



BGP Attributes and Policy Control

ISP/IXP Workshops

Agenda

- **BGP Attributes**
- **BGP Path Selection**
- **Applying Policy**



BGP Attributes

The “tools” available for the job

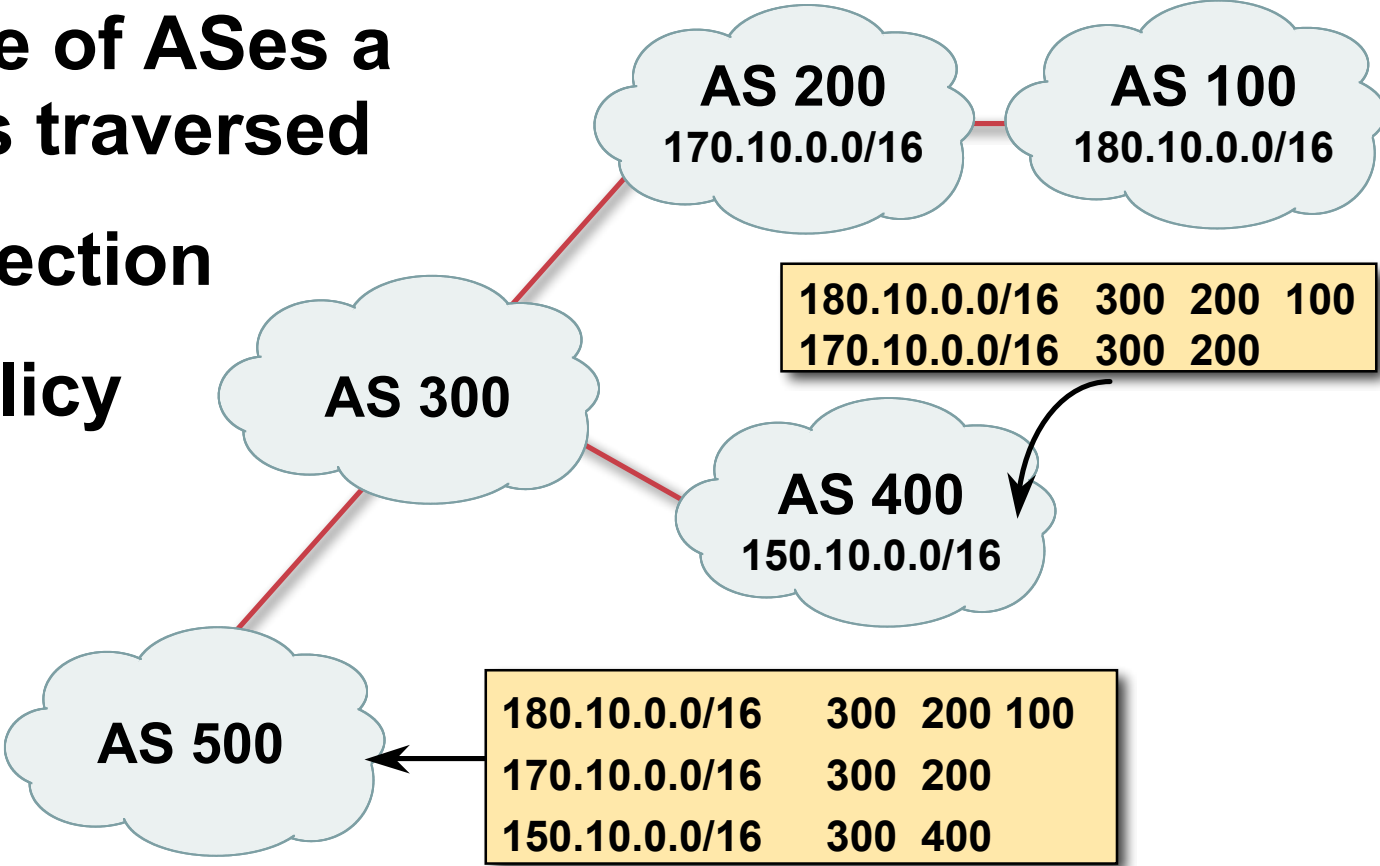
What Is an Attribute?



