</u>
<u>6-ATOMIC_AGGREGATE</u>	<u>Well-known discretionary</u>
<u>7-AGGREGATOR</u>	<u>Well-known discretionary</u>
<u>8-COMMUNITY</u>	<u>Optional transitive (Cisco)</u>
<u>9-ORIGINATOR_ID</u>	<u>Optional non-transitive (Cisco)</u>
<u>10-Cluster List</u>	<u>Optional non-transitive (Cisco)</u>
<u>11-Destination Preference</u>	<u>(MCI)</u>
<u>12-Advertiser</u>	<u>(Baynet)</u>
<u>13-rcid_path</u>	<u>(Baynet)</u>
<u>255-Reserved</u>	<u>[md]</u>

Some BGP Attributes

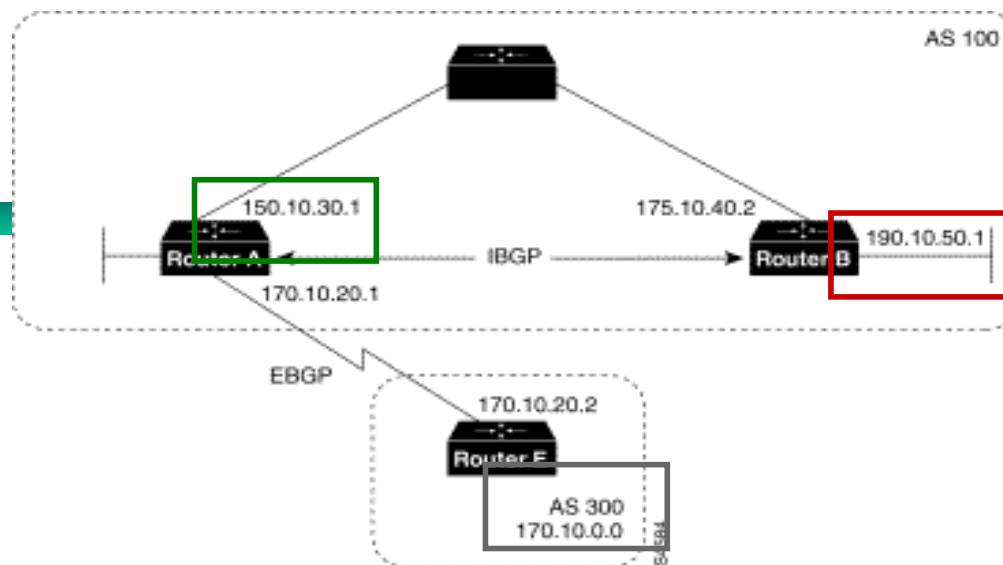
- + ORIGIN
- + NEXT_HOP
- + AS_PATH
- + LOCAL_PREF
- + **Weight**
- + MULTI_EXIT_DISC (MED)

BGP Path Selection Process

- + BGP selects only one path as the best path.
- + When the path is selected, BGP puts the selected path in its (BGP) routing table and propagates the path to its neighbors.
- + BGP uses the following criteria, in the order presented, to select a path for a destination:
 - **“We Love Oranges AS Oranges Mean Pure Refreshment”**
 - W Weight (Highest)
 - L LOCAL_PREF (Highest)
 - O Originate (local)
 - AS AS_PATH (shortest)
 - O ORIGIN Code (IGP > EGP > Incomplete)
 - M MED (lowest)
 - P Paths (External > Internal)
 - R RID (lowest)
 - <http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html>

The ORIGIN attribute

- + Well-known mandatory attribute (type code 1)
- + Indicates the origin of the routing update
 - **IGP**: The prefix is internal to the originating AS.
 - **EGP**: The prefix was learned via some EGP, such as BGP.
 - **INCOMPLETE**: The prefix was learned by some other means, probably redistribution.
- + 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.



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 the `set origin` command to manipulate the ORIGIN attribute.

```
route-map SETORIGIN permit 10
```

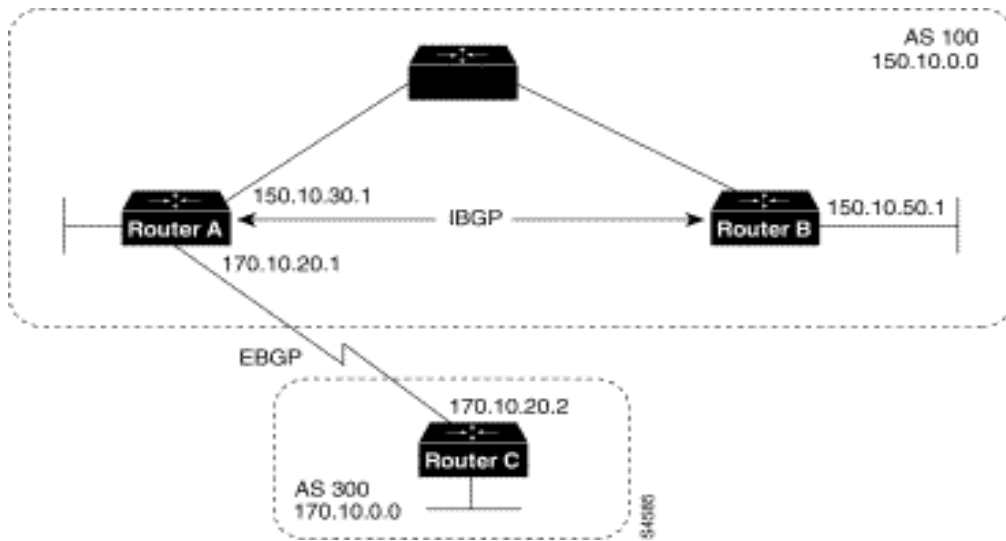
```
    set origin igp
```


NEXT_HOP

- + 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.

NEXT_HOP

- + 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.

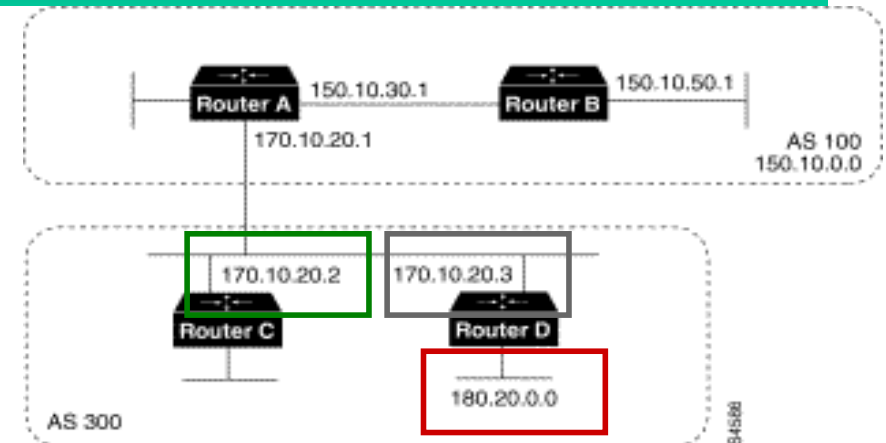
- For that reason, the configuration must ensure that Router B can reach 170.10.20.2 via an IGP.

- 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.

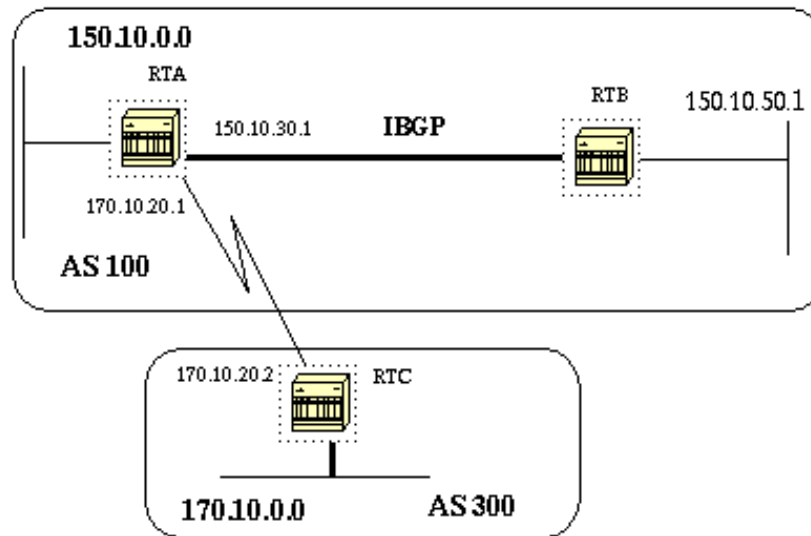
- You might want to make IGRP passive on the link to Router C so that only BGP updates are exchanged.

Next Hop Attribute and Multiaccess Media



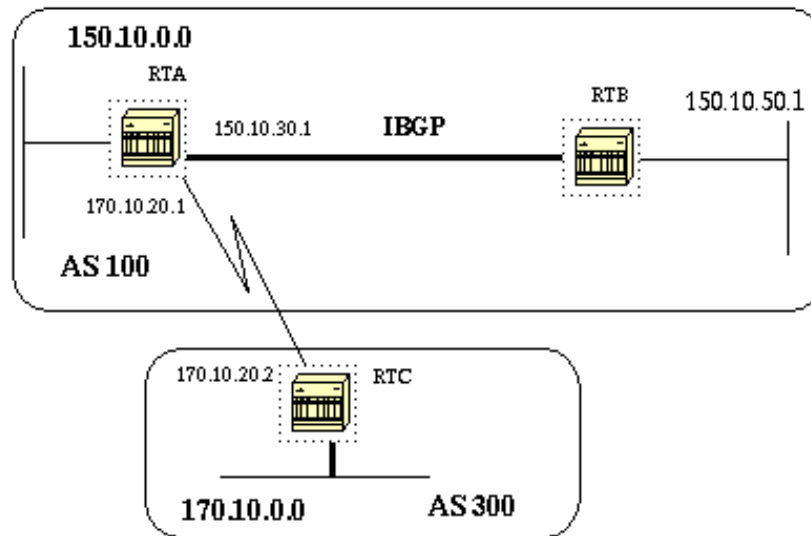
- Routers C and D are in AS 300 are running OSPF.
- Router C is running BGP with Router A.
- Router C can reach network **180.20.0.0** via **170.10.20.3**.
- When Router C sends a BGP update to Router A regarding **180.20.0.0**, it sets the next hop attribute to **170.10.20.3**, instead of its own IP 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-SELF



- RTC will advertise 170.10.0.0 to RTA with a next hop of 170.10.20.2 and RTA will advertise 150.10.0.0 to RTC with a next hop of 170.10.20.1.
- For IBGP, the protocol states that the next hop advertised by EBGP should be carried into IBGP.
- Because of that rule, RTA will advertise 170.10.0.0 to its IBGP peer RTB with a next hop of 170.10.20.2. So according to RTB, the next hop to reach 170.10.0.0 is 170.10.20.2 and NOT 150.10.30.1.
- You should make sure that RTB can reach 170.10.20.2 via IGP, otherwise RTB will drop packets destined to 170.10.0.0 because the next hop address would be inaccessible.

NEXT-HOP-SELF



- **neighbor {ip-address|peer-group-name} next-hop-self**
- The next-hop-self command will allow us to force BGP to use a specified IP address as the next hop rather than letting the protocol choose the nexthop.
 - **RTA**
 - **router bgp 100**
 - **neighbor 150.10.50.1 remote-as 100**
 - **neighbor 150.10.50.1 next-hop-self**
- RTA will relay BGP advertise for 170.10.0.0 to RTB with a NextHop = 150.10.30.1, known by RTB by a IGP (e.g. OSPF)

Route Maps

Route maps are similar to a scripting language for these reasons:

- + They work like a more sophisticated access list:
 - Top-down processing
 - Once there is a match, leave the route map

- + Lines are sequence-numbered for easier editing:
 - Insertion of lines
 - Deletion of lines

- + Route maps are named rather than numbered for easier documentation.

- + Match criteria and set criteria can be used, similar to the “if, then” logic in a scripting language.

Route Map Applications

The common uses of route maps are as follows:

+ Redistribution route filtering:

- A more sophisticated alternative to distribute lists

+ Policy-based routing:

- The ability to determine routing policy based on criteria other than the destination network

+ BGP policy implementation:

- The primary tool for defining BGP routing policies

Route Map Operation

- + A list of statements composes a route map.
- + The list is processed top-down like an access list.
- + The first match found for a route is applied.
- + The sequence number is used for inserting or deleting specific route map statements.
- + **implicit deny at the end of each route map**
- + Statement with no explicit match, imply an implicit match *any*

```
route-map my_bgp permit 10
    { match statements }
    { match statements }
    { set statements }
    { set statements }
route-map my_bgp deny 20
    ::          ::  ::
    ::          ::  ::
route-map my_bgp permit 30
    ::          ::  ::
    ::          ::  ::
```

Route Map Operation (Cont.)

