Border Gateway Protocol

ROUTE Module 6

Agenda

- Introduction and AS Routing
- Ground Principles and Packet Exchange
- Basic Configuration and Resetting BGP Neighborship
- BGP Attributes
- Additional Features

Autonomous System ①

- **Autonomous system (AS)** is group of networks and routers with the same routing policy under one administrative domain
  - Routing policy = path determination process for different networks
  - Administrative domain = network administrator scope of influence
- Usually one or more IGPs run inside AS but as whole it is under management of one organization
- AS is considered as the one undivided entity from outside point of view
  - From outside view all networks inside AS are considered reachable inside AS

Autonomous System ②

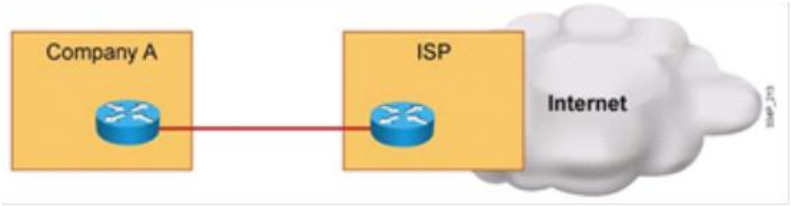
- AS has number – **Autonomous System Number**
  - IANA (ICANN) distributes ASN between regional internet registries and RIRs continue in distribution to its customers
  - 5 RIRs cover all continents (ARIN, RIPE NCC, APNIC, LACNIC, AfriNIC)
  - ASN is 2B long (from 0 to 65535)
  - RFC 4893 specifies ASN 4B long (written as 2B.2B)
  - Numbers between 64512 and 65535 are used for private AS usage (analogy to private IP addresses)
- ICANN insists that organization with only one ISP SHOULD use ISPs routing policy and more importantly ASN from private range
  - Private ASNs are used only inside ISP network
  - Private ASN is replaced with ISPs ASN when routing information are transferred to other AS

Autonomous System ③

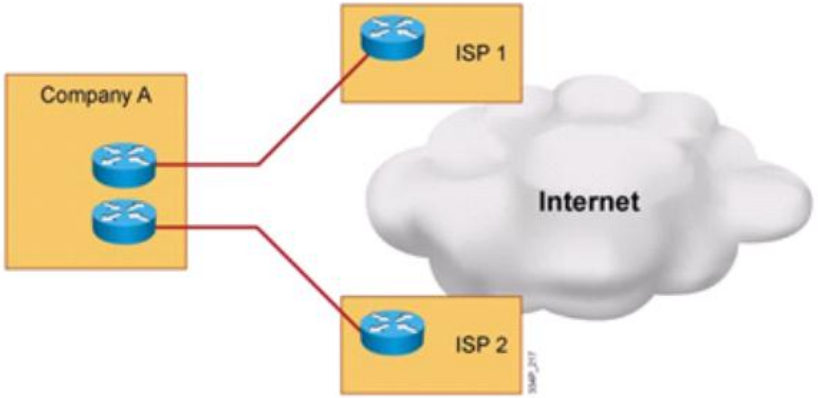
- *Historically and by usage AS are divided into 3 types*
- **Single-homed**
  - AS with only one border router to “outside” world
  - Single-homed AS does not usually require EGP routing
- **Multihomed**
  - AS with multiple border routers to “outside” world
  - Despite the fact it is connected with multiple egress points this AS type does not allow to transfer traffic intended to other AS
- **Transit**
  - AS type with multiple border routers which could transit traffic for different AS
  - *Usually transit AS interconnects ASs together*

Autonomous System ④

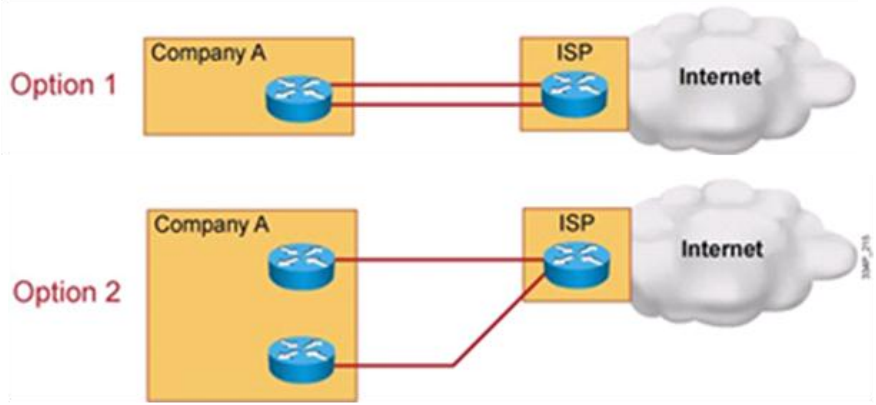
Single-homed



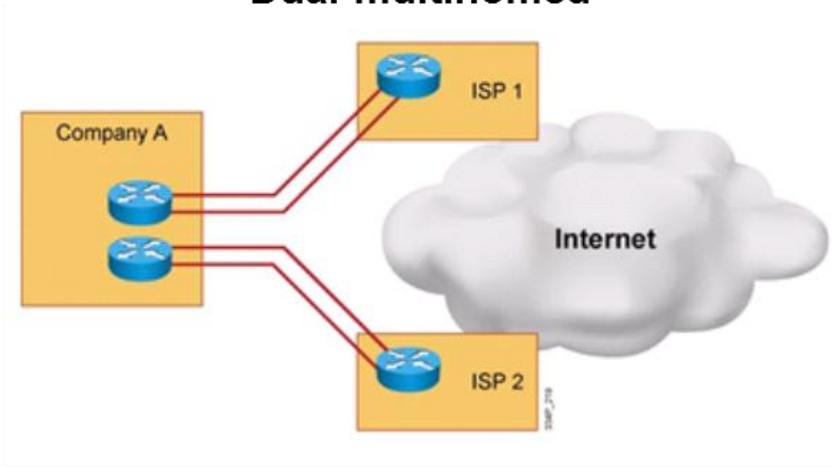
Multihomed



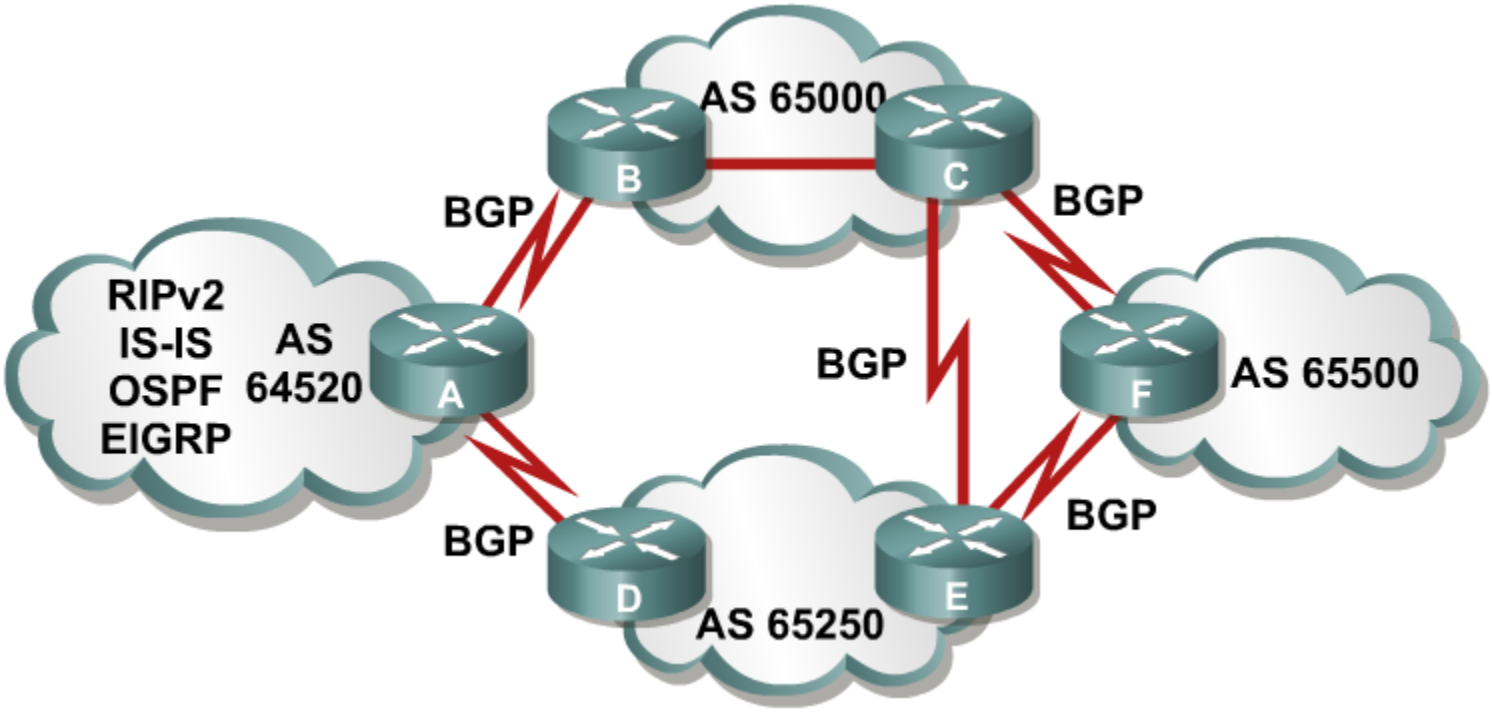
Dual-homed



Dual-multihomed



Routing between ASs ①



- Networks are “glued” with IGP inside AS
- Routing information are exchanged by BGP between ASs
  - *Which AS should be used next when routing to the destination?*
  - *Which path is shortest to get to neighbor AS?*