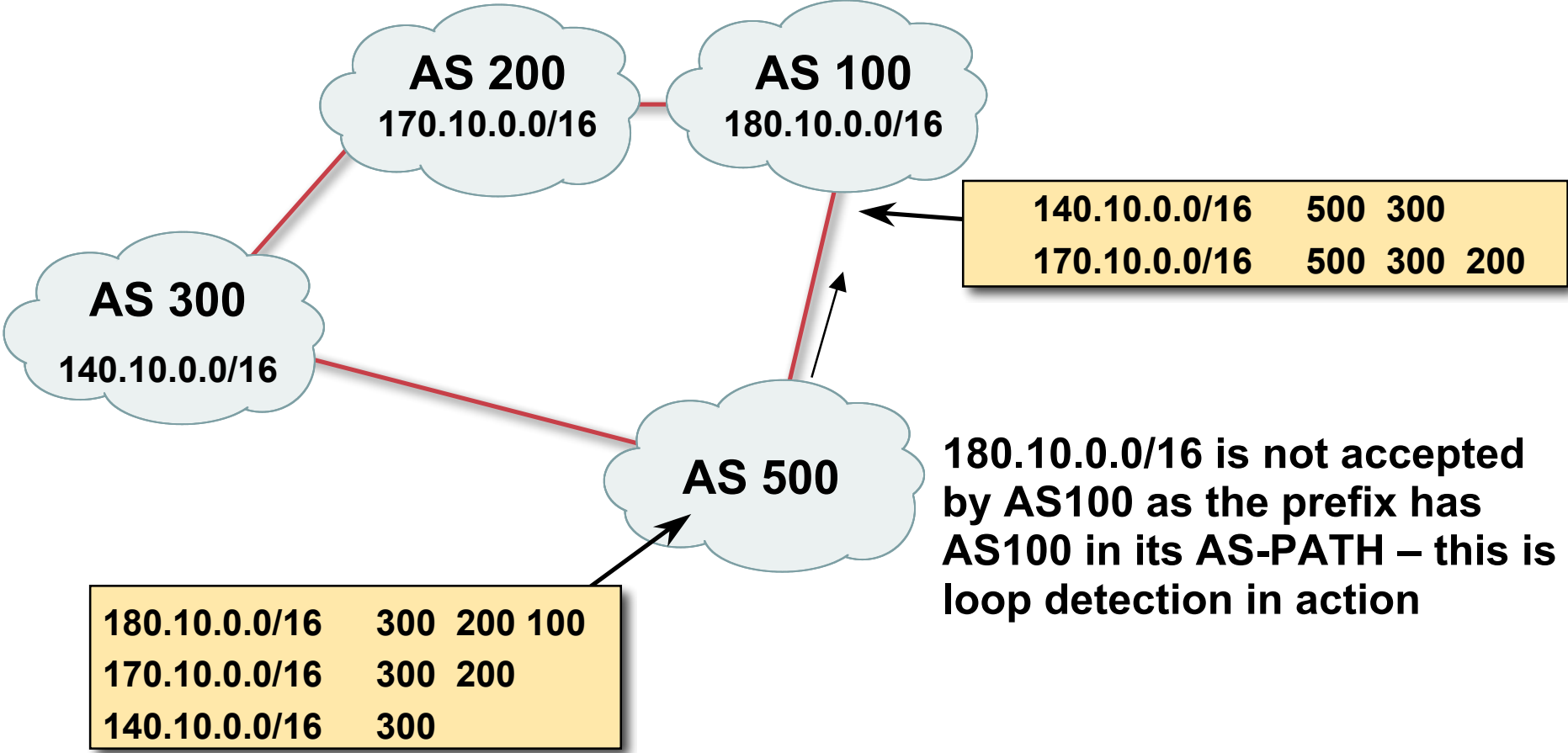
- **Describes the characteristics of prefix**
- **Transitive or non-transitive**
- **Some are mandatory**

AS-Path

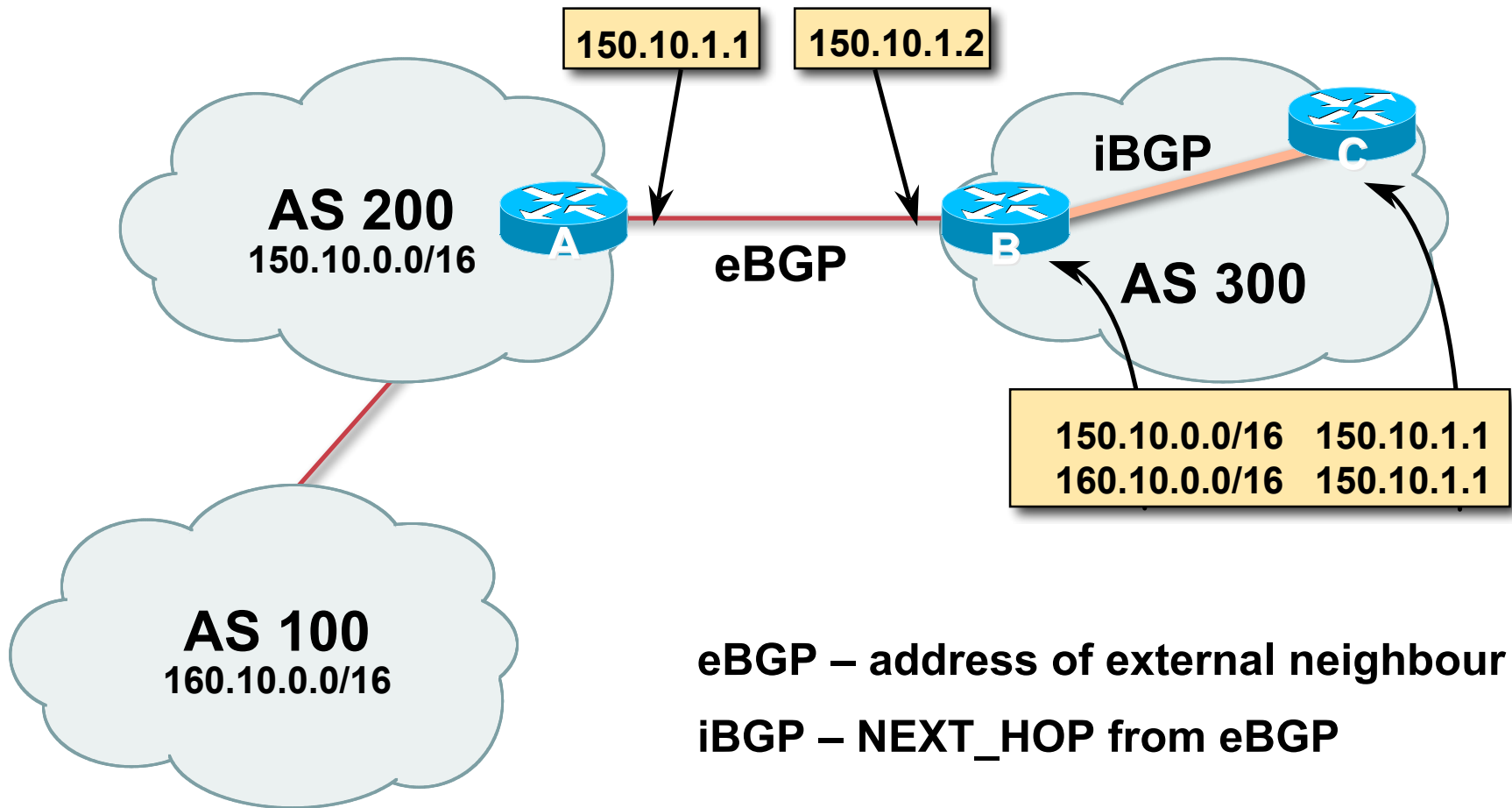
- Sequence of ASes a route has traversed
- Loop detection
- Apply policy