- The match statement may contain multiple references.
- Multiple match criteria in the same line use a logical OR.
- At least one reference must permit the route for it to be a candidate for redistribution.

```
route-map my_bgp permit 10
match ip address x y z
                ───────────▶
                Logical OR
```

```
route-map my_bgp deny 20
Logical AND ↓ match ...a
              match ...b
              match ...c
```

014G_330

- Each vertical match uses a logical AND.
- All match statements must permit the route for it to remain a candidate for redistribution.
- Route map permit or deny determines if the candidate will be redistributed.

route-map Commands

`router (config) #`

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

- Defines the route map conditions

`router (config-route-map) #`

```
match {conditions}
```

- Defines the conditions to match

`router (config-route-map) #`

```
set {actions}
```

- Defines the action to be taken on a match

`router (config-router) #`

```
redistribute protocol [process id] route-map map-tag
```

- Allows for detailed control of routes being redistributed into a routing protocol

The match Command

- The `match` commands specify criteria to be matched.
- The associated route map statement permits or denies the matching routes.

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

```
match {options}

  options :
  ip address ip-access-list

  as-path as-path-access-list
  ip route-source ip-access-list
  ip next-hop ip-access-list
  interface type number
  metric metric-value
  route-type [external | internal | level-1 | level-2 | local]

  ...
```

The set Command

- The `set` commands modify matching routes.
- The command modifies parameters in redistributed routes.

`router (config-route-map) #`

```
set {options}
  options :
  metric metric-value
  metric-type [type-1 | type-2 | internal | external]
  level [level-1 | level-2 | level-1-2 | stub-area | backbone]
  ip next-hop next-hop-address
```

The set commands

Command	Description
<code>set as-path</code>	Modifies an AS path for BGP routes
<code>set automatic-tag</code>	Computes automatically the tag value
<code>set community</code>	Sets the BGP communities attribute
<code>set default interface</code>	Indicates where to output packets that pass a match clause of a route map for policy routing and have no explicit route to the destination
<code>set interface</code>	Indicates where to output packets that pass a match clause of a route map for policy routing
<code>set ip default next-hop</code>	Indicates where to output packets that pass a match clause of a route map for policy routing and for which the Cisco IOS software has no explicit route to a destination
<code>set ip next-hop</code>	Indicates where to output packets that pass a match clause of a route map for policy routing
<code>set level</code>	Indicates where to import routes for IS-IS and OSPF
<code>set local-preference</code>	Specifies a BGP local preference value
<code>set metric</code>	Sets the metric value for a routing protocol
<code>set metric-type</code>	Sets the metric type for the destination routing protocol
<code>set tag</code>	Sets tag value for destination routing protocol
<code>set weight</code>	Specifies the BGP weight value

Regular Expressions

- + A **regular expression** is a pattern to match against an input string.
- + The input string, in the case of the **ip as-path access-list** command, is the **AS_PATH attribute**.
- + Once you specify a pattern (or patterns) using this command, the router tests BGP routes to see if the AS_PATH attribute matches the pattern or not.
- + For example, the following command will match any AS_PATH that includes 2150:

```
Router(config)#ip as-path access-list 1 permit 2150
```

Or

```
Router# show ip bgp regexp 2150
```

- + Unfortunately, the regular expression, 2150, will match not only AS 2150, but also 12150, 21502, 21503, etc.
- + Because policy routing demands a certain degree of precision, you will typically use one or more these special characters when creating a regular expression.

Regular Expressions

- + **^** : Matches the beginning of the input string.
- + **\$** : Matches the end of the input string.
- + **_** : Matches a space, comma, left brace, right brace, the beginning of an input string, or the ending of an input stream
- + **.** : Match any single character
- + ***** : Matches 0 or more single or multiple character pattern

Regular Expressions

- + Thus, if you want to match an AS_PATH that contains AS 2150 somewhere in the string, you would use the regular expression:

`_2150_`

- + If you want to match AS 2150, but only if it appears at the beginning of the AS_PATH, you would use this regular expression.
- + You match previous AS using `^`, since it is at the leftmost side of the AS path.

`^2150_`

- + Similarly, you can match an AS_PATH that ends with 2150, which means that the route originated at AS 2150:
- + You match originating AS using `$`, since it is at the rightmost side of the AS path.

`_2150$`

- + void AS_PATH (AS_PATH for locally generated routes)

`^$`

- + AS_PATH with only 2150

`^2150$`

as-path access-list

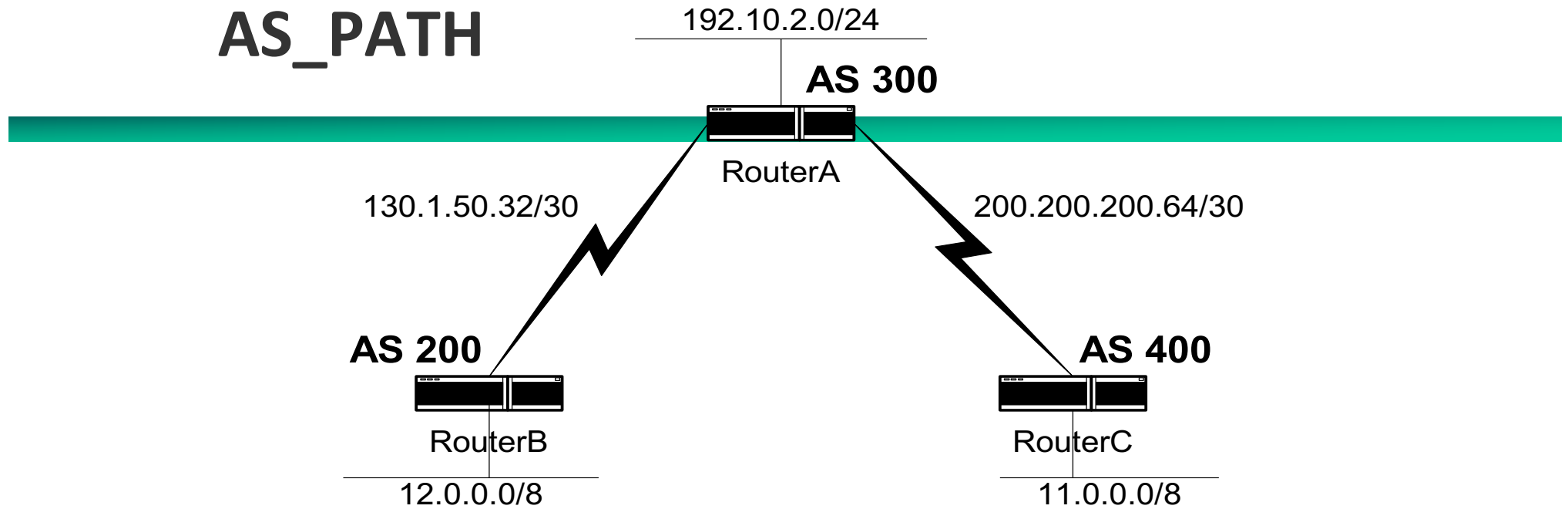
- + Used to specify a match on the AS_PATH attributes by means of a regular expression

```
ip as-path access-list [number] permit [regular  
expression]
```

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

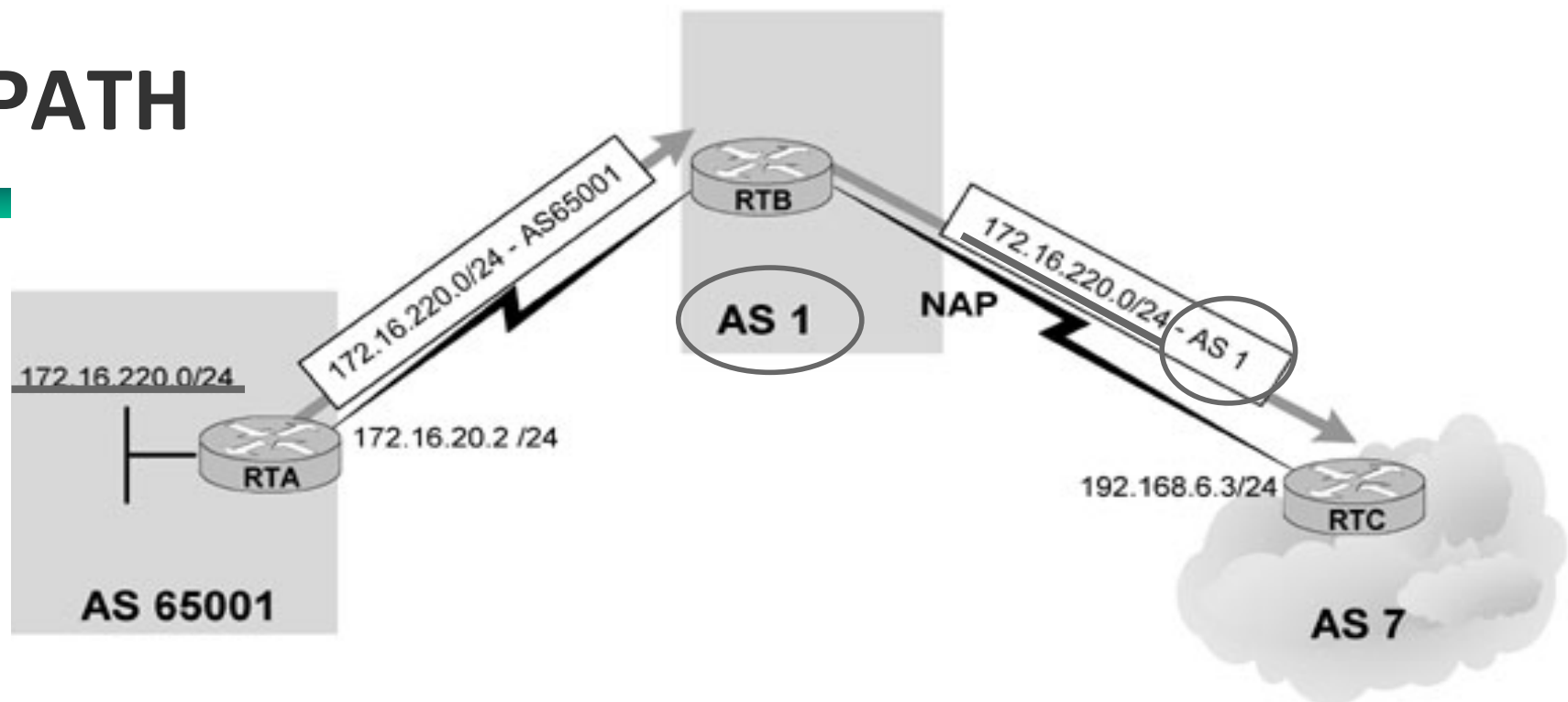
AS_PATH – private AS numbers

- + 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

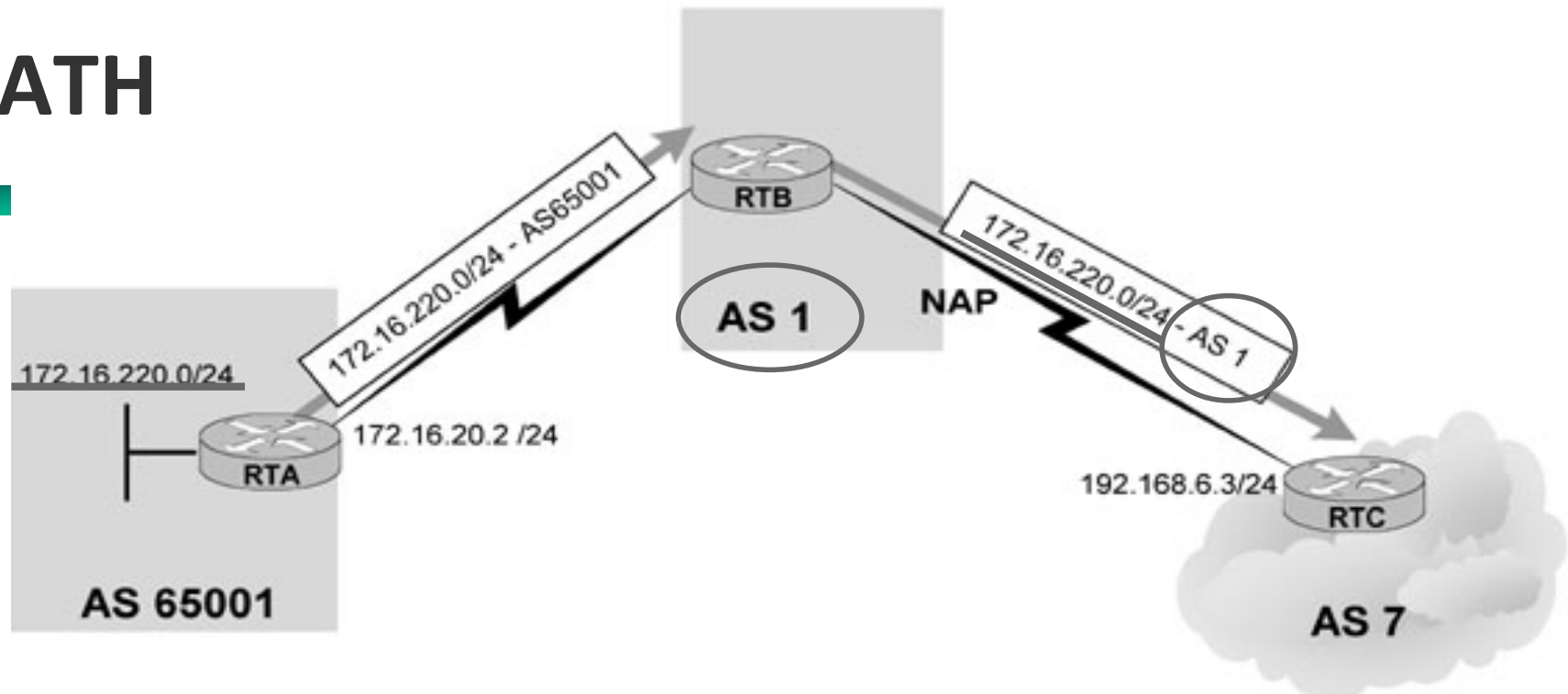
- + **AS_PATH** information is one of the attributes BGP looks at to determine the best route to take to get to a destination.
 - In comparing two or more different routes, given that all other attributes are identical, a shorter path is always preferred.
 - In case of a tie in AS_PATH length, other 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