Routing between ASs ②

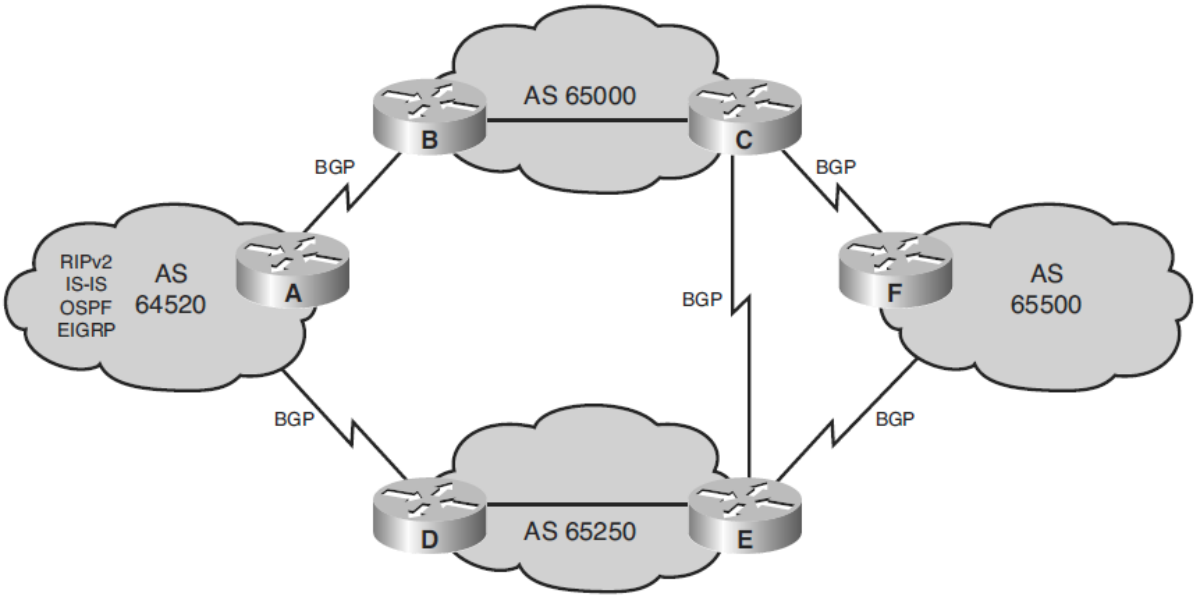
- *Routing between ASs majorly differs from routing inside AS*
- **IGP protocols**
  - Neighbor routers are discovered automatically
  - The main goal of IGP is to exchange as detailed routing information as possible about topology and reachable networks
  - World behind AS borders is “blurred” – viewed only through summary or default routes without any topological knowledge
  - Metric reflect “suitability” of routes based on number of hops, link speed, bandwidth, delay, load, etc.

Routing between ASs ③

- **EGP protocols**
  - Neighbor routers MUST be explicitly configured for mutual communication
  - EGP does not concern about intra-AS topology; dealing with intra-AS network reachability is in the hands of IGP
  - EGP are interested only in
    - border routers on the edges of AS
    - interconnection between AS
  - Metric consists of multiple attributes which reflects networks origin, path through all transit AS and administrative properties – it does not necessary reflects physical character of path

Routing between ASs ④

- Loop-less routing MUST be guaranteed
- The volume of routing information is huge – tens even hundreds of MB in routing tables
- Route determination is based on mutually agreed routing policies and administrative decisions between ASs



Basic Terminology and Principles

Border Gateway Protocol

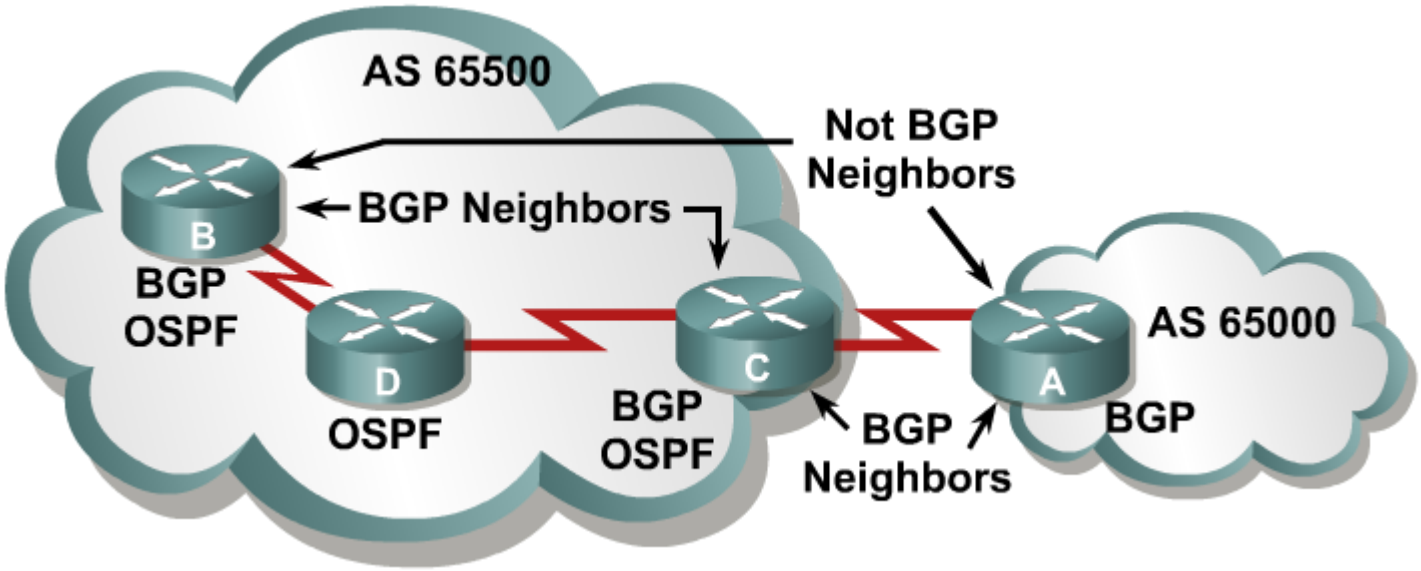
- BGP is currently the only one EGP for inter-AS routing
- Current version is BGPv4 specified in RFC 4271
  - Many related RFCs extend functionality of BGP to support multicast, MPLS, VPNs and others
- BGP runs above TCP on port 179
- EGP is not only the name of routing protocol family but also the name of BGPs old ancestor
  - Exterior Gateway Protocol described in RFC 904

BGP Tables

- **Neighbor Table**
  - The list and states of BGP neighbors
- **Forwarding Database**
  - The list of all routes to destination networks
  - For one target MAY exist multiple route records with all BGP attributes
- **Routing Table**
  - Only the best routes to destination networks leak from forwarding database as result of decision process