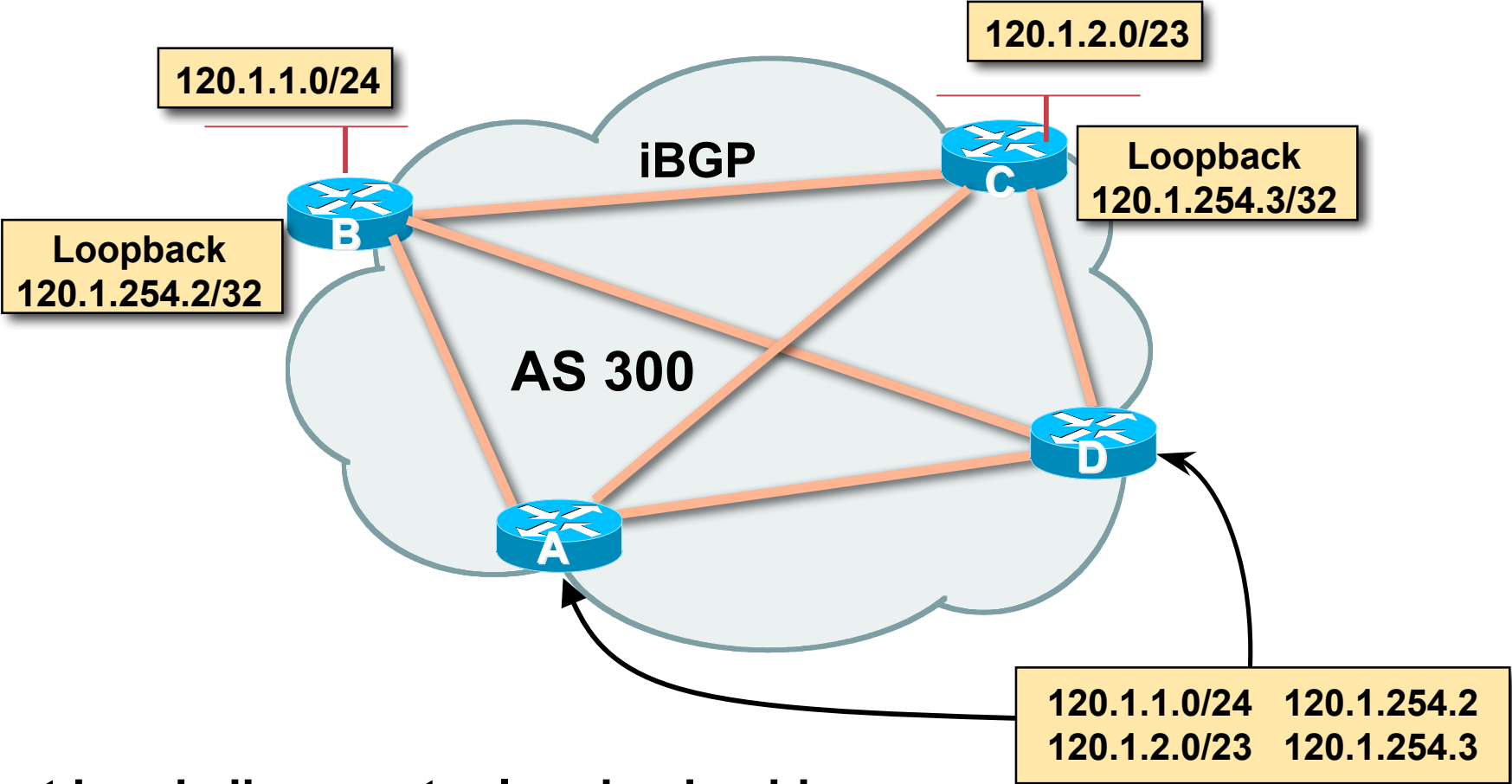
AS-Path loop detection



Next Hop



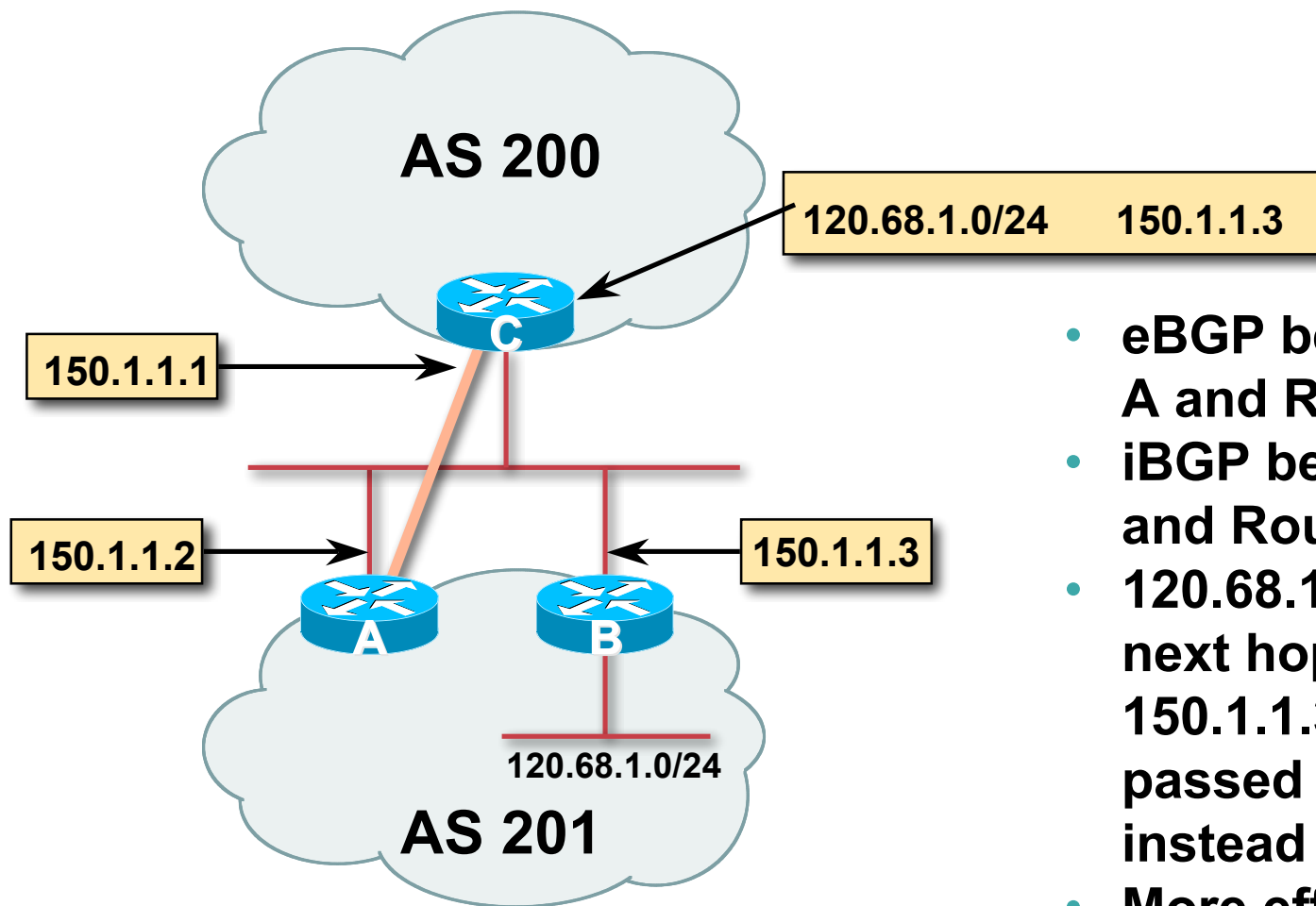
iBGP Next Hop



Next hop is ibgp router loopback address

Recursive route look-up

Third Party Next Hop



- eBGP between Router A and Router C
- iBGP between Router A and Router B
- 120.68.1/24 prefix has next hop address of 150.1.1.3 – this is passed on to Router C instead of 150.1.1.2
- More efficient
- No extra config needed

Next Hop (summary)

- **IGP should carry route to next hops**
- **Recursive route look-up**
- **Unlinks BGP from actual physical topology**
- **Allows IGP to make intelligent forwarding decision**