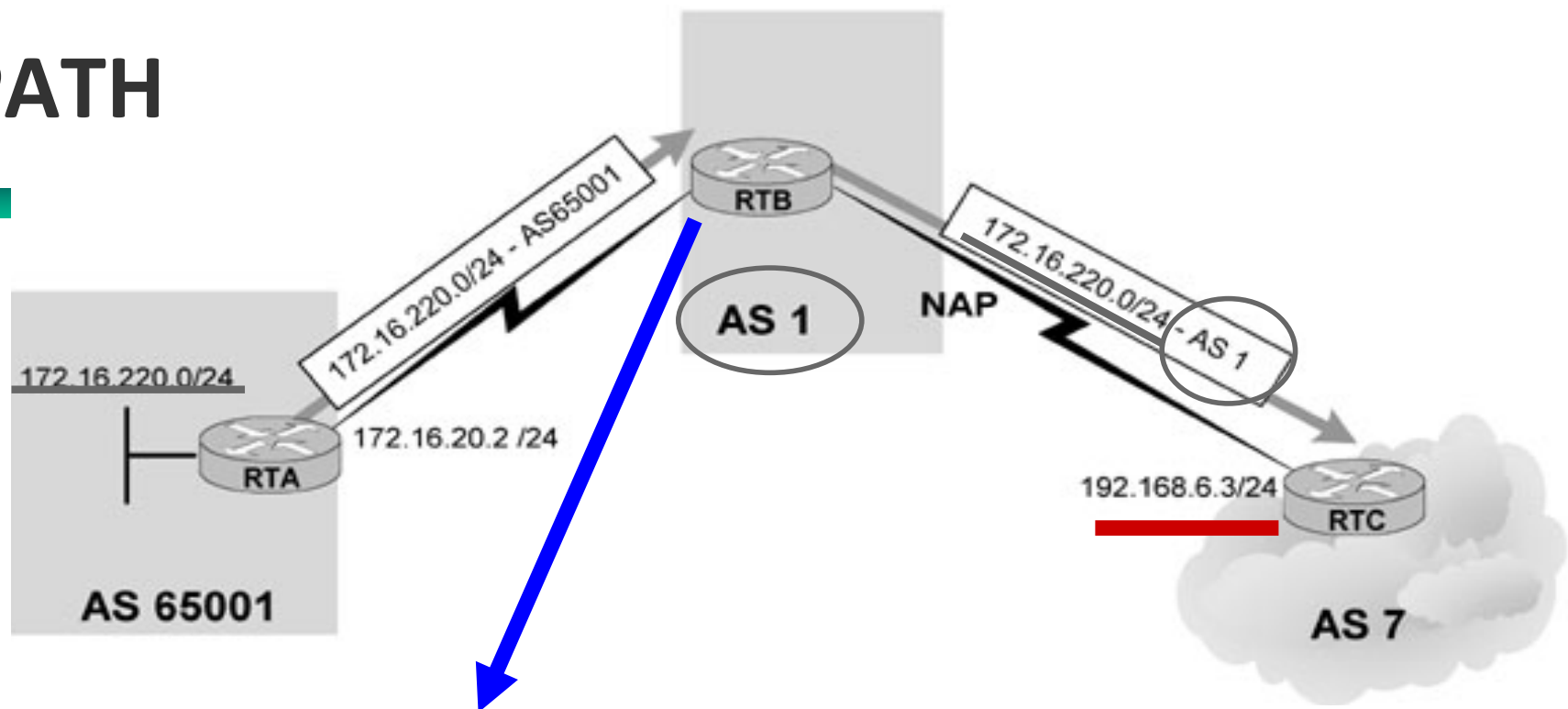
- + AS1 is providing Internet connectivity to its customer AS 65001.
- + Because the customer connects to only this provider and no plans to connect to an additional provider in the near future, the customer has been allocated a private AS number.
- + **BGP will strip private AS numbers** only when propagating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

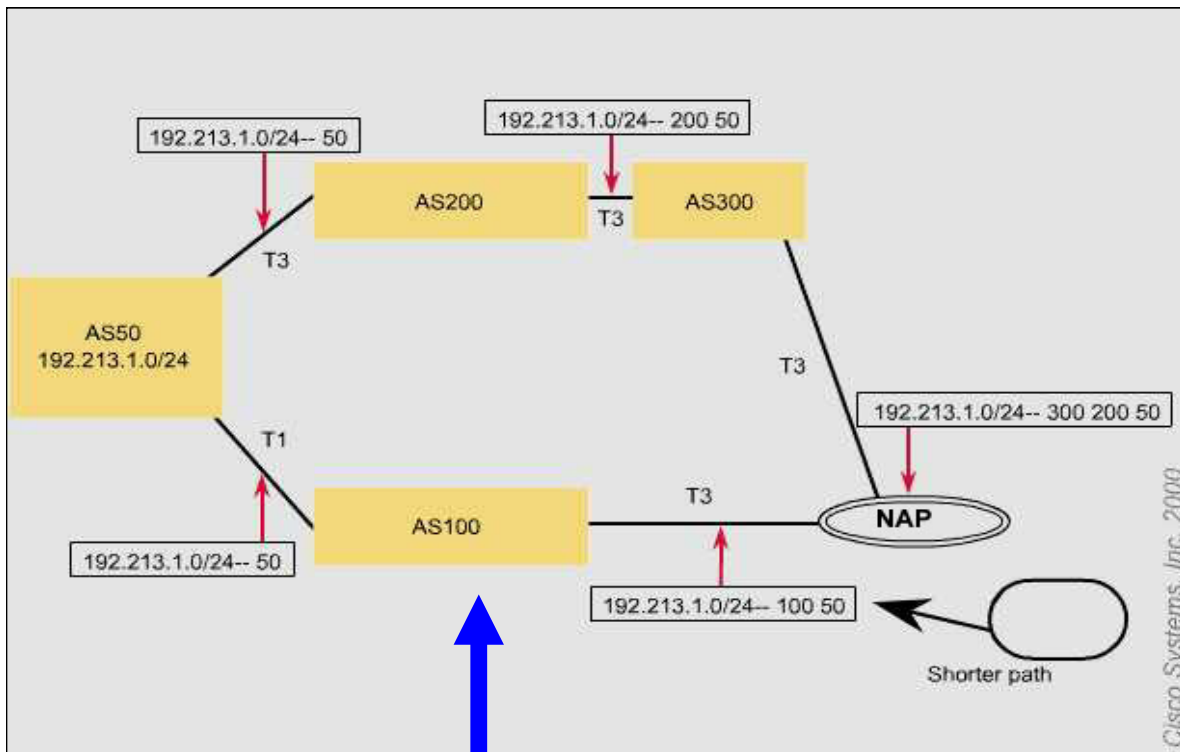
AS_PATH - prepend

- + **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.

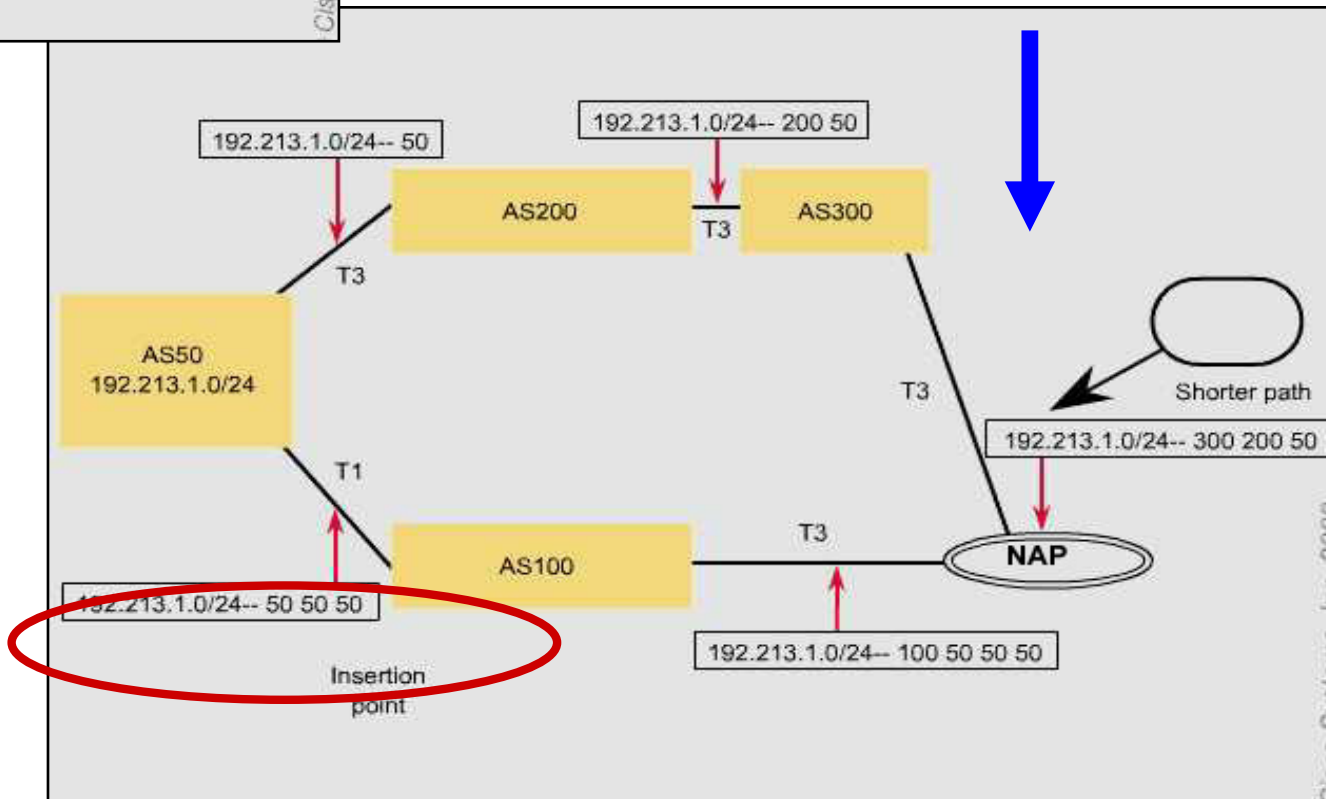
AS_PATH – prepend

Concept

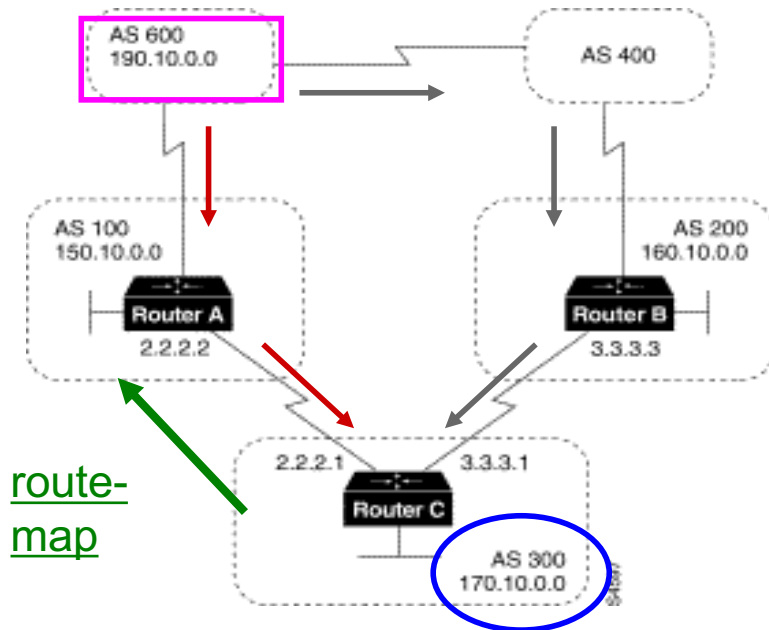
New “shorter path”