Speakers, Neighbors, Peers

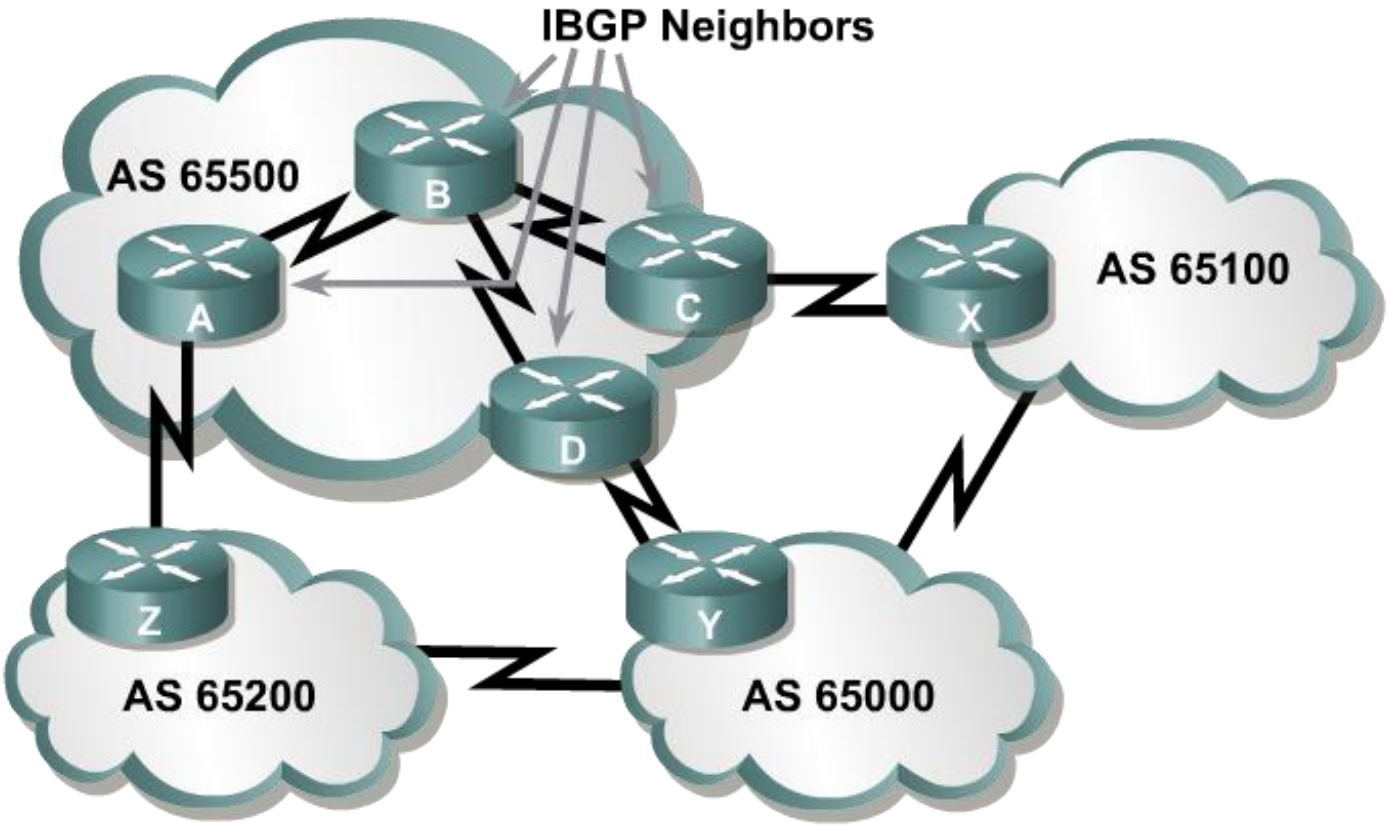
- **BGP speaker** is every router with enabled BGP routing process
- **BGP neighbors** (or also called **peers**) are pair of BGP routers configured to form BGP adjacency and exchange routing information



BGP Communication

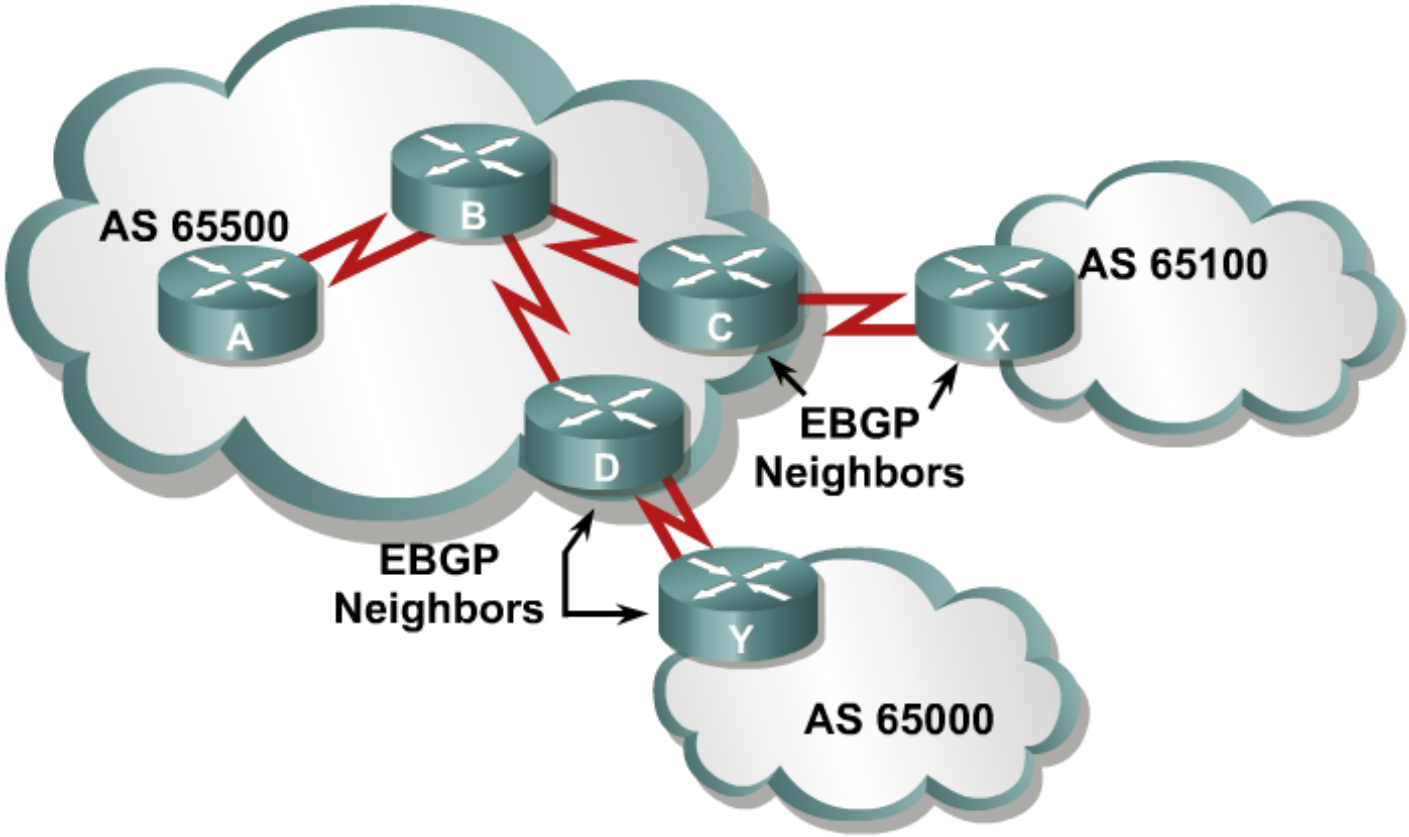
- Whenever BGP peers create neighborship they synchronize theirs routing information with each other
- After initial synchronization only incremental updates are sent – just changes in topology (insertion or withdrawal of route)
  - Incremental updates are more safer for bandwidth consumption than transfer of complete routing tables
  - *Seems like nothing but it's key issue of BGP – backbone BGP transfers could be in tenths or even hundreds of MB*

Internal BGP



- = BGP communication between neighbor routers in the same AS (a.k.a. **iBGP**)
- Neighbors does not have to be connected directly