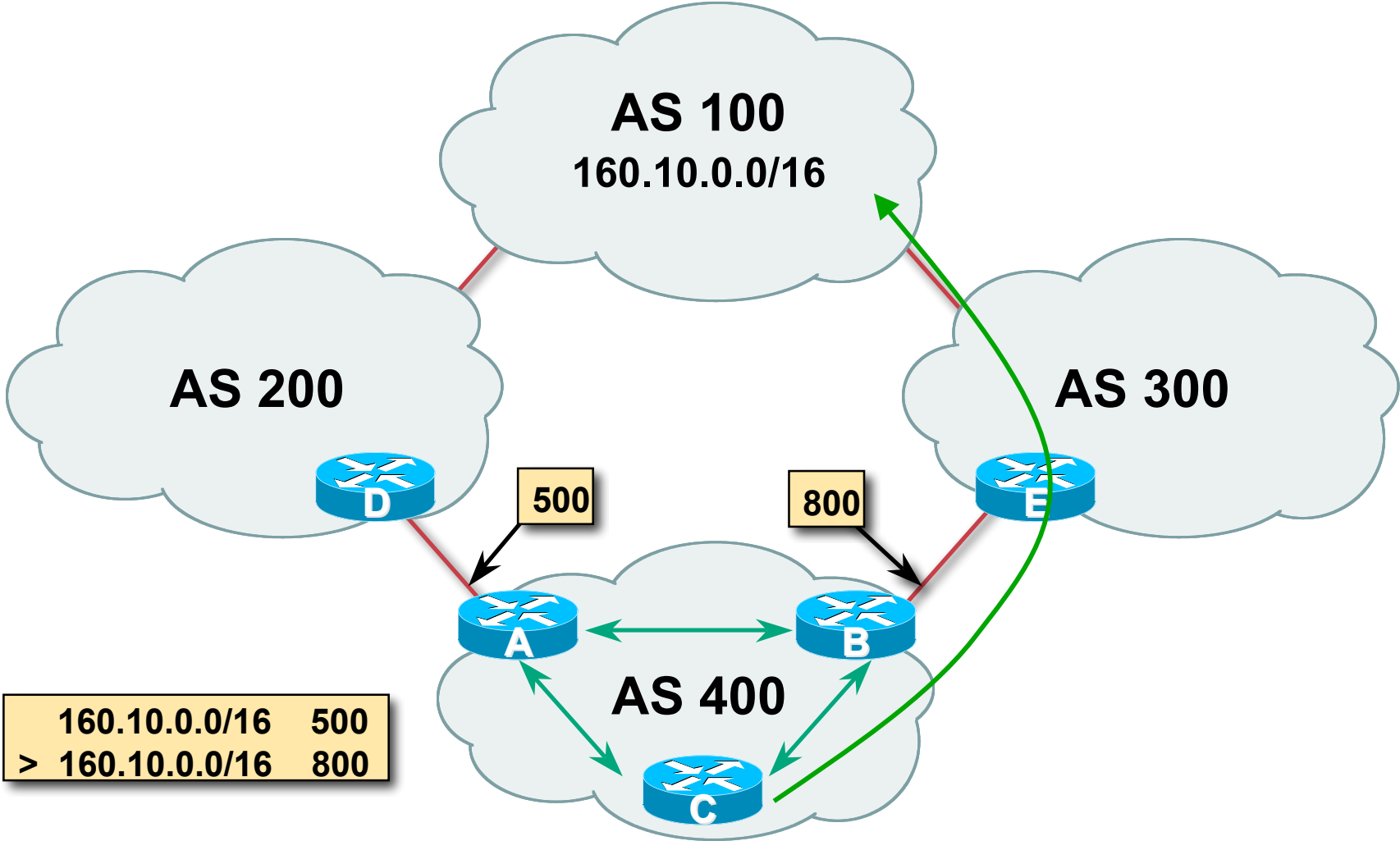
Origin

- **Conveys the origin of the prefix**
- **“Historical” attribute**
- **Influences best path selection**
- **Three values: IGP, EGP, incomplete**
 - IGP – generated by BGP network statement**
 - EGP – generated by EGP**
 - incomplete – redistributed from another routing protocol**

Aggregator

- **Useful for debugging purposes**