Current “shorter path”



AS_PATH – prepend - Example



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

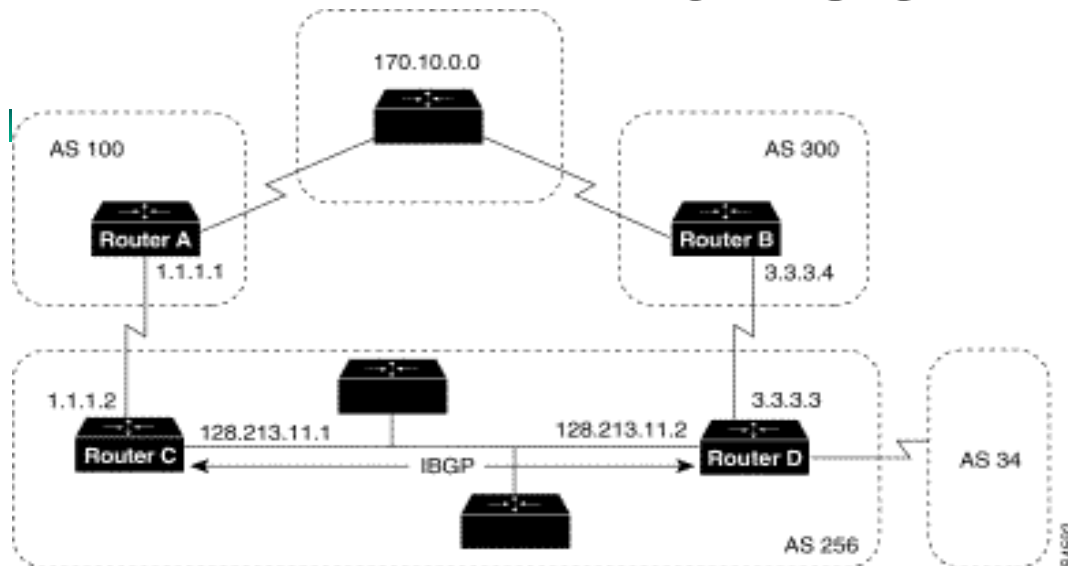
route-map SETPATH permit 10
 set as-path prepend 300 300
```

- + 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 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.

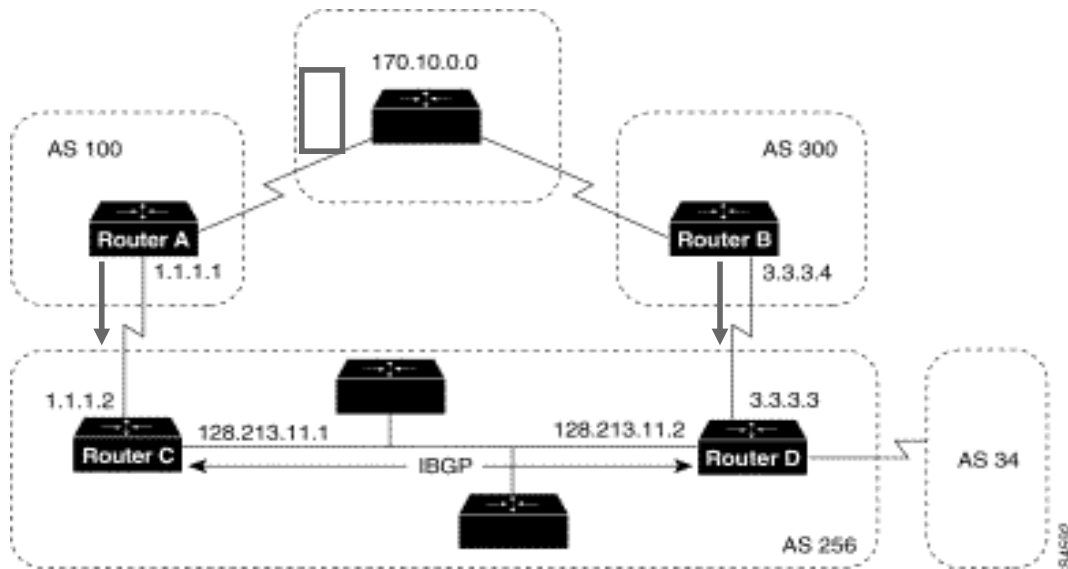
The LOCAL_PREF Attribute



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.**

The LOCAL PREF Attribute



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
```

As a result, all traffic in AS 256 destined for network 170.10.0.0 is sent to Router D as the exit point

Using a Route Map to Set Local Preference

- + **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**.


Using a Route Map to Set Local Preference

- + 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 access-list 7 permit ^300$
route-map SETLOCALIN permit 10
  match as-path 7
  set local-preference 200
route-map SETLOCALIN permit 20
```

Last Statement with no match permits all other UPDATES to be accepted.
Without this line all other than as-path 7 will be blocked



- + 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.

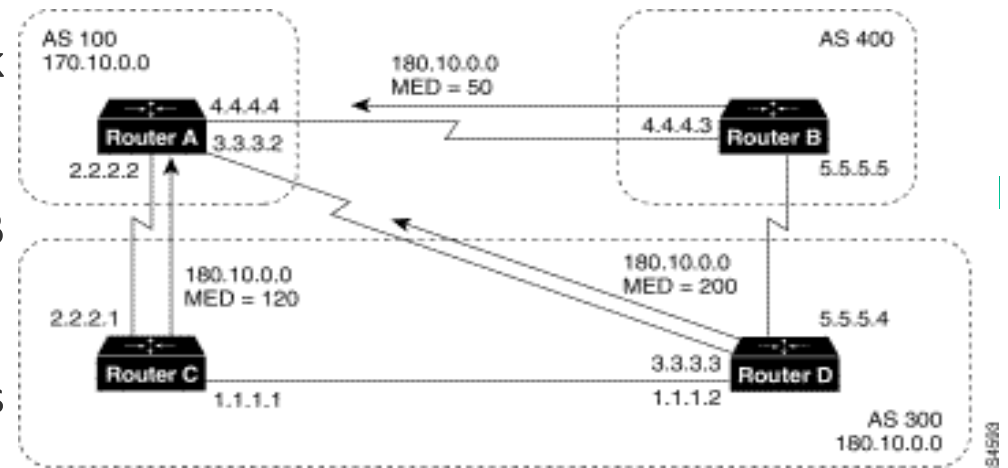
The MED attribute

- + 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.
- + When a BGP speaker learns a route from a peer, **the route's MED is passed to other interior BGP (iBGP) peers, but not to exterior BGP (eBGP) peers.**

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 ASes**, 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.
- + With following conf, Router A prefers routes from router B



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
```

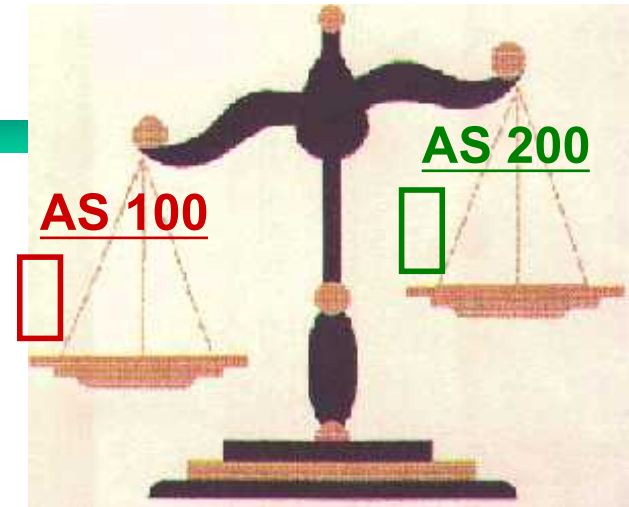
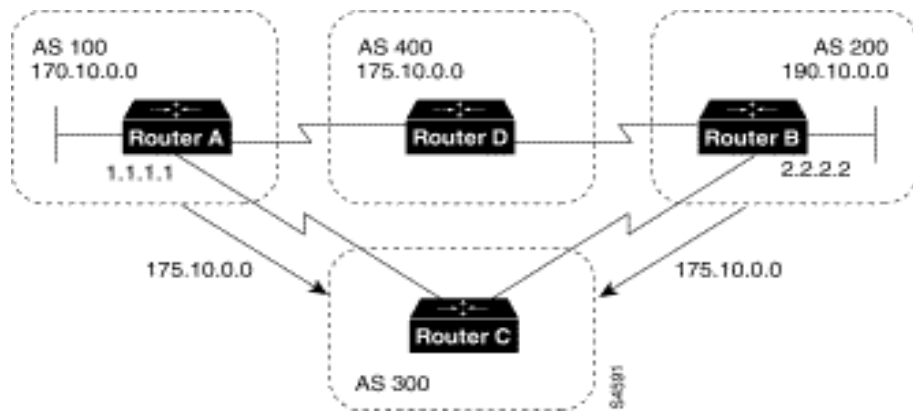
Router C

```
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
```

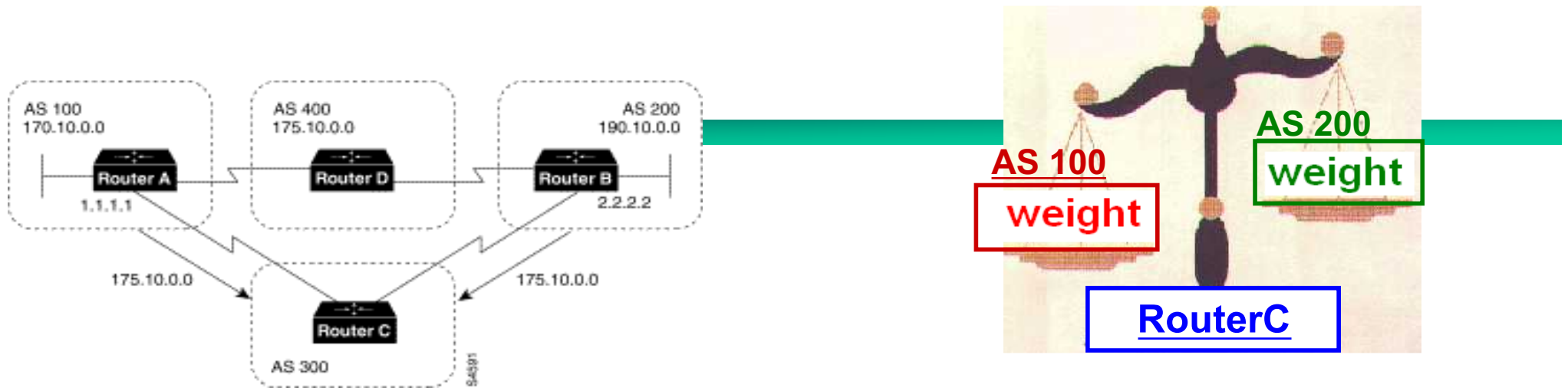
Router D

```
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
```