External BGP



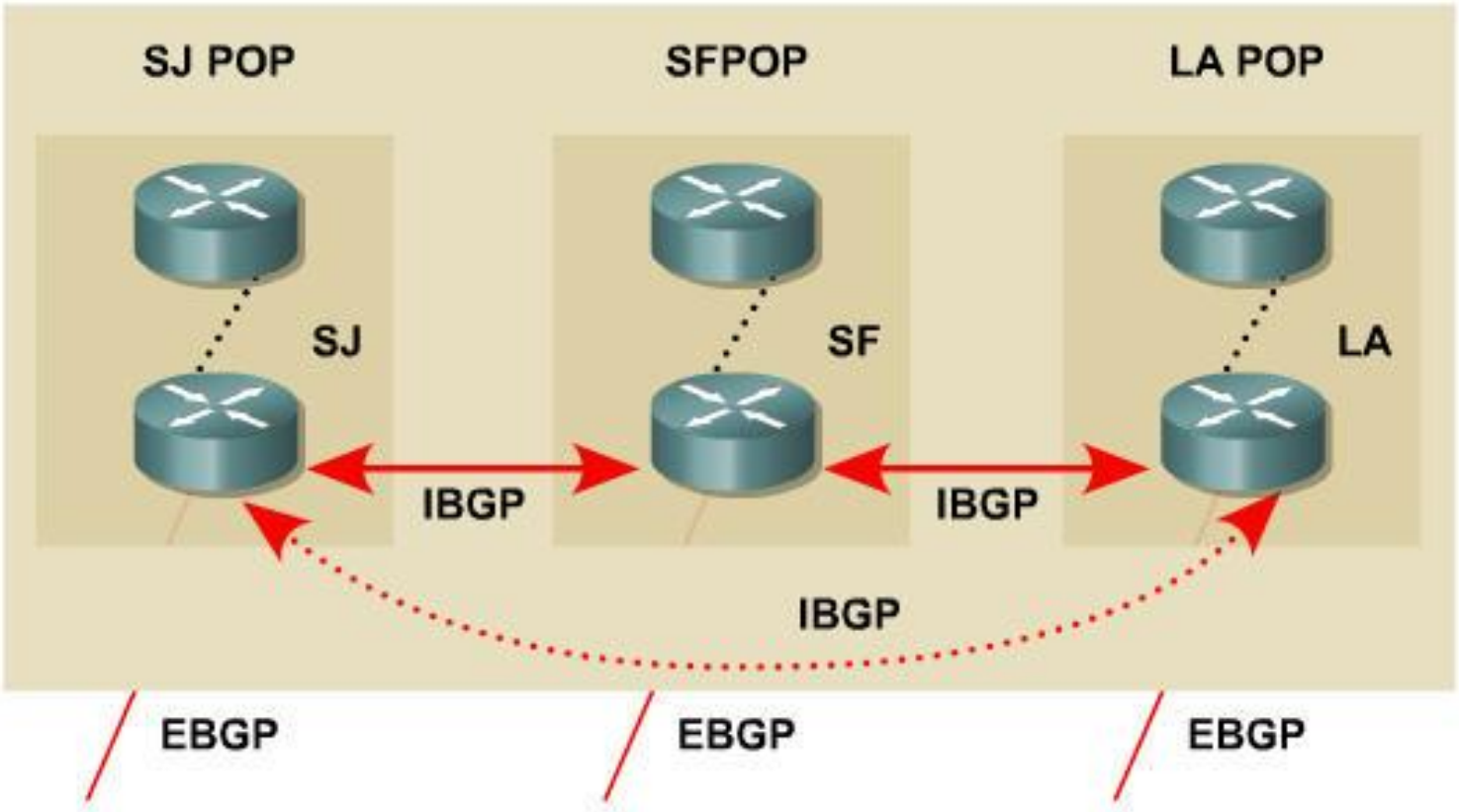
- = BGP communication between neighbor routers in different AS (a.k.a. **eBGP**)
- eBGP neighbors have to be directly connected (one hop away) under normal circumstances

iBGP vs. eBGP ①

- BGP behaves differently for iBGP or eBGP neighborship
- The main difference:
  - eBGP neighbors exchange routes connected and learned via BGP
  - iBGP neighbors exchange only connected networks
    - `IF` iBGP router learns about some route from its iBGP neighbor `THEN` it CAN NOT pass the information to another iBGP peer
    - *It's strict limitation, but also very efficient protection against routing loops*

iBGP vs. eBGP ②

- All BGP routers inside AS MUST have full-mesh BGP peering (but not necessarily in directly connected physical manner)



iBGP vs. eBGP ③

- Whenever there is large number of BGP speakers inside AS then their full-mesh peering is inefficient and bothers some for configuration
- This problem is in real world solved by:
  - Peer-groups
  - Route reflectors
  - By dividing of AS into subparts – internal ASs and binding them to so called confederations
- *Other differences exist between iBGP and eBGP (like TTL and source-address) but more about them on a right place*

Packet Exchange

Message Types ①

- **Open**
  - Contains version, ASN, holdtime, BGP router-id
  - This message is sent when BGP connection is opened between neighbors
- **Keepalive**
  - Periodically sent in every holdtime interval to neighbors to advertise that router is still “up and running”
- **Update**
  - Carries information about the one route (where term “route” is path through autonomous system to all possible destination networks accessible through this “route”)
  - Contains route’s attributes and Network Layer Reachability Information (NLRI – is list of all destination networks accessible through this route)
- **Notification**
  - This message is generated in case of problem and it contains description of error
  - BGP connection is immediately closed upon receivial of notification

Message Types ②

Open Message

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

Update Message

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

Notification Message

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

Keepalive Message

Octets	16	2	1
	Marker	Length	Type

States in BGP Communication

- **Idle**: Initial state. Neighbor is defined but router does not attempt to contact it
- **Connect**: Router successfully connected neighbor via TCP
- **Open sent**: Router sent to neighbor OPEN message with mutual parameters
- **Open confirm**: Router received neighbors OPEN message with agreement on mutual parameters. Neighbor is willing to communicate with router
  - `IF` message OPEN was sent AND open confirmation had not been received in 5 seconds `THEN` router transits to **Active** state
- **Established**: Routers successfully established neighborship and they could start exchange routing information

Troubleshooting of Active and Idle States

- **Active**: Router sent OPEN message `THEN` it is waiting to receive reply OR is waiting for TCP connection through neighbor
- **Idle**: `WHILE` router is staying in Idle state `THEN` for any reason TCP connection CAN NOT be established
  - Is there valid route in our routing table to target neighbor? This route CAN NOT be default route!
  - *Possible typo error in neighbor definition?*
- State could flap between Active and Idle states
- This situation points to problem between mutual neighbor communication. Possible causes:
  - Router or neighbor does not have backward route
  - Wrong addresses in BGP neighbor configuration
  - Neighbor does not have this router configured as neighbor
  - Mismatch in ASN
  - Firewall is blocking TCP communication on port 179

BGP Configuration

Basic Configuration

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

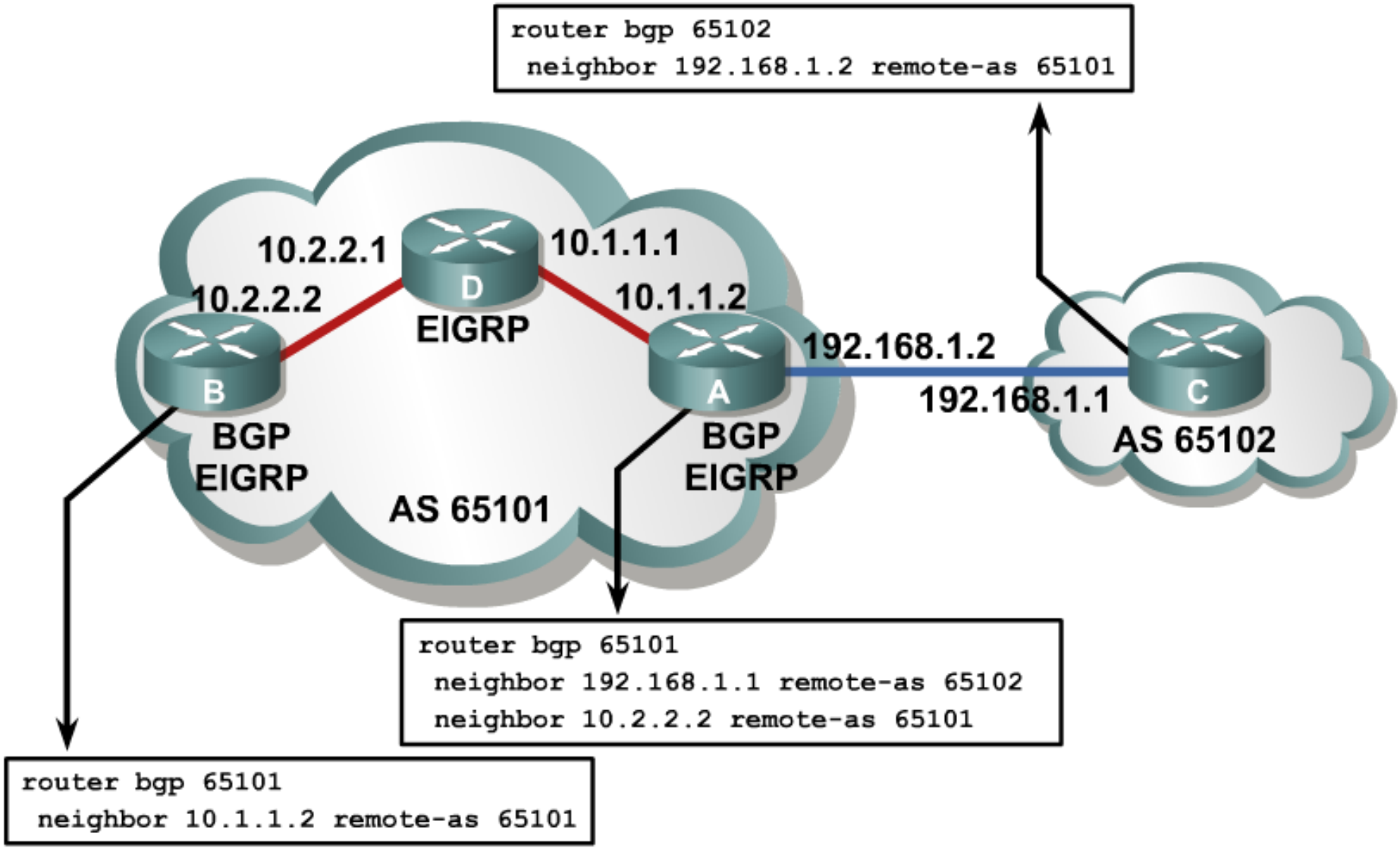
- Command starts BGP process on a router and sets its ASN
- Only the one instance of BGP could run on router (because router could not be in more AS)
- Configured ASN is compared with ASN of neighbors
  - *Why comparing?*
  - Way how to determine whether router is peering via iBGP or eBGP
- BGP router also has its own Router ID which is chosen in same manner as in EIGRP or OSPF