- **Conveys the IP address of the router/BGP speaker generating the aggregate route**
- **Does not influence path selection**

Local Preference



Local Preference

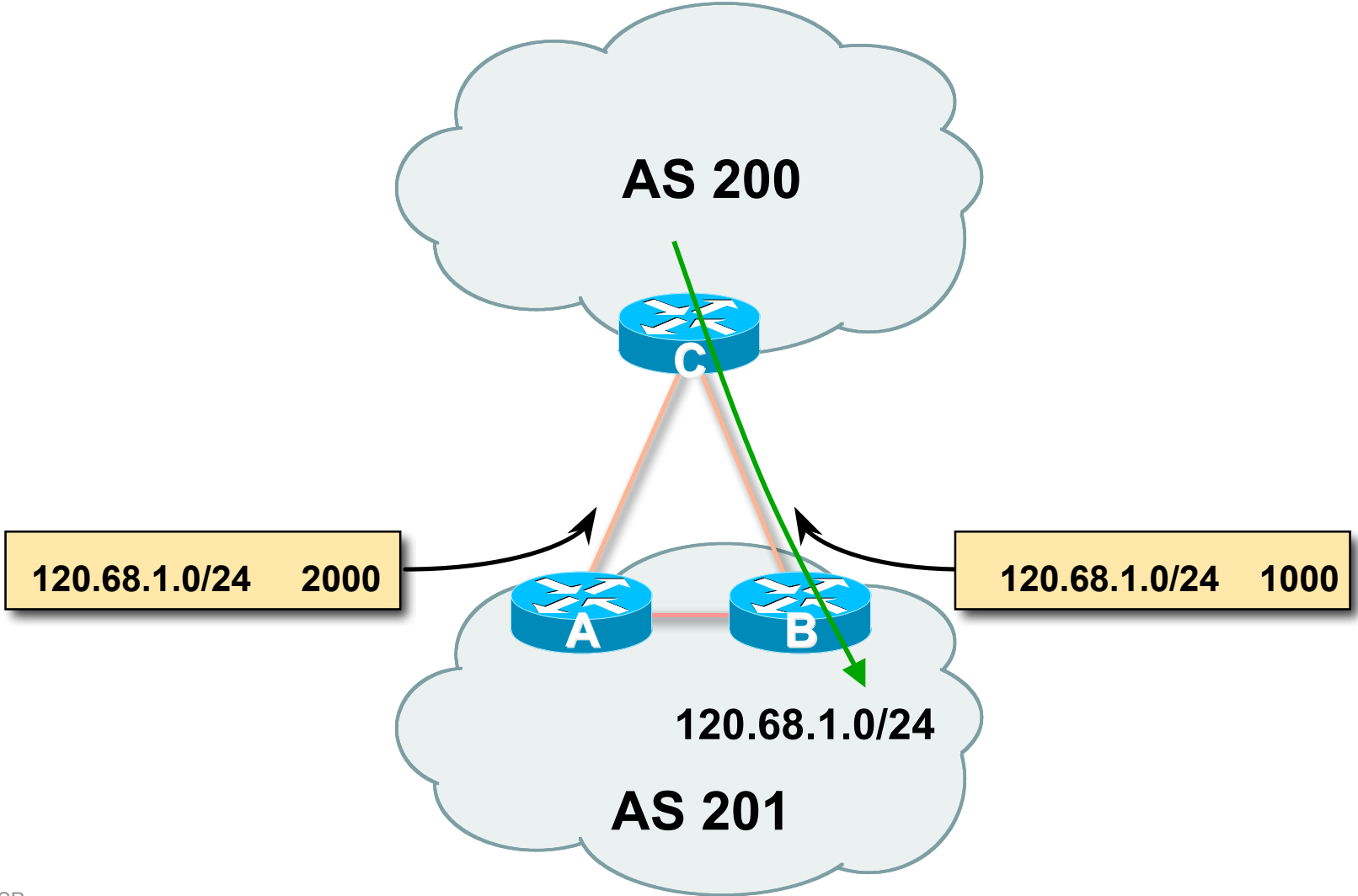
- **Local to an AS – non-transitive**
 - local preference set to 100 when heard from neighbouring AS
- **Used to influence BGP path selection**
 - determines best path for *outbound* traffic
- **Path with highest local preference wins**