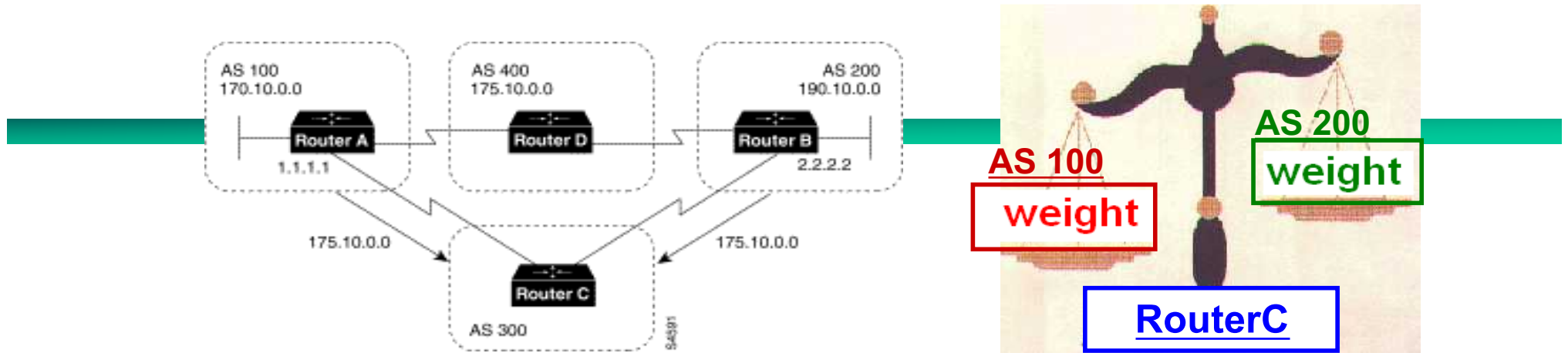
The WEIGHT attribute



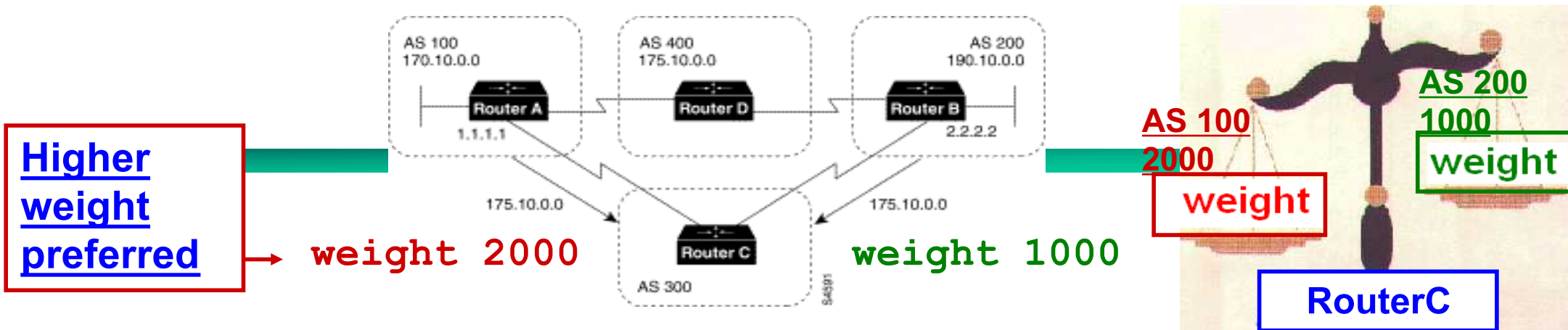
- + The weight attribute is a **special Cisco attribute** that is used in the path selection process **when there is more than one route to the same destination**.
- + The weight attribute is **local to the router on which it is assigned**, and it is **not propagated** in routing updates.
- + By **default**,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



- + **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**.



- + 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 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**.