The `neighbor remote-as` Command ①

```
Router(config-router)#
neighbor {ip-address | peer-group-name} remote-as ASN
```

- Command **`neighbor`** defines neighbor that router wishes to peer with
- *`ip-address`* specifies destination address on which BGP packets will be sent for this neighbor
- For target address MUST exist record in routing table (and it CAN NOT be default route because it is never used for establishing communication)
- Argument **`remote-as`** tells to which AS neighbor belongs to
- With this one command both iBGP and eBGP peers are defined
- Between the pair of neighbors MUST match IP addresses in **`neighbor`** command
  - Source IP address for BGP packets on one neighbor MUST be destination IP address in **`neighbor`** command on neighboring router and vice versa

The `neighbor remote-as` Command ②



The `neighbor update-source` Command ①

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

- Source IP address of BGP packets are set according to interface IP address provided as *IFACE* parameter
- The most suitable are loopbacks, but their addresses MUST be properly advertise by IGP (or static routes)
- IP address in **neighbor** command of our neighbor must be set to IP address of update-source interface of ours router
- This command is usually used for iBGP peers – in case of eBGP peering SHOULD be neighbor directly connected (*which Loopback interface surely isn't*)

The `neighbor update-source` Command ②


```
router bgp 65101
 neighbor 172.16.1.1 remote-as 65100
 neighbor 3.3.3.3 remote-as 65101
 neighbor 3.3.3.3 update-source Loopback0
!
router eigrp 1
network 10.0.0.0
network 2.0.0.0
```

```
router bgp 65101
 neighbor 192.168.1.1 remote-as 65102
 neighbor 2.2.2.2 remote-as 65101
 neighbor 2.2.2.2 update-source Loopback0
!
router eigrp 1
network 10.0.0.0
network 3.0.0.0
```

The `neighbor ebgp-multihop` Command ①

- By default TTL in BGP packets is:
  - 255 in case of iBGP
  - 1 in case of eBGP
- *So is there a chance how to eBGP peer with loopbacks?*

```
Router(config-router)#
neighbor {ip-address | peer-group-name} ebgp-multihop [ttl]
```

- Previous command increases default TLL between eBGP peers
- Without *ttl* value 255 is used by default

The `neighbor ebgp-multihop` Command ②

The `neighbor shutdown` Command

```
Router(config-router)#
neighbor {ip-address | peer-group-name} shutdown
```

- Administratively deactivates neighbor
- This command is used when changing routing politics or maintenance of configuration

```
Router(config-router)#
no neighbor {ip-address | peer-group-name} shutdown
```

- Reactivates neighbor which was deactivated
  - Be aware of command **`neighbor activate`** which is doing something else (it activates neighbor for concrete network addresses)

The `network` Command

```
Router(config-router)#
network A.B.C.D [mask netmask]
```

- Network with exact NetID and subnet mask is put into BGP routing process and it is advertised to neighbors
- Command is doing something completely different than its equivalent for IGP:
  - Please note, that in case of IGP it searches through all interfaces – `IF` interface has IP address specified with network command `THEN` interface is added to routing protocol instance
  - In case of BGP it searches through routing table – `IF` exact match on target network is found in routing table `THEN` the network is advertised to all BGP neighbors of router

Summarization/Aggregation in BGP

- Term “aggregation” stands for same thing in BGP as term “summarization” in IGP
- Routes for aggregation are usually imported to BGP through redistribution

```
Router(config-router)#
aggregate-address NETWORK MASK [summary-only]
```

- With parameter **`summary-only`** only aggregated network is advertised – components are excluded
  - *Useful in cases when we want to send either only aggregated network or only components*

Default Route

- BGP can generate default route inside routing updates sent to target neighbor with command:

```
Router(config-router)#
neighbor {ip-address | peer-group-name} default-originate
```

- Default route NEED NOT exist in routing table of router – *Note that there are none default routes on Internet backbone*

Example of Properly Configured Peering

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

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

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

Display the BGP Topology Database

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

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

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

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

This route is offered to the IP routing table.

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

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

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

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

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

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

If blank the route is from the current autonomous system.

The last column displays the ORIGIN attribute).

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

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

Reset of BGP Connection

Restart of BGP Connection

- Changes in routing policy (ACL filtering, change in attributes) are applied only on the newly received or send prefixes
- Existing routes remains unaffected hence network administrator MUST manually take some action
- Ways how to enforce actualization of prefixes:
  - Hard reset
  - Soft reset
  - Route refresh

Hard Reset

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

- Cancels BGP connection with all neighbors
- All BGP table is dropped
- All connection transits to Idle state and all routing information must me resend to/from neighbors

```
Router#
clear ip bgp neighbor-address
```

- Cancels BGP connection with target neighbor
- Connection with this neighbor transits to Idle state and all information must me newly exchanged
- “Not so drastic” as **`clear ip bgp *`** but with same consequences

Outbound Soft Reset

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

- Routes from target neighbor remain in routers forwarding table but router itself resends all its routes to all/this neighbor(s)
- Connection remains in Established state
- This option is suitable when outbound policy is changed hence **`soft out`** is ineffective when changing inbound policy

Inbound Soft Reset

```
Router(config-router)#
neighbor [ip-address] soft-reconfiguration inbound
```

- Idea of Inbound Soft Reset is to remember all routing information from target neighbor untouched by routing policy in separate database
  - When inbound policy is changed content of this separate database is used instead of querying neighbor
- This additional configuration is needed for **`clear`** command to work properly and it is also memory intensive

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

- With this command all information from all (or just target neighbors) are forgotten and restored from separate database

Route Refresh

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

- Or so called Dynamical Inbound Soft Reset
- With additional BGP message called ROUTE-REFRESH specified in RFC 2918 router is able to ask neighbor to resend routing information to renew routes according to changed inbound policy
- Routes sent to neighbor remains unaffected
- No need for separate database
- Connection remains in Established state
- Supported from version IOS 12.0(2)S and 12.0(6)T
- Whenever all neighbors support Route Refresh feature there's no need to use **`soft`** attribute

Example: debug ip bgp updates

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

Attributes of BGP Routes

Attributes ①

- Attribute is property of route and this attribute usually influences BGP best route selection process
- There are planty of attributes in BGP but they're logically divided into 4 groups:
  - **Well-known mandatory**: support of this attribute IS required by any BGP implementation (well-known) and MUST be mandatory present in every route update (mandatory)
  - **Well-known discretionary**: support of this attribute IS required by any BGP implementation (well-known), but it NEED NOT be present in every route update (discretionary)
  - **Optional transitive**: support of this attribute IS NOT required by any BGP implementation (optional), but it MUST be send to neighbors – even in case that router doesn't recognize what to do with it (transitive)
  - **Optional nontransitive**: support of this attribute IS NOT required by any BGP implementation (optional), but `IF` router does not recognize it `THEN` router NEED NOT send it to neighbors (nontransitive)
- All "well-known" attributes are "transitive"

Attributes ②

- **Well-known mandatory** (*Help*: "**MONA**")
  - **O**RIGIN: origin of route (iBGP, eBGP, unknown)
  - **N**EXT_HOP: next hop for route
  - **A**S_PATH: list of AS through which route is reachible
- **Well-known discretionary** (*Help*: "**DALA**")
  - **A**TOMIC_**A**GGREGATE: info about undivided network
  - **L**OCAL_PREF: preferred "exit" from AS
- **Optional Transitive**
  - AGGREGATOR: source of aggregate information
- **Optional Nontransitive**
  - MULTI_EXIT_DISC: preferred "entrance" to AS

Attributes ③

- Except previous attributes Cisco invented proprietary attribute called weight which is locally significant to router and IS NOT sent to any router
  - *…more about it few slides ahead!*
- BGP attributes COULD be set or matched by route-maps
- Every match/set or filtering of BGP routes is done independently via **`neighbor`** command against target neighbor or peer/group
  - *Compare it with "global" filtering in case of any IGP*
- BGP has fixed order of evaluation of attributes to determine best route – strict ladder of attribute preferences

AS_PATH

- Attribute **AS_PATH** is the list of all ASN to traverse from router to destination network (including ASN of this router in the beginning and ASN of destination network in the end)
- `IF` route is passed in routing update between eBGP neighbors `THEN` sender adds its ASN in the beginning of this list – *prepending of ASN*
- AS_PATH is used to eliminate routing loops in BGP topologies
  - Route in the one ASN CAN NOT accept route with AS_PATH already containing its ASN somewhere inside