Local Preference

- **Configuration of Router B:**

```
router bgp 400
  neighbor 120.5.1.1 remote-as 300
  neighbor 120.5.1.1 route-map local-pref in
!
route-map local-pref permit 10
  match ip address prefix-list MATCH
  set local-preference 800
!
ip prefix-list MATCH permit 160.10.0.0/16
```

Multi-Exit Discriminator (MED)



Multi-Exit Discriminator

- **Inter-AS – non-transitive & optional attribute**
- **Used to convey the relative preference of entry points**
 - determines best path for *inbound* traffic
- **Comparable if paths are from same AS**
 - bgp always-compared-med** allows comparisons of MEDs from different ASes
- **Path with lowest MED wins**
- **Absence of MED attribute implies MED value of zero (draft-ietf-idr-bgp4-26.txt)**

MED & IGP Metric

- **IGP metric can be conveyed as MED**

set metric-type internal in route-map

enables BGP to advertise a MED which corresponds to the IGP metric values

changes are monitored (and re-advertised if needed) every 600s

bgp dynamic-med-interval <secs>

Multi-Exit Discriminator

- **Configuration of Router B:**

```
router bgp 400
  neighbor 120.5.1.1 remote-as 200
  neighbor 120.5.1.1 route-map set-med out
!
route-map set-med permit 10
  match ip address prefix-list MATCH
  set metric 1000
!
ip prefix-list MATCH permit 120.68.1.0/24
```

Weight

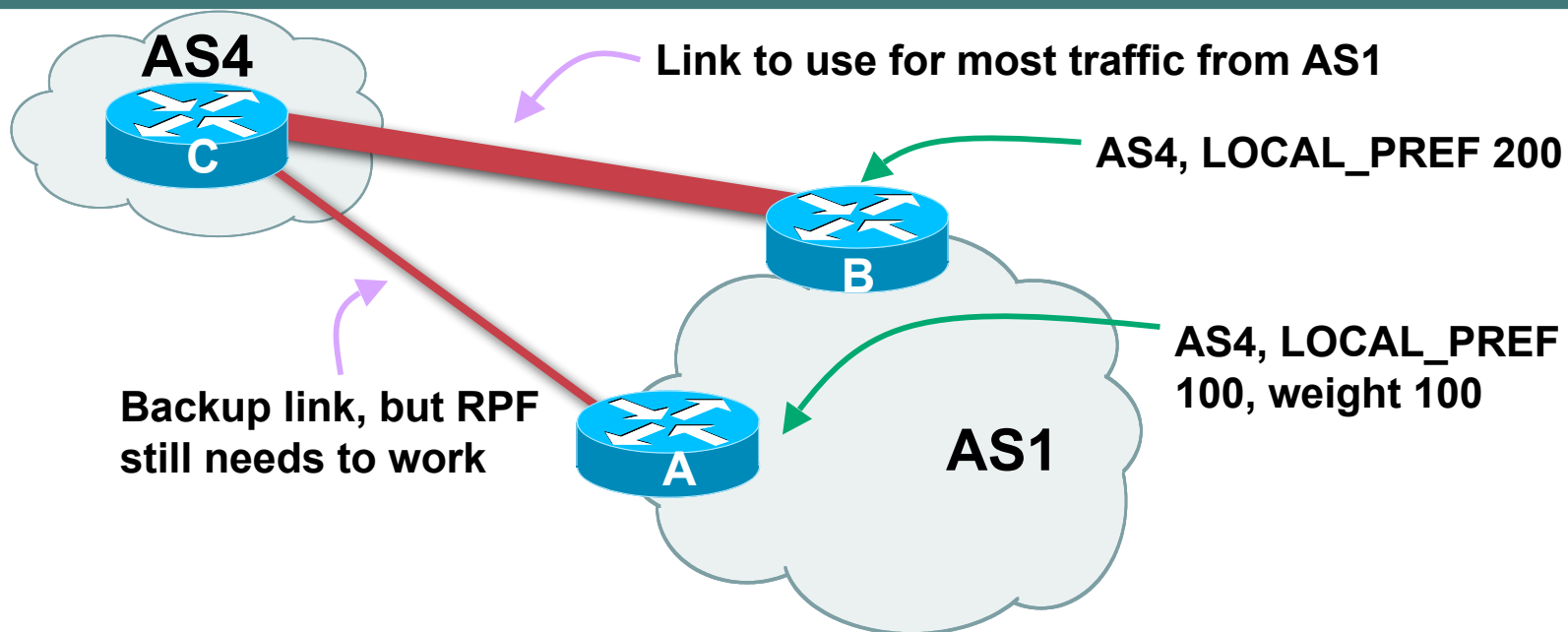
- **Not really an attribute – local to router**
- **Highest weight wins**
- **Applied to all routes from a neighbour**

```
neighbor 120.5.7.1 weight 100
```

- **Weight assigned to routes based on filter**

```
neighbor 120.5.7.3 filter-list 3 weight 50
```