Using a Route Map to Set the Weight Attribute

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
```

BGP Policies in Practice

Business Relationships

+ Common relationships

- Customer-provider
- Peer-peer
- Backup, sibling, ...

+ Implementing in BGP

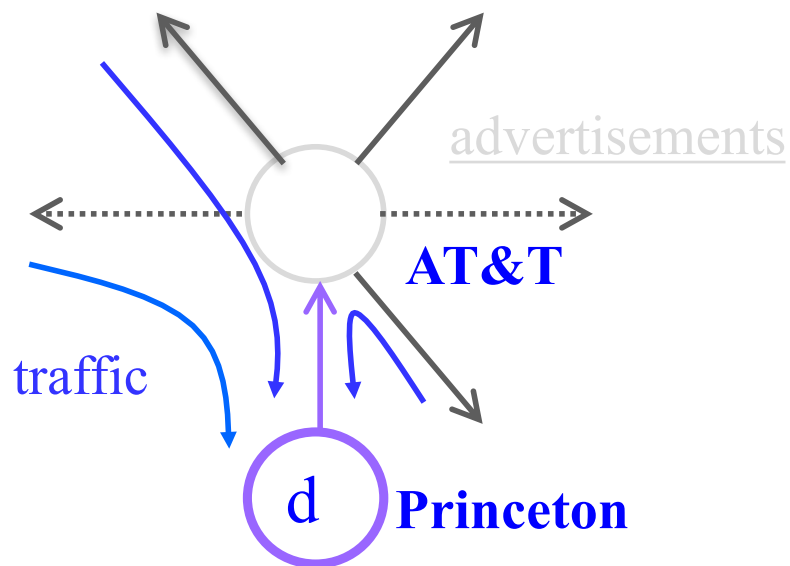
- Import policy
 - Ranking customer routes over peer routes
- Export policy
 - Export only customer routes to peers and providers

Customer-Provider Relationship

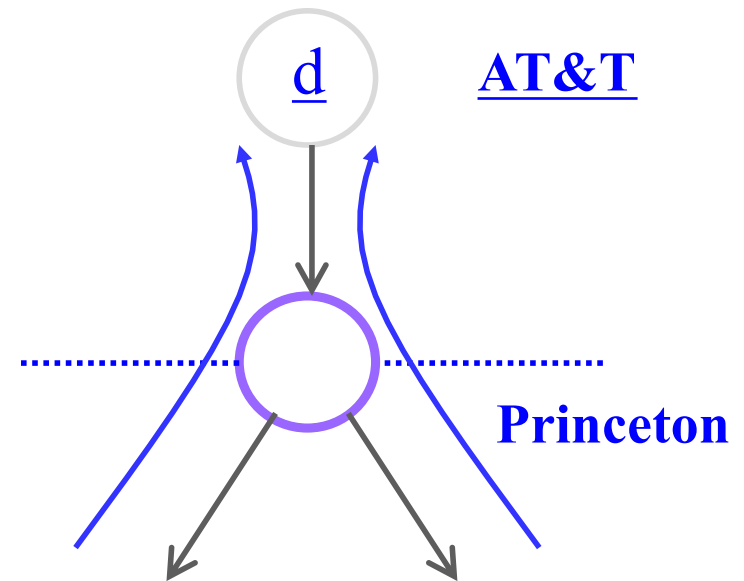
+ Customer pays provider for access to Internet

- Provider exports customer's routes to everybody
- Customer exports provider's routes to customers

Traffic to the customer



Traffic from the customer

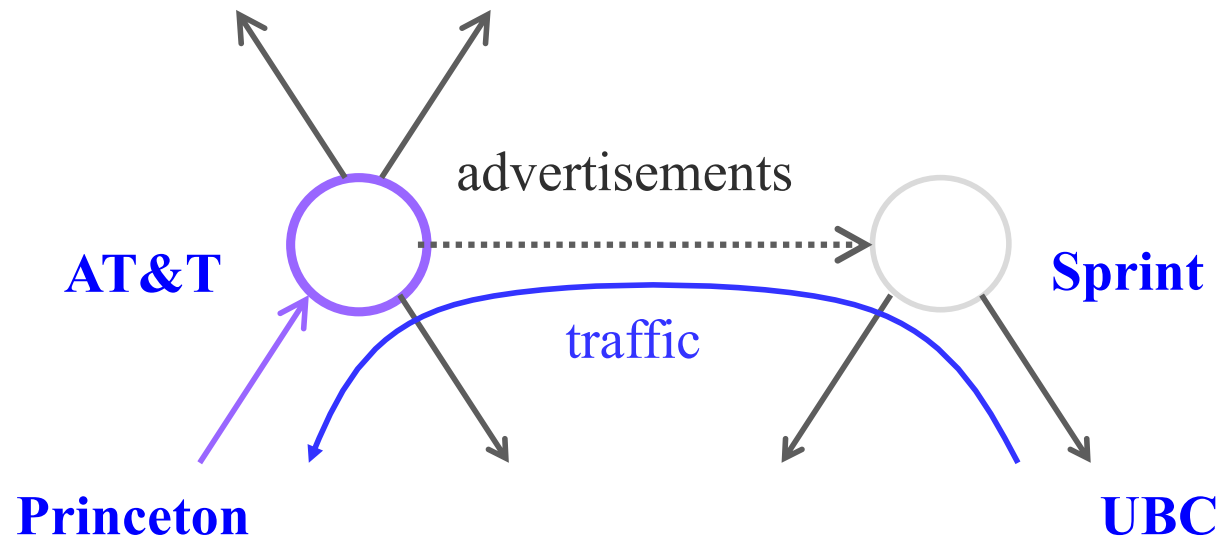


Peer-Peer Relationship

+ Peers exchange traffic between customers

- AS exports *only* customer routes to a peer
- AS exports a peer's routes *only* to its customers

Traffic to/from the peer and its customers



How Peering Decisions are Made?

Peer

- + Reduces upstream transit costs
- + Can increase end-to-end performance
- + May be the only way to connect your customers to some part of the Internet (“Tier 1”)

Don't Peer

- + You would rather have customers
- + Peers are usually your competition
- + Peering relationships may require periodic renegotiation

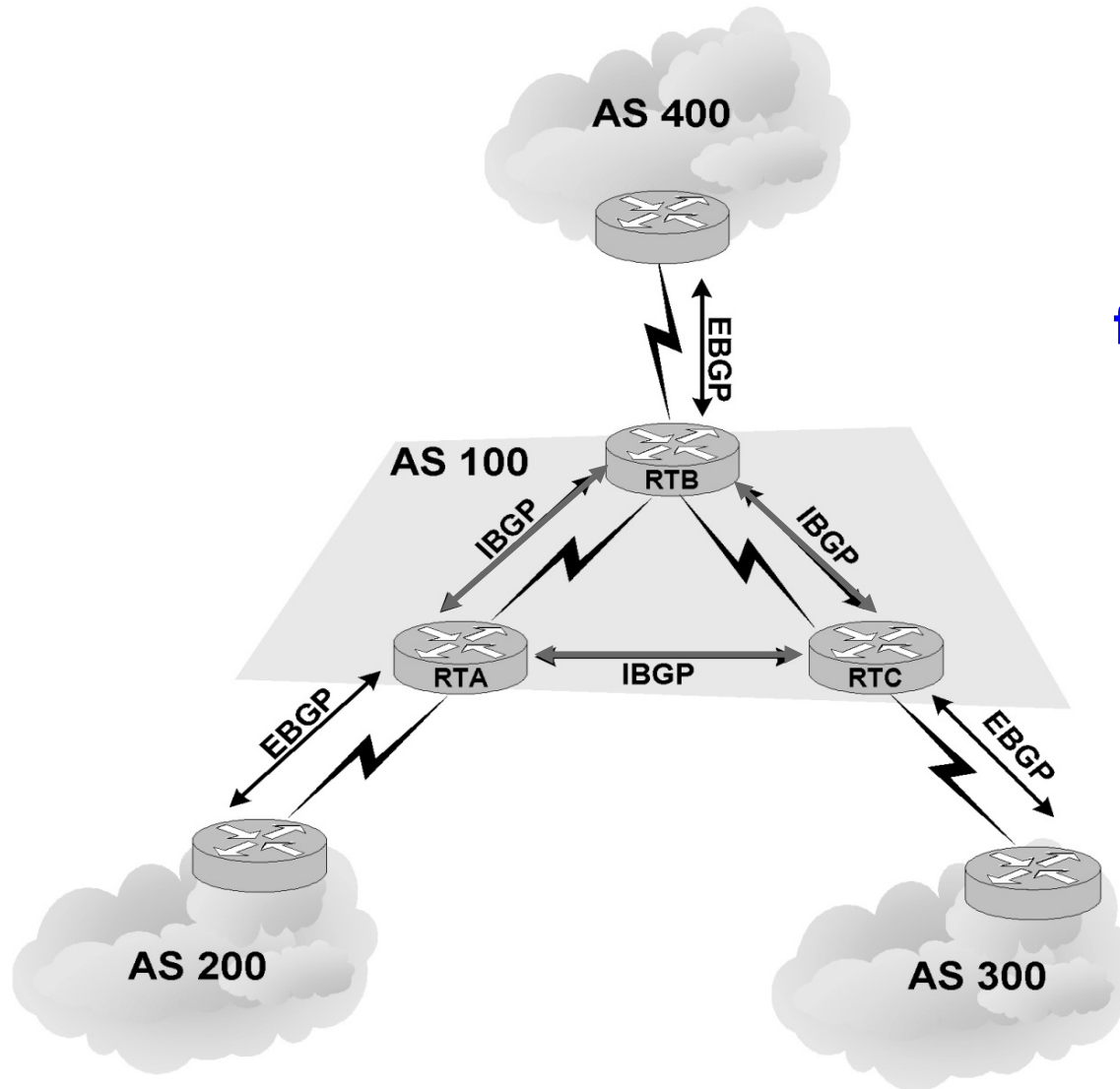


SCALING BGP

Route Reflector (RR)

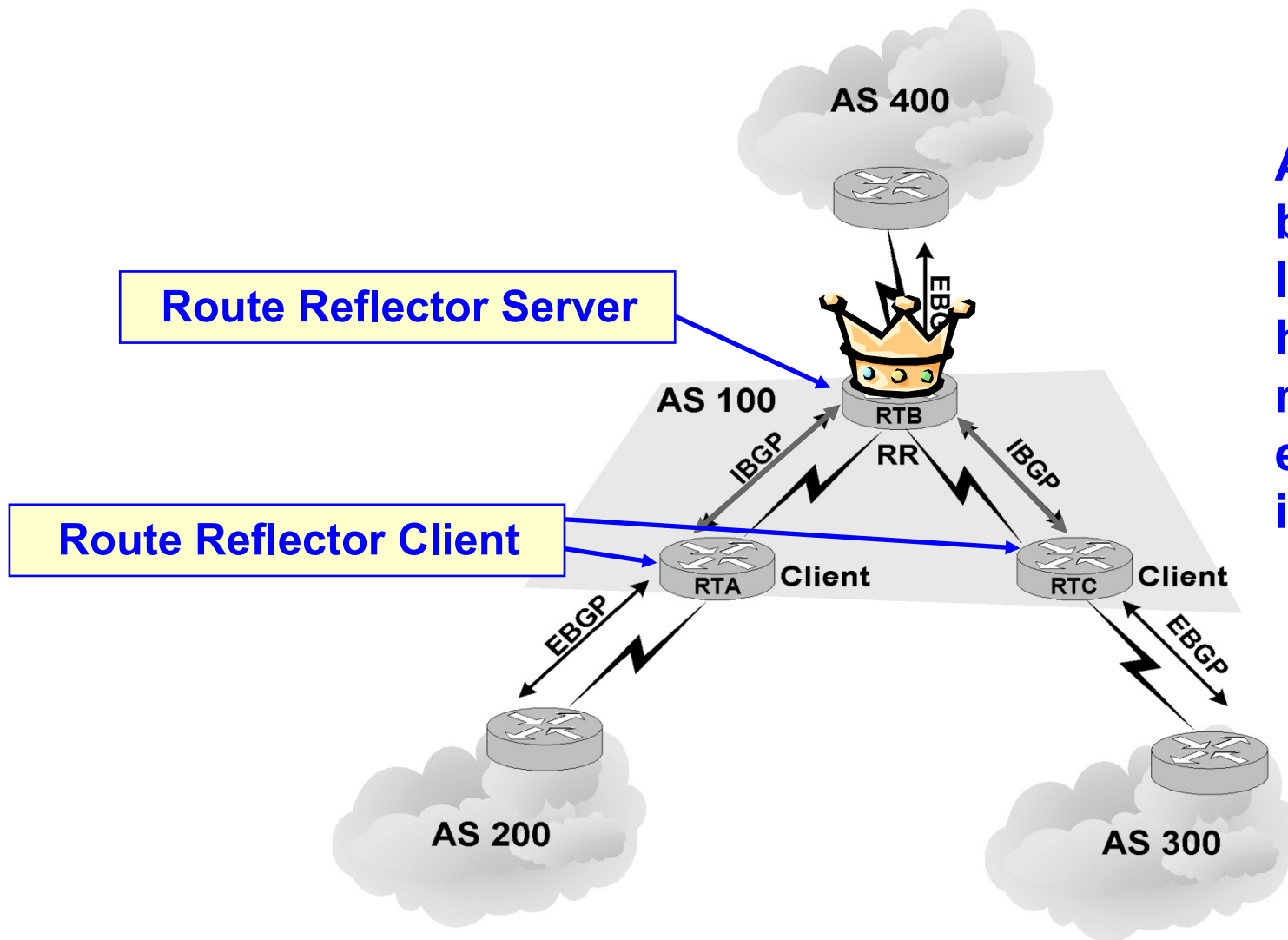
- + Offers an **alternative to the logical full-mesh requirement** of IBGP.
- + **Acts as a focal point for IBGP sessions.**
 - Multiple BGP routers can peer with a central point (the RR), rather than peer with every other router in a full mesh.
 - similar to OSPF's DR/BDR feature
- + Provides large ISPs with added BGP scalability.
- + The use of route reflectors is recommended only for autonomous systems that support a large internal BGP mesh, on the order of more than 100 sessions per router.
 - Introduces processing overhead on the routers that act as route reflectors
 - If configured incorrectly, can cause routing loops and instability.

Route Reflector (RR)



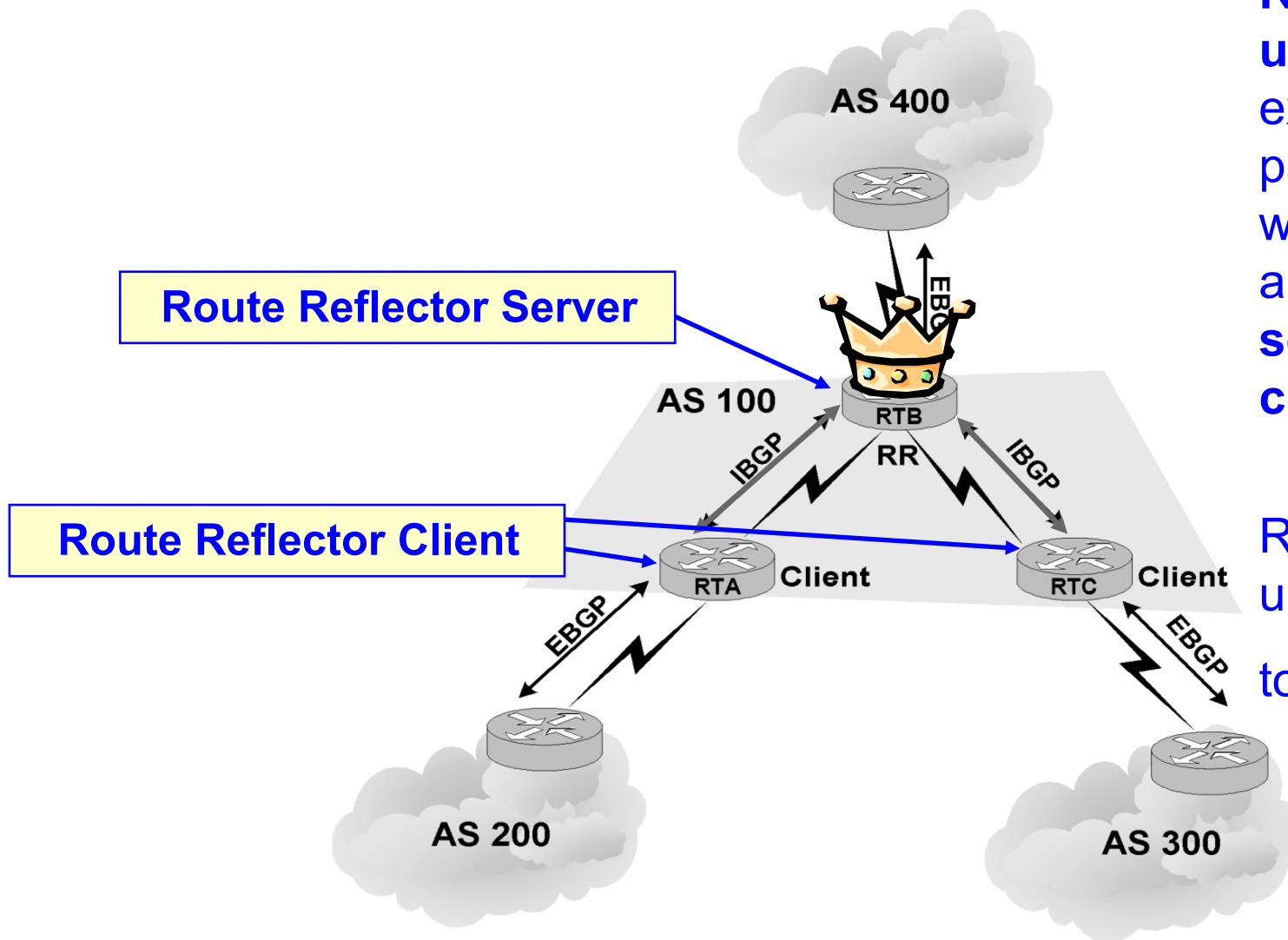
**IBGP routers
are typically
fully meshed.**

Route Reflector (RR)



A route reflector can be configured so that IBGP routers don't have to be in a full mesh to completely exchange routing information.

Route Reflector (RR)



RTA receives an update from an external peer and passes it on to **RTB**, which is configured as a **route reflector server** with **two clients, RTA and RTC.**

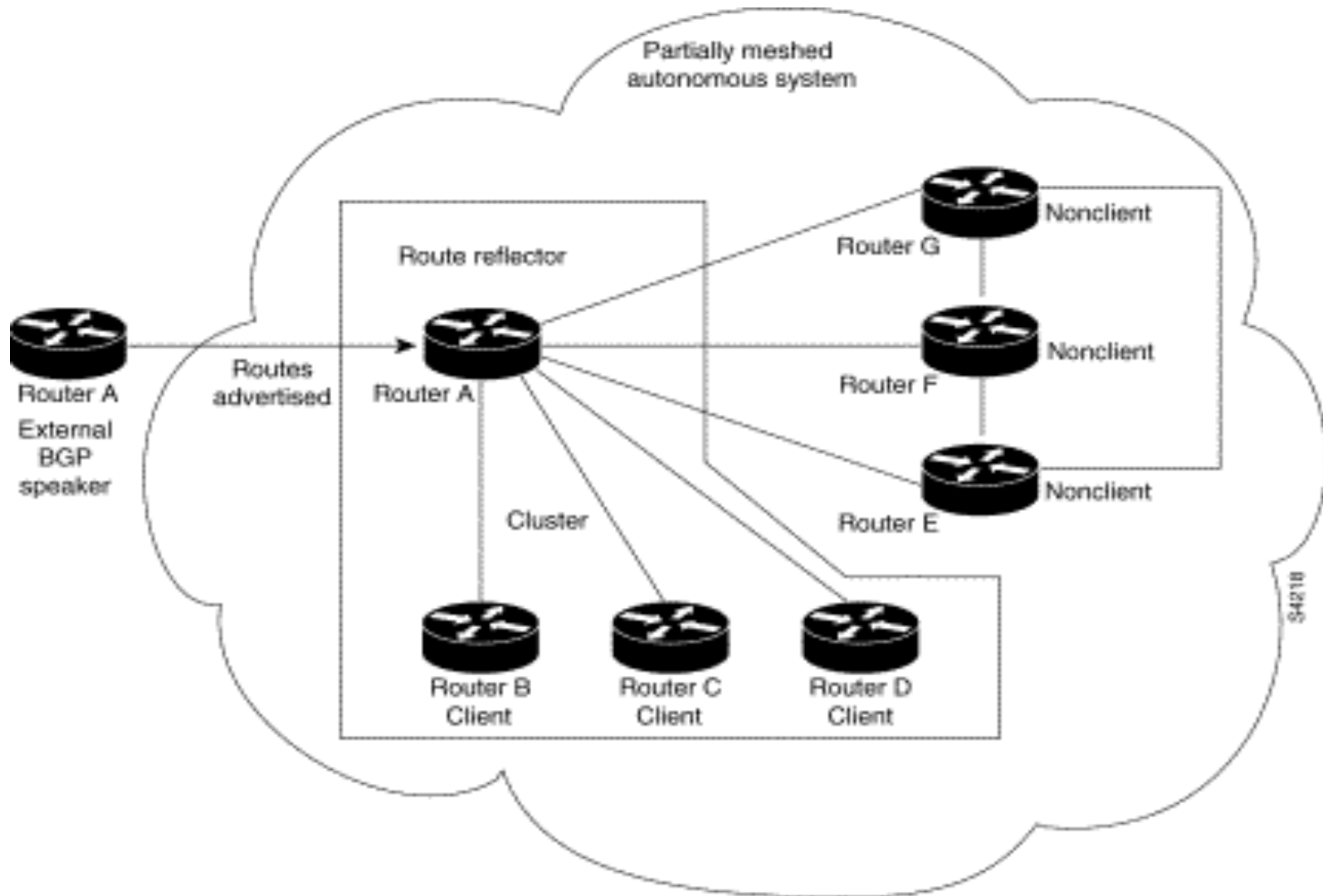
RTB will reflect the update from client RTA to client RTC.

Route Reflector (RR)

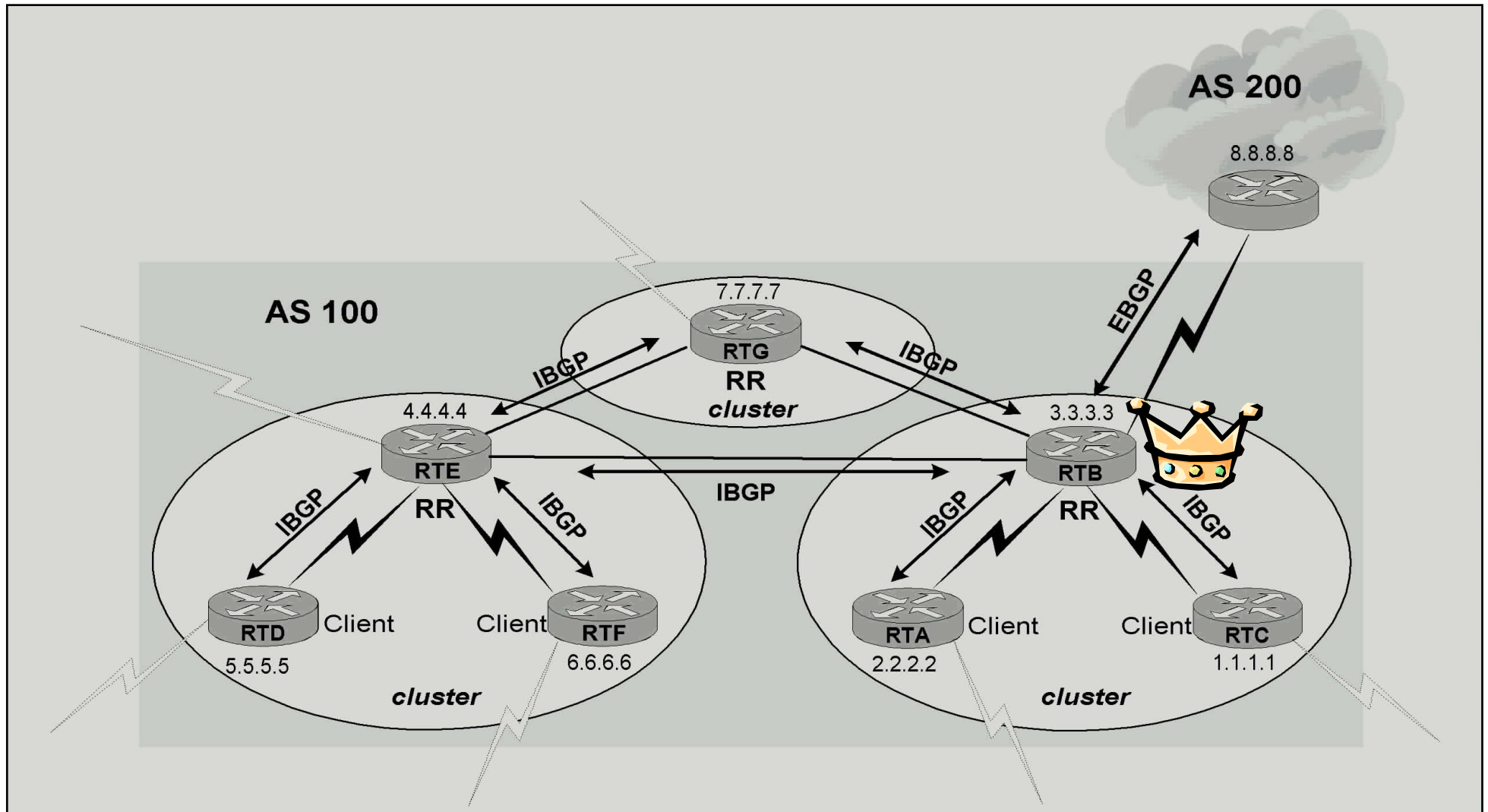
- + The IBGP peers of a route reflector fall under two categories:
 - **Clients**
 - **Nonclients**
- + A route reflector and its clients form a **cluster**.
- + All **IBGP peers** of the route reflector that are **not part of the cluster are nonclients** and must be fully meshed to all other nonclients and RR servers.
 - Never configure route reflector clients to peer with IBGP speakers outside their cluster.
- + Clients and nonclients don't even know that route reflection is occurring.
- + To identify clients and clusters, use the **neighbor** command, which has the following syntax, on the **route reflector server**:

```
Router(config-router) #neighbor      IP-address      route-  
                        reflector-client
```

Route Reflector (RR)



Route Reflector (RR)



Configuring a RR server:

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

```
RTB (config-router) #neighbor 1.1.1.1 remote-as 100
```

```
RTB (config-router) #neighbor 1.1.1.1 route-reflector-client
```

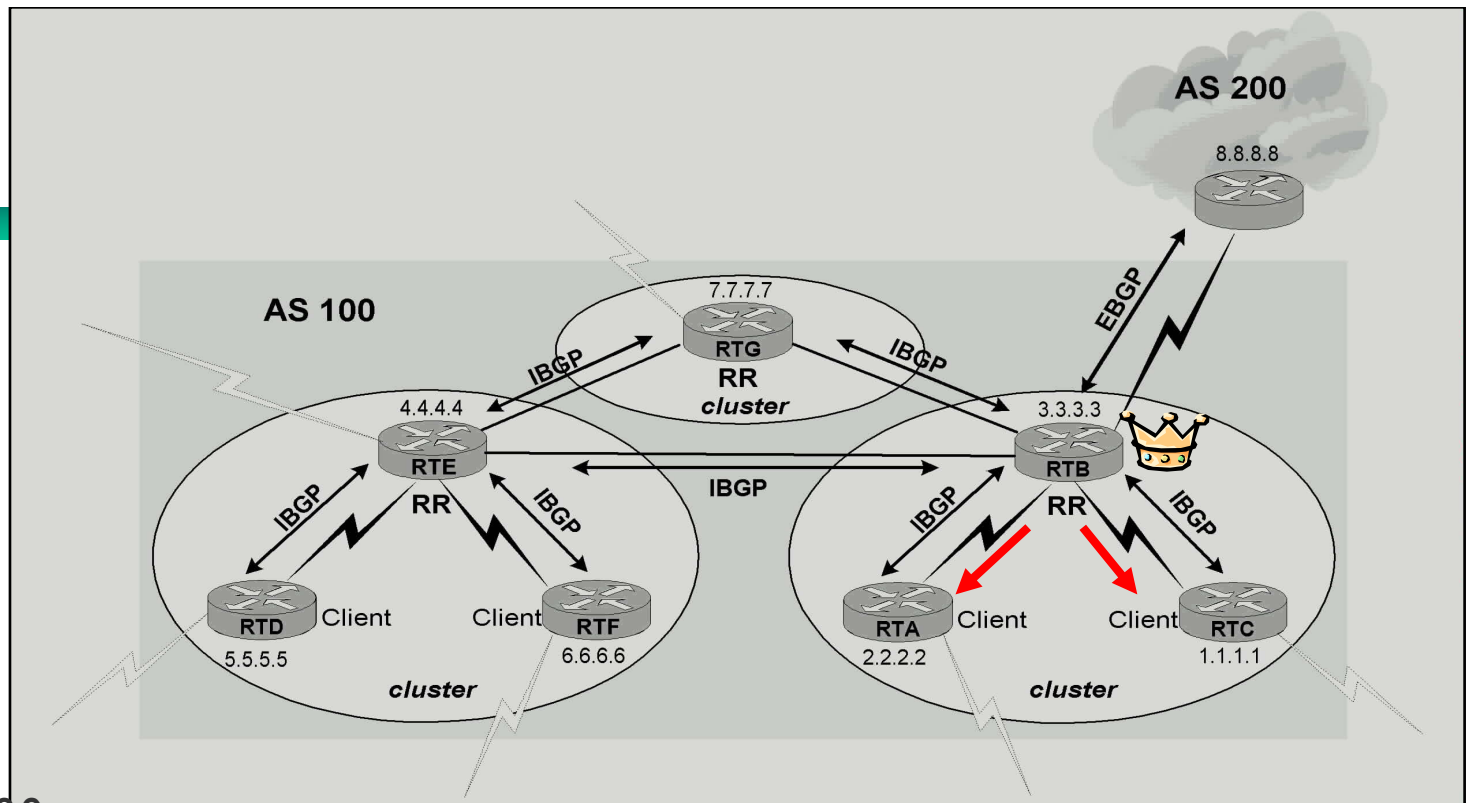
```
RTB (config-router) #neighbor 2.2.2.2 remote-as 100
```

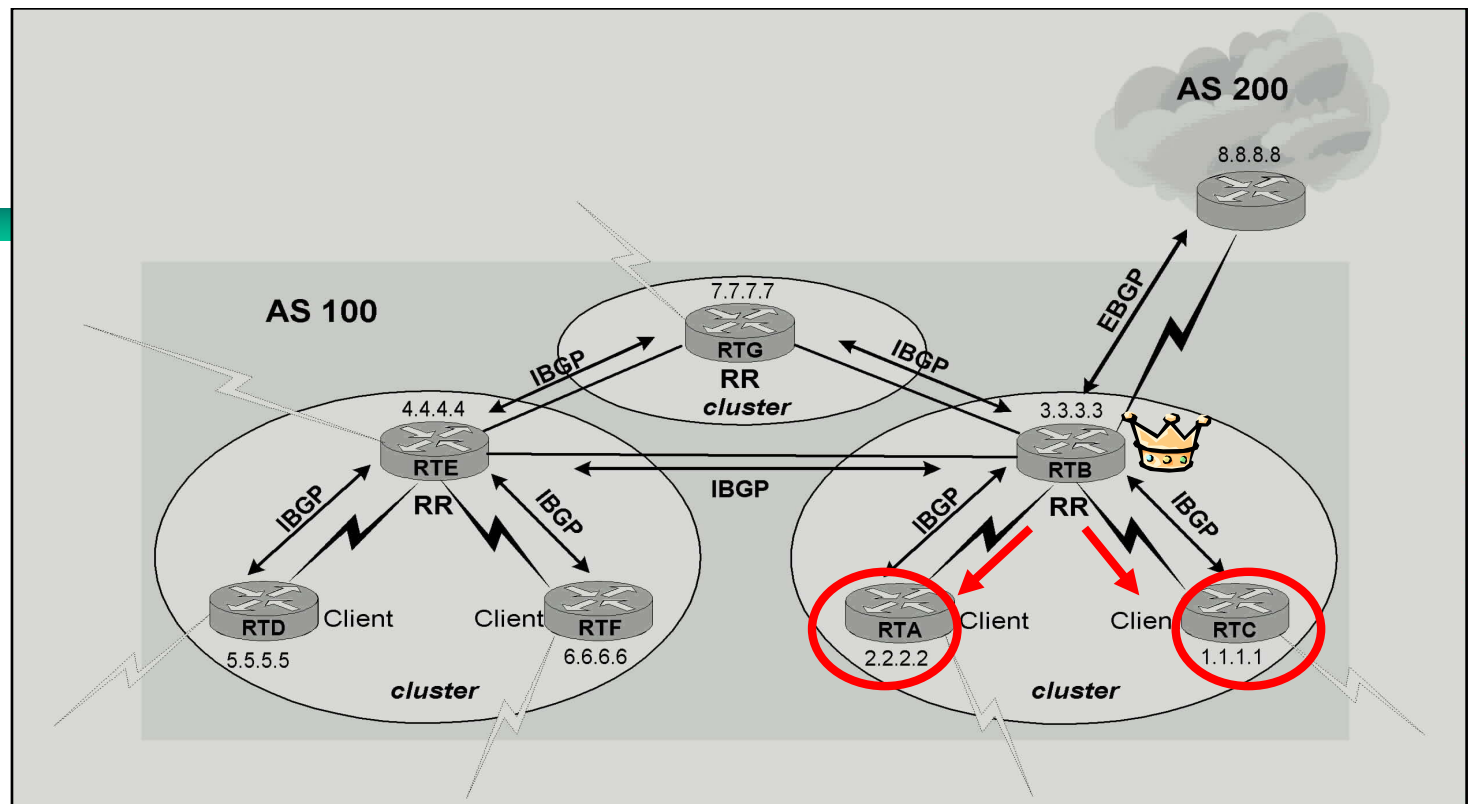
```
RTB (config-router) #neighbor 2.2.2.2 route-reflector-client
```

```
RTB (config-router) #neighbor 4.4.4.4 remote-as 100
```

```
RTB (config-router) #neighbor 7.7.7.7 remote-as 100
```

```
RTB (config-router) #neighbor 8.8.8.8 remote-as 200
```





Configuring a RR client:

+ Doesn't even know!

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

```
RTA(config-router)#neighbor 3.3.3.3 remote-as 100
```


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 **route map** command.

BGP Route Filtering

- + Only two steps are required to configure BGP basic route filtering with route maps:
 - Create a route map using the route-map command, and from route map configuration mode, use match commands to specify the attributes that are to be matched.
 - Apply the route map to a neighbor using the following command:
 - neighbor ip-address route-map route-map-name {in | out}

BGP Route Filtering

- + The following example shows how you can use a simple route map to limit route advertisements to locally generated routes.