NEXT_HOP ①

- Attribute **NEXT_HOP** is IP address of next-hop router
  - BGP sees route as a chain of ASs not as a chain of routers…
  - …hence NEXT_HOP contains IP address of border router in next AS
- NEXT_HOP behavior complies with previous statement
  - `WHILE` eBGP peers are exchanging routing information `THEN` eBGP sender sets NEXT_HOP to its own IP address for target route (according to update-source)
  - `WHILE` iBGP peers are exchanging routing information `THEN` NEXT_HOP is leaved intact
- This has serious consequence!
  - Route to border router of different AS MUST be known inside AS
  - Otherwise it is not be possible to insert target route into routing table

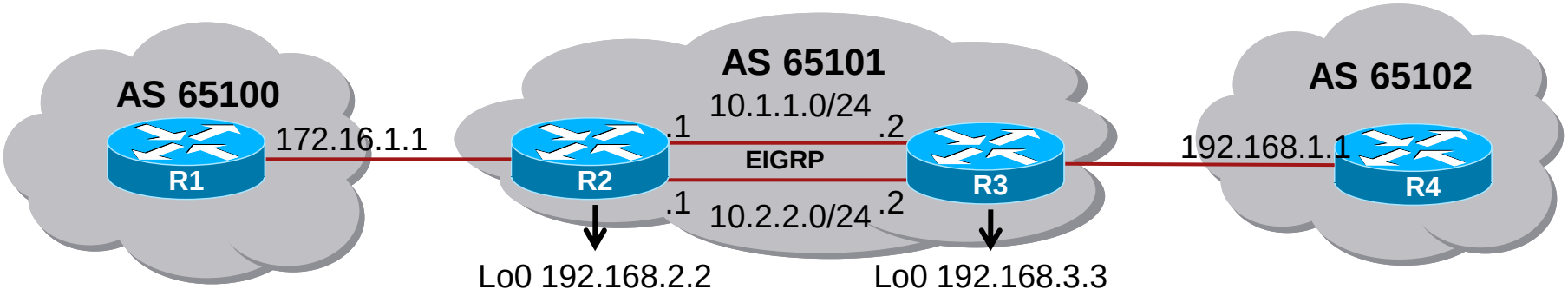
NEXT_HOP ②

- This behavior of NEXT_HOP could complicate situation in NBMA networks or in networks where is route to border router unknown to all BGP speakers
- Solution is command:

```
Router(config-router)#
neighbor A.B.C.D next-hop-self
```

  - In routes received from eBGP peer and exchanged with iBGP peer *A.B.C.D* is NEXT_HOP replaced with this router address
  - Command does not work on Route Reflector clients – in that case NEXT_HOP could be changed with route-map

NEXT_HOP ③




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

ORIGIN and WEIGHT

- Attribute **ORIGIN** express origin of route
  - „i“ – network has origin in current AS (it was inserted into BGP by `network` command)
  - „e“ – network is redistributed from former EGP protocol
  - „?“ – origin of network is unknown (usually it means it was imported into BGP by `redistribute` command)
- Attribute **WEIGHT** means „micro-local“ preferential of direction
  - It is Cisco proprietary locally significant attribute and it’s never send to neighbors
  - Higher value is more preferred – *Higher means better!*
  - Routes imported into BGP by this router has WEIGHT 32768
  - Routes imported into BGP by others has WEIGHT 0

LOCAL_PREF and MED

- Attribute **LOCAL_PREF** describes route preference
  - This attribute is exchanged only through iBGP neighborship because routers in different AS could preferre other routes
  - *Higher means better!*
  - By default routes have LOCAL_PREF set on 100
- **MED** indicates to other AS which route to use when entering our AS
  - MED is initially send from our AS via eBGP to neighbor AS. There MED is spread but does not cross borders of this AS (it traverses to other ASs with value set on 0)
  - MED is called also metric and it behaves just like metric of any IGP – *Lower means better*
  - Default MED is 0

Matching/Setting of Attributes ①

- Attribute manipulation is usually done by route-maps
- NEXT_HOP
  - **match ip next-hop {1-99 | 1300-1999 |** *NAME* **| prefix-list** *NAME*}
  - **set ip next-hop** *A.B.C.D*
- ORIGIN
  - **match route-type local**
  - **set origin {igp | egp | incomplete}**
- AS_PATH
  - **match as-path 1-500**
  - **set as-path prepend** *ASN ASN ASN...*

Matching/Setting of Attributes ②

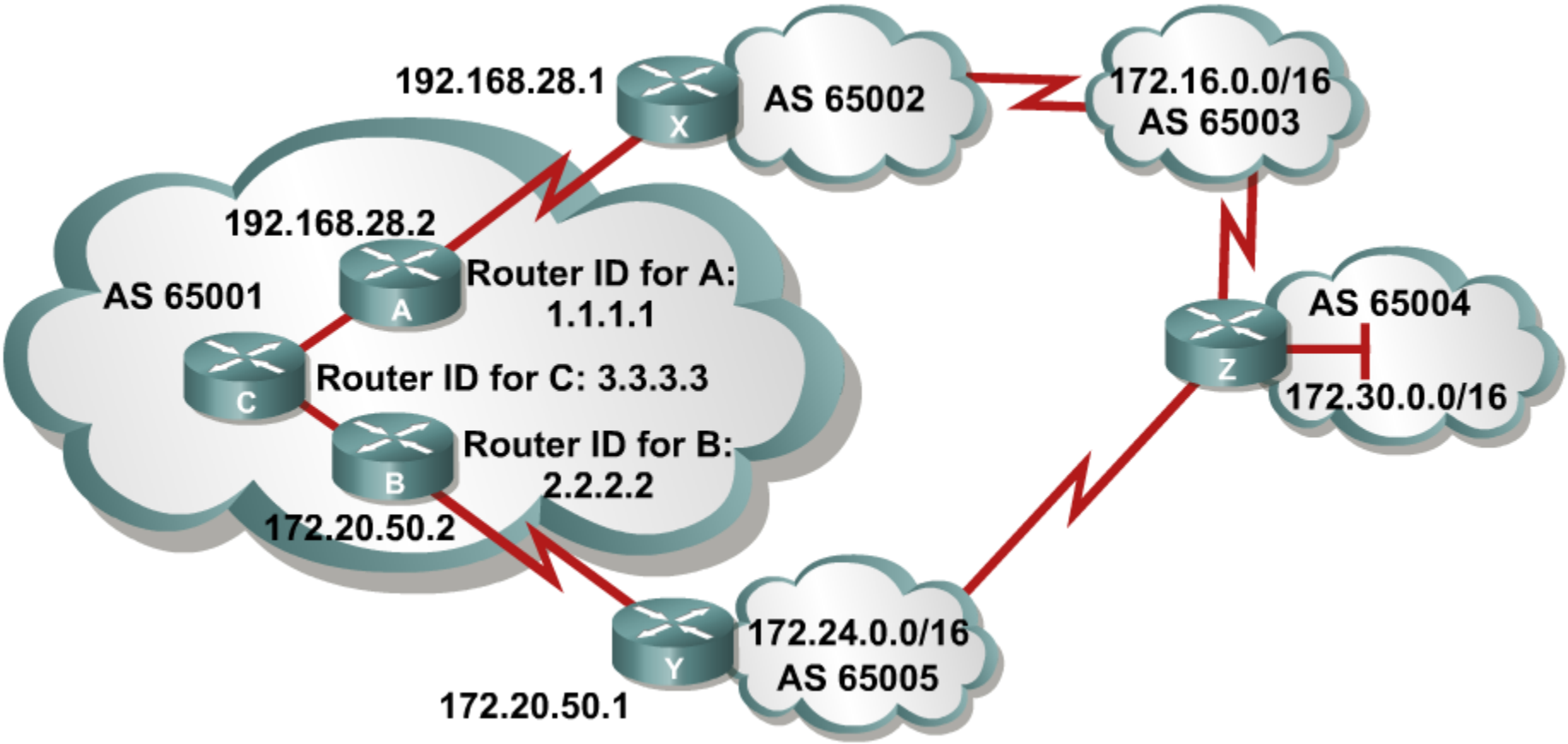
- LOCAL_PREF
  - **`match local-preference`** *`value`*
  - **`set local-preference`** *`value`*
- MULTI_EXIT_DISC
  - **`match metric`** *`value`*
  - **`set metric`** *`value`*
- WEIGHT
  - **`match`** doesn't exist (attribute is not sent)
  - **`set weight`** *`value`*

Change of Default LOCAL_PREF Value

```
Router(config-router)#
bgp default local-preference value
```

- Command changes LOCAL_PREF from standard value 100 to defined one
- All routes advertised to iBGP neighbors will have LOCAL_PREF set to this value