Weight – Used to help Deploy RPF



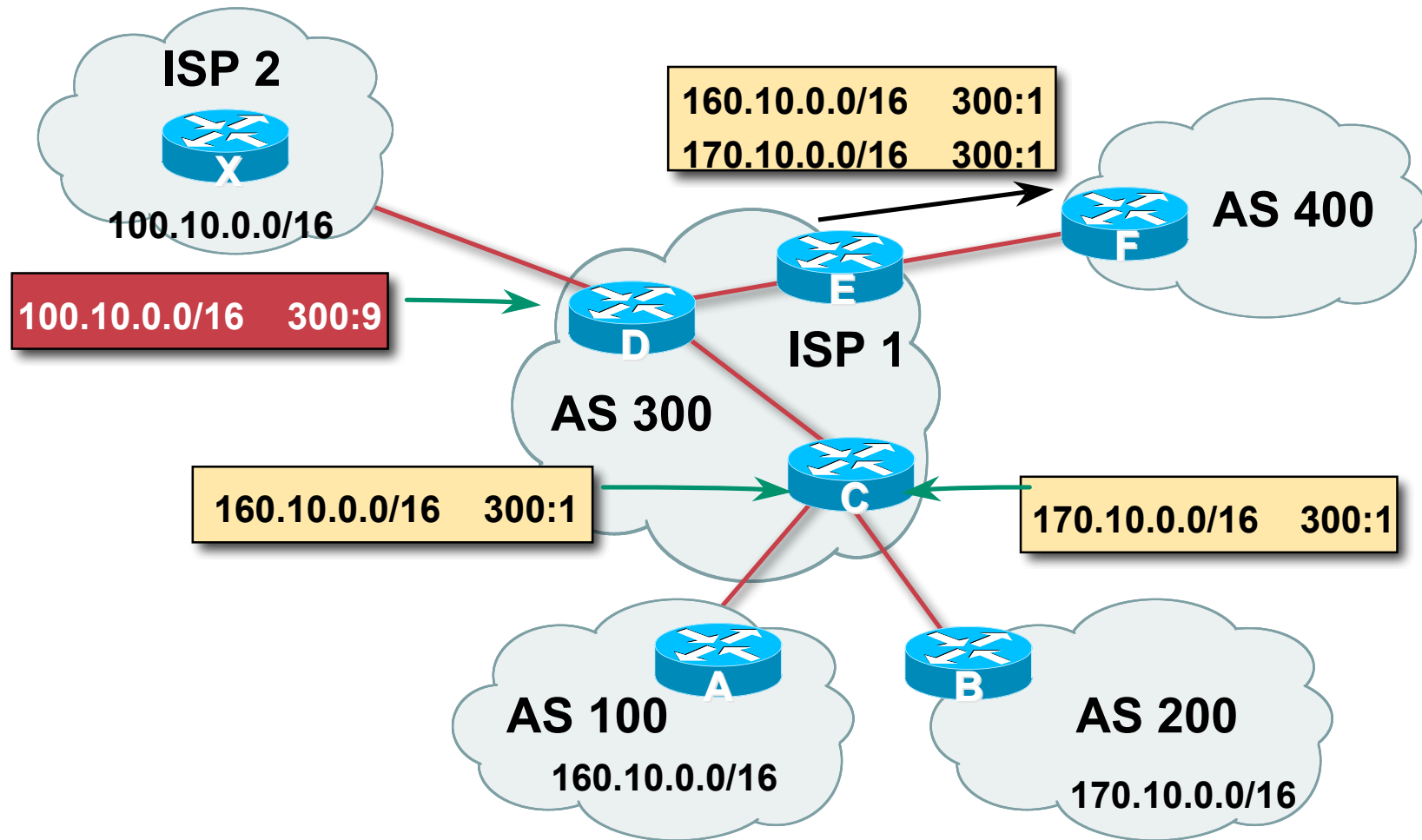
- **Best path to AS4 from AS1 is always via B due to local-pref**
- **But packets arriving at A from AS4 over the direct C to A link will pass the RPF check as that path has a priority due to the weight being set**

If weight was not set, best path back to AS4 would be via B, and the RPF check would fail

Community

- **Communities are described in RFC1997**
Transitive & Optional attribute
- **32 bit integer**
Represented as two 16 bit integers (RFC1997/8)
Common format is <local-ASN>:xx
0:0 to 0:65535 and 65535:0 to 65535:65535 are reserved
- **Used to group destinations**
Each destination could be member of multiple communities
- **Very useful for applying policies within and between ASes**