```
router bgp 2001
network 62.128.60.0 mask 255.255.254.0
network 62.128.64.0 mask 255.255.254.0
network 62.128.68.0 mask 255.255.254.0
network 62.128.72.0 mask 255.255.254.0
network 62.128.76.0 mask 255.255.254.0
```

```
neighbor all-peers peer-group
neighbor all-peers route-map route-filter out
neighbor 62.128.47.6 remote-as 11151
neighbor 62.128.47.6 peer-group all-peers
neighbor 62.128.47.194 remote-as 645
neighbor 62.128.47.194 peer-group all-peers
neighbor 62.128.47.198 remote-as 645
neighbor 62.128.47.198 peer-group all-peers
no auto-summary
!
route-map route-filter permit 10
match route-type local
```

BGP Route Filtering – Prefix lists

- + 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

- + 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**, (i.e. 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
```

-
- + 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
```

- + 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
```


The COMMUNITIES attribute

- + A BGP community is a **group of destinations that share some common property**.
- + A community is not restricted to one network or one AS.
- + Communities are used to simplify routing policies by identifying routes based on a logical property rather than an IP prefix or an AS number.
- + A BGP speaker can use this attribute in conjunction with other attributes to control which routes to accept, prefer, and pass on to other BGP neighbors.
- + A route map is configured to manipulate community values.

The COMMUNITIES attribute

+ NO_EXPORT

- A route carrying this community value should not be advertised to peers outside a confederation (or the AS if it is the only AS in the confederation).

+ NO_ADVERTISE

- A route carrying this community value, when received, should not be advertised to any BGP peer

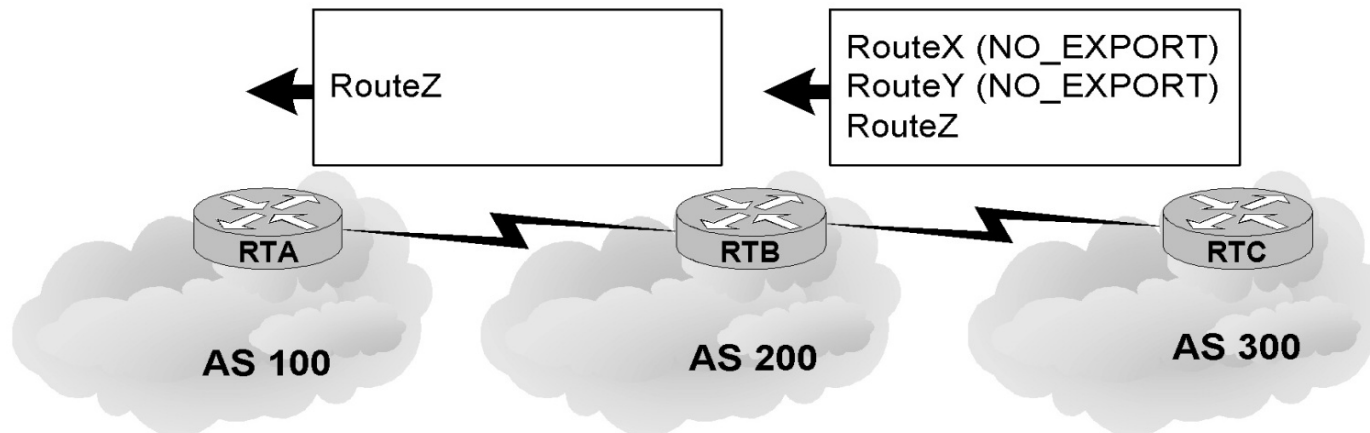
+ Internet

- A route carrying this community value, when received, should be advertised to all other routers.

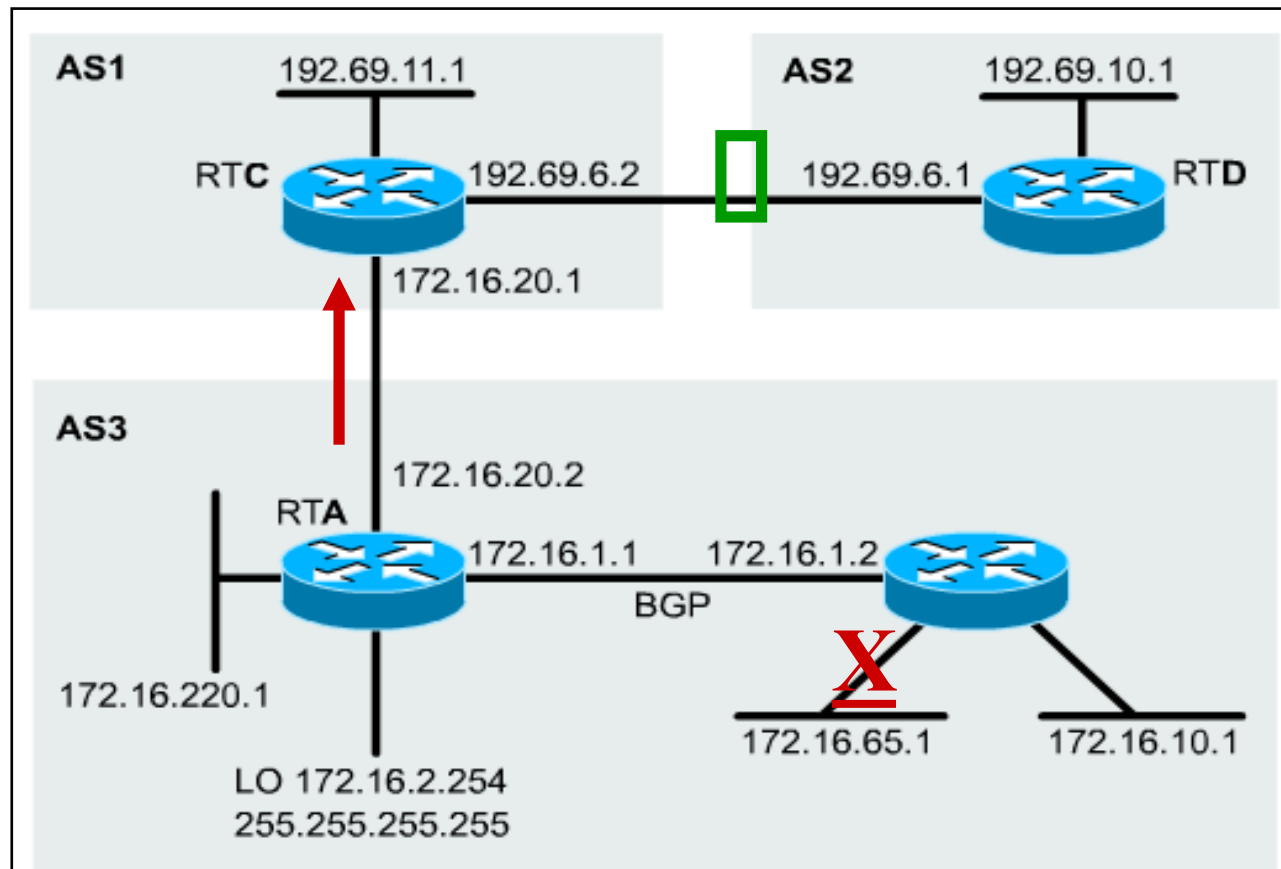
+ Local-as

- A route carrying this community value, when received, should be advertised to peers within the AS, but not advertised to peers in an external system.

The COMMUNITIES attribute



The COMMUNITIES attribute



To prevent **AS2** from learning the 172.16.65.0/24 route from AS1, we can configure RTA (AS3) as follows:

To prevent **AS2** from learning the 172.16.65.0/24 route from AS1, we can configure RTA (AS3) as follows:

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

```
RTA(config-router)#no auto-summary
```

```
RTA(config-router)#network 172.16.1.0 mask 255.255.255.0
```

```
RTA(config-router)#network 172.16.10.0 mask 255.255.255.0
```

```
RTA(config-router)#network 172.16.65.0 mask 255.255.255.192
```

```
RTA(config-router)#network 172.16.220.0 mask 255.255.255.0
```

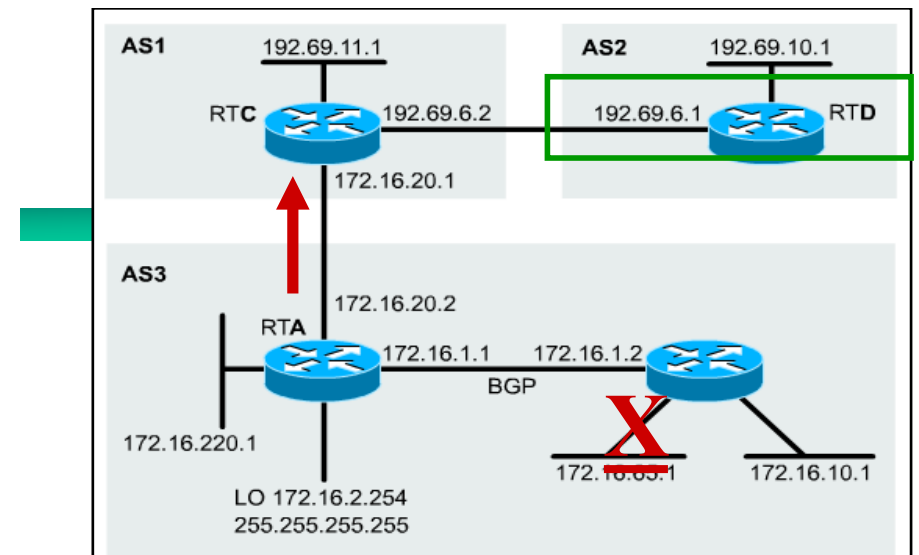
```
RTA(config-router)#neighbor 172.16.1.2 remote-as 3
```

```
RTA(config-router)#neighbor 172.16.1.2 update-source lo0
```

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

```
RTA(config-router)#neighbor 172.16.20.1 send-community
```

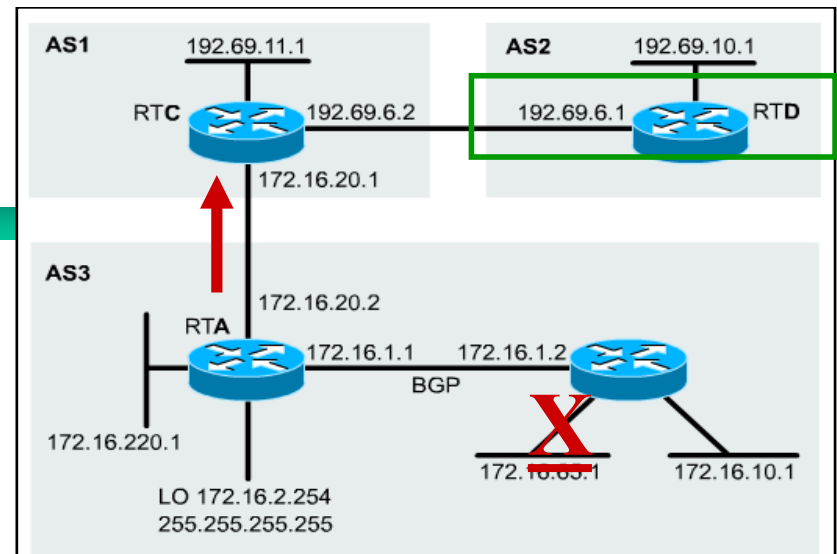
```
And: RTA(config-router)#neighbor 172.16.20.1 route-map SETCOMMUNITY out
```



```

RTA(config)#router bgp 3
RTA(config-router)#neighbor 172.16.20.1 send-
community
RTA(config-router)#neighbor 172.16.20.1 route-map
SETCOMMUNITY out
RTA(config-router)#exit
RTA(config)#route-map SETCOMMUNITY permit 10
RTA(config-route-map)#match ip address 1
RTA(config-route-map)#set community no-export
RTA(config)#route-map SETCOMMUNITY permit 20
RTA(config-route-map)#exit
RTA(config)#access-list 1 permit 172.16.65.0
0.0.0.255

```



- + RTA has defined a route map SETCOMMUNITY, and will send that value toward neighbor 172.16.20.1 (RTC).
- + Clause 10 of the route map will match on prefix 172.16.65.0/24 and will set its COMMUNITIES attribute to NO_EXPORT.
- + Clause 20 of the route map will enable all other networks to be passed with no change.
- + Notice that RTA is configured with the **send-community** option in the **neighbor** statement.
- + This option is necessary to instruct RTA to send the assigned community value out to that neighbor.