LOCAL_PREF Use-Case



LOCAL_PREF Use-Case: Route-Map on A

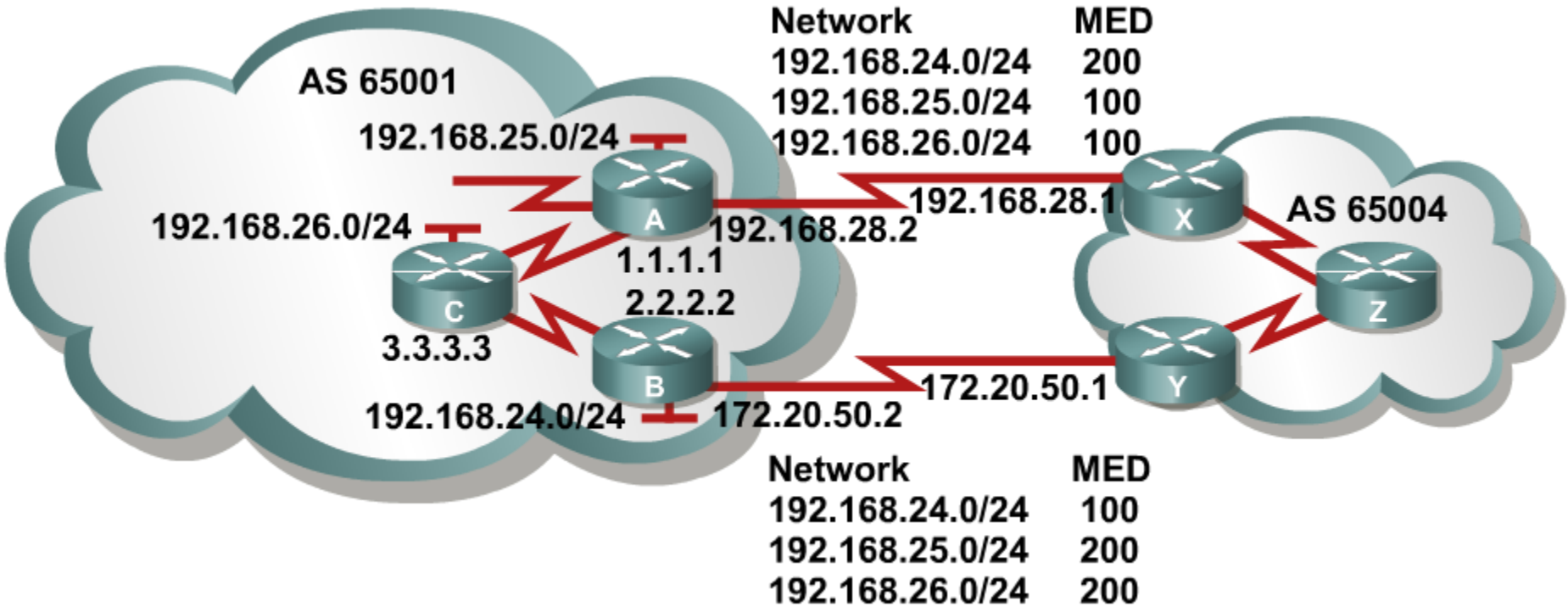
```
router bgp 65001
neighbor 2.2.2.2 remote-as 65001
neighbor 3.3.3.3 remote-as 65001
neighbor 2.2.2.2 remote-as 65001 update-source loopback0
neighbor 3.3.3.3 remote-as 65001 update-source loopback0
neighbor 192.168.28.1 remote-as 65002
neighbor 192.168.28.1 route-map local_pref in
!
access-list 65 permit 172.30.0.0 0.0.255.255
!
route-map local_pref permit 10
match ip address 65
set local-preference 400
!
route-map local_pref permit 20
```

Change of Default MED Value

```
Router(config-router)#
default-metric number
```

- Command changes MED from standard value 0 to the defined one
- All routes advertised to eBGP neighbors will have MED set to this value

MED Use-Case


Network MED
192.168.24.0/24 200
192.168.25.0/24 100
192.168.26.0/24 100
AS 65001
192.168.25.0/24
192.168.26.0/24
A
C
B
192.168.28.2
1.1.1.1
2.2.2.2
3.3.3.3
192.168.24.0/24
172.20.50.2
192.168.28.1
172.20.50.1
X
Y
Z
AS 65004
Network MED
192.168.24.0/24 100
192.168.25.0/24 200
192.168.26.0/24 200

MED Use-Case: Route-map on A

```
router bgp 65001
neighbor 2.2.2.2 remote-as 65001
neighbor 3.3.3.3 remote-as 65001
neighbor 2.2.2.2 update-source loopback0
neighbor 3.3.3.3 update-source loopback0
neighbor 192.168.28.1 remote-as 65004
neighbor 192.168.28.1 route-map med_65004 out
!
access-list 66 permit 192.168.25.0.0 0.0.0.255
access-list 66 permit 192.168.26.0.0 0.0.0.255
!
route-map med_65004 permit 10
match ip address 66
set metric 100
!
route-map med_65004 permit 100
set metric 200
```

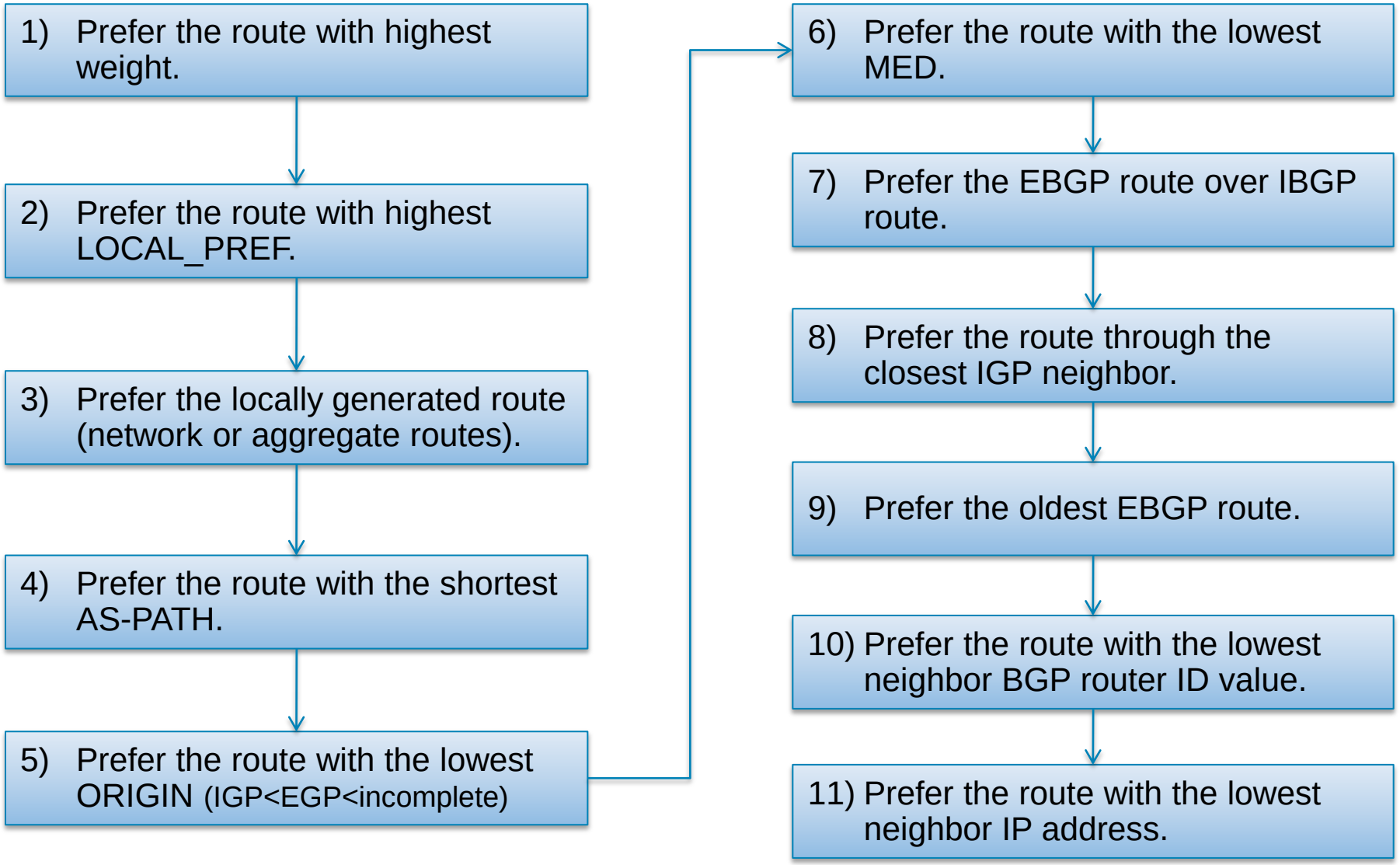
MED Use-Case: Route-map on B

```
router bgp 65001
neighbor 1.1.1.1 remote-as 65001
neighbor 3.3.3.3 remote-as 65001
neighbor 1.1.1.1 update-source loopback0
neighbor 3.3.3.3 update-source loopback0
neighbor 172.20.50.1 remote-as 65004
neighbor 172.20.50.1 route-map med_65004 out
!
access-list 66 permit 192.168.24.0.0 0.0.0.255
!
route-map med_65004 permit 10
match ip address 66
set metric 100
!
route-map med_65004 permit 100
set metric 200
```

MED Use-Case: What Z sees?

```
RouterZ# show ip bgp
BGP table version is 7, local router ID is 122.30.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal, r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*>i192.168.24.0     172.20.50.2              100    100        0 65001 i
* i                 192.168.28.2             200    100        0 65001 i
* i192.168.25.0     172.20.50.2              200    100        0 65001 i
*>i                 192.168.28.2             100    100        0 65001 i
* i192.168.26.0     172.20.50.2              200    100        0 65001 i
*>i                 192.168.28.2             100    100        0 65001 i
```

Best Route Selection Process



BGP Peer Groups, Router Reflectors & Autentification

Peer-Group Templates ①

```
Router(config-router)#
neighbor peer-group-name peer-group
```

- Command creates new peer-group with defined name

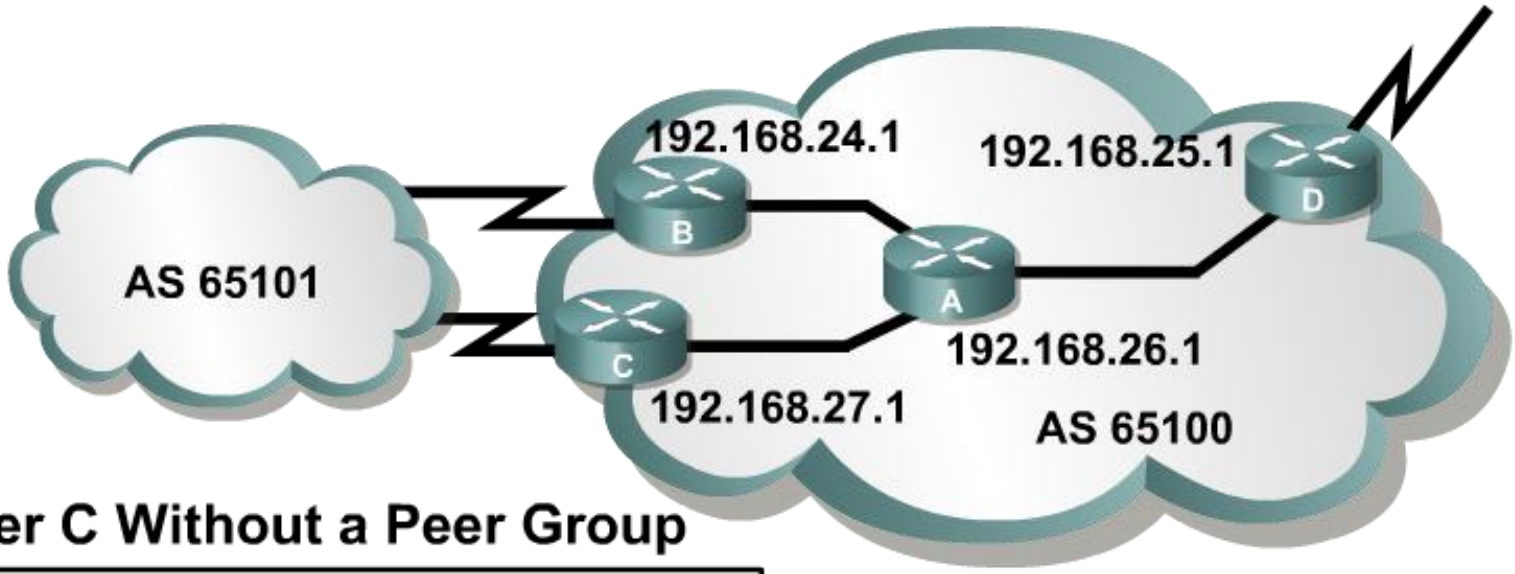
```
Router(config-router)#
neighbor ip-address peer-group peer-group-name
```

- Neighbor becomes part of peer-group
- When outbound policy is same for set neighbors it is suitable to use peer-group templates
- Members of peer-group share all peer-group settings but they could have different inbound policies

Peer-Group Templates ②

- Peer-groups have many benefits:
  - Route updates for all members of peer-group are generated only once (not for each member separately)
  - Simpler configuration – all filters, route-maps a similar constructions are applied on whole peer-group (not for each member separately)
  - CPU time and memory is spared, because routing table for peer-group is checked only once
- Peer-group simplificates settings but peers full-mesh connectivity requirement still exists

Peer-Group Use-Case



Router C Without a Peer Group

```
router bgp 65100
 neighbor 192.168.24.1 remote-as 65100
 neighbor 192.168.24.1 update-source Loopback 0
 neighbor 192.168.24.1 next-hop-self
 neighbor 192.168.24.1 distribute-list 20 out
 neighbor 192.168.25.1 remote-as 65100
 neighbor 192.168.25.1 update-source Loopback 0
 neighbor 192.168.25.1 next-hop-self
 neighbor 192.168.25.1 distribute-list 20 out
 neighbor 192.168.26.1 remote-as 65100
 neighbor 192.168.26.1 update-source Loopback 0
 neighbor 192.168.26.1 next-hop-self
 neighbor 192.168.26.1 distribute-list 20 out
```

Router C Using a Peer Group

```
router bgp 65100
 neighbor internal peer-group
 neighbor internal remote-as 65100
 neighbor internal update-source Loopback 0
 neighbor internal next-hop-self
 neighbor internal distribute-list 20 out
 neighbor 192.168.24.1 peer-group internal
 neighbor 192.168.25.1 peer-group internal
 neighbor 192.168.26.1 peer-group internal
```

Route Reflector

- **Route reflector (RR)** is BGP router which breaks rule about iBGP-iBGP forbidden exchange of via BGP learned routes
- Configuration inside AS could be simpler with help of RR – logical hub-and-spoke topology is created instead of full-mesh connectivity
- RR configures all spokes as its neighbors and adds following command:

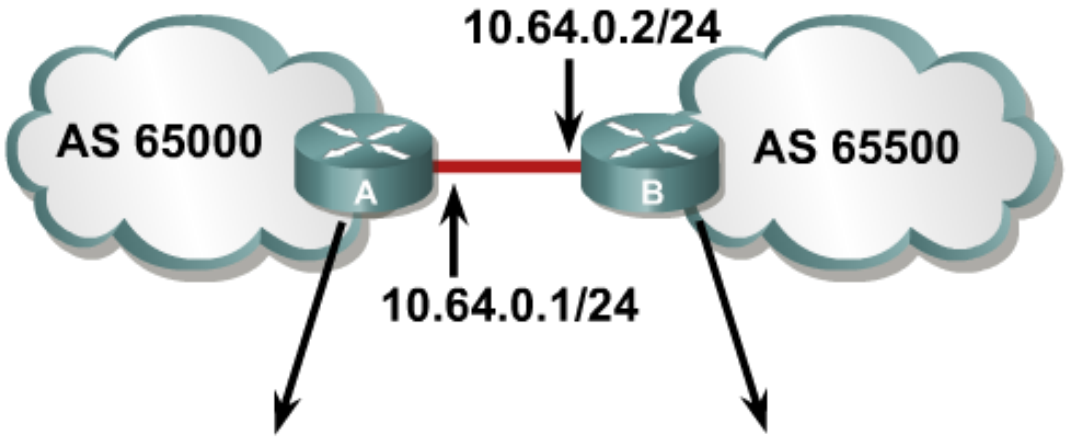
```
Router(config-router)#
neighbor {ip-address | pg-name} route-reflector-client
```

  - All other BGP routers configures RR neighbor in normal manner

Authentication

```
Router(config-router)#
neighbor {ip-address | peer-group-name} password string
```

- BGP uses MD5 authentication
- Independent key could be defined for each neighbor or peer-group




```
router bgp 65500
 neighbor 10.64.0.2 remote-as 65500
 neighbor 10.64.0.2 password v61ne0qkel33&
```

```
router bgp 65500
 neighbor 10.64.0.1 remote-as 65000
 neighbor 10.64.0.1 password v61ne0qkel33&
```

Useful Commands

- `show ip bgp`
- `show ip bgp neighbors`
- `show ip bgp summary`
- `show ip bgp rib-failure`
- `debug ip bgp [dampening | events | keepalives | updates]`

Where to Go Next?

- BGP Case Studies
  - http://www.cisco.com/en/US/customer/tech/tk365/technologies_tech_note09186a00800c95bb.shtml
- Using Regular Expressions
  - http://www.cisco.com/en/US/customer/tech/tk365/technologies_tech_note091
  - http://www.cisco.com/en/US/customer/products/hw/switches/ps718/products_command_reference_chapter09186a008009166c.html86a0080094a92.shtml

Slides adapted by Vladimír Veselý partially from official course materials but the most of the credit goes to CCIE#23527 Ing. Peter Palúch, Ph.D.

Last update: 2012-09-11