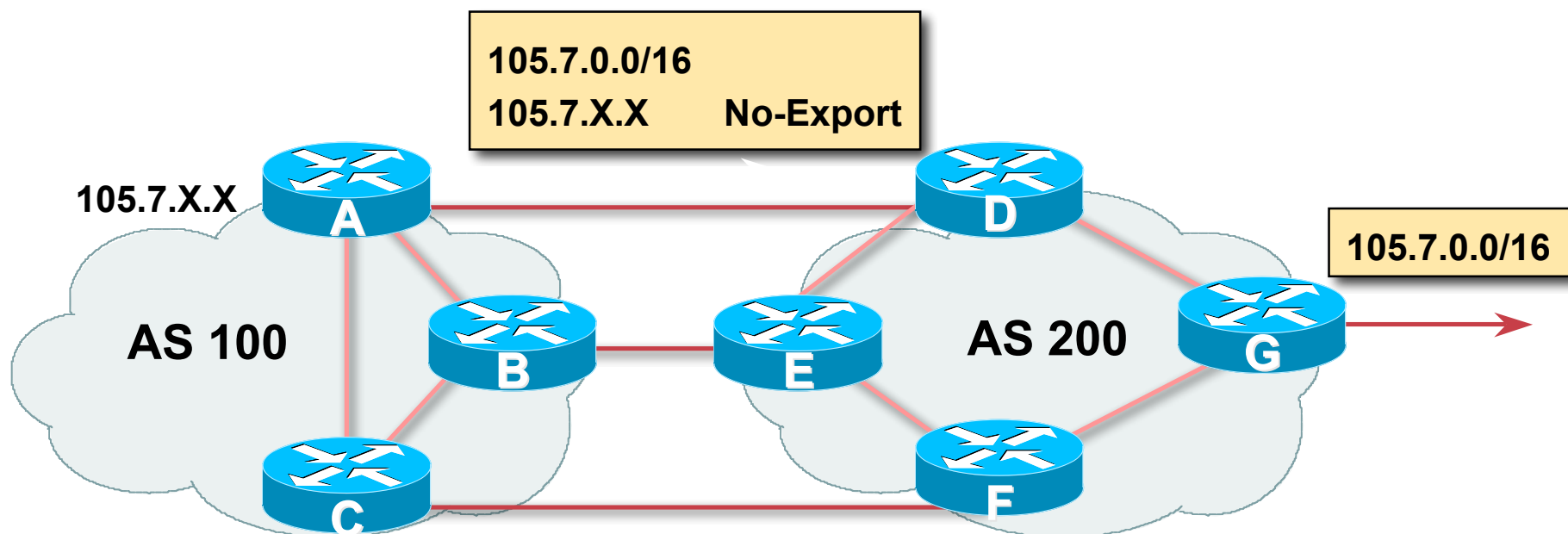
Community



Well-Known Communities

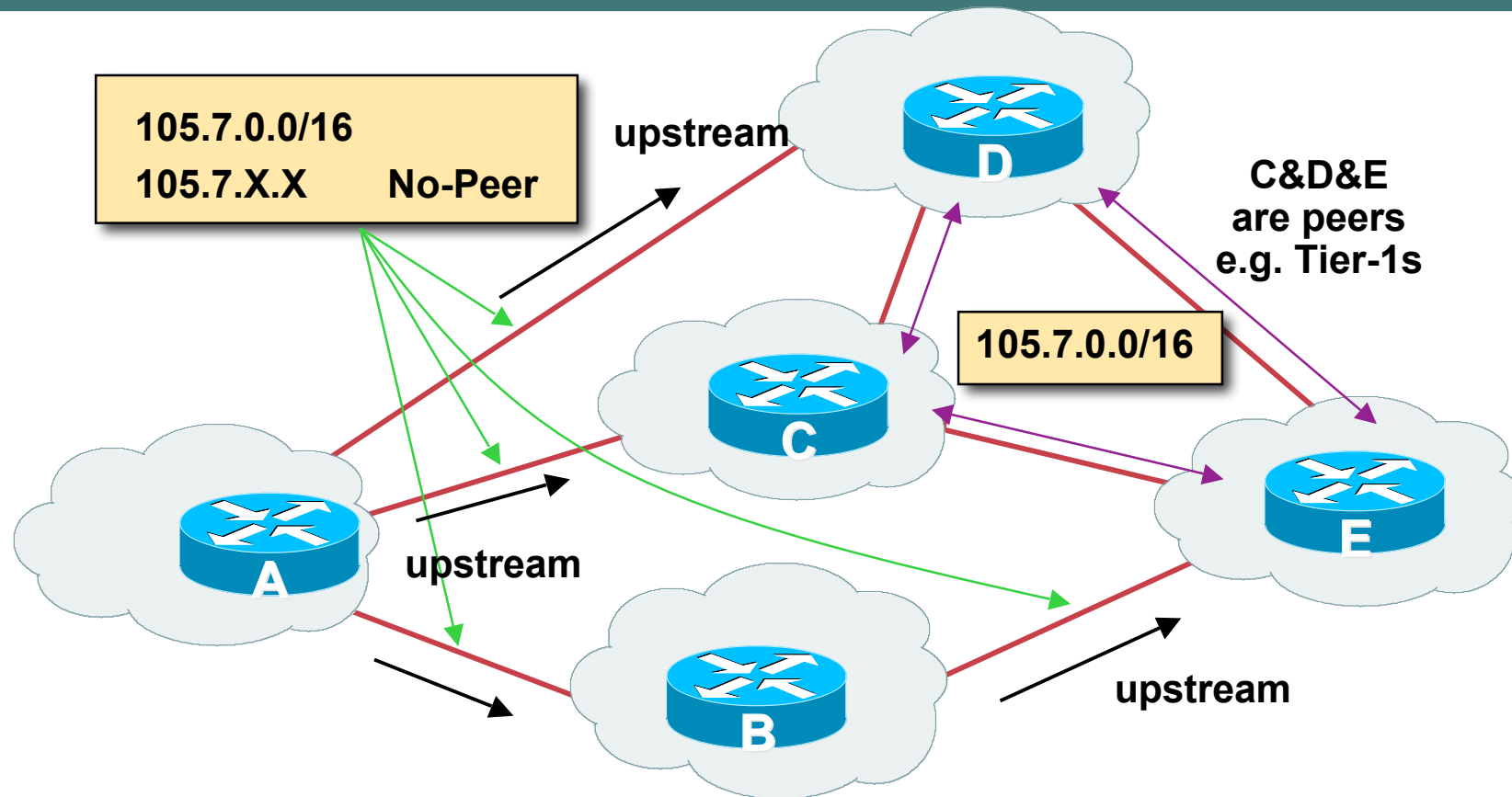
- **Several well known communities**
www.iana.org/assignments/bgp-well-known-communities
- **no-export** **65535:65281**
do not advertise to any eBGP peers
- **no-advertise** **65535:65282**
do not advertise to any BGP peer
- **no-export-subconfed** **65535:65283**
do not advertise outside local AS (only used with confederations)
- **no-peer** **65535:65284**
do not advertise to bi-lateral peers (RFC3765)

No-Export Community



- AS100 announces aggregate and subprefixes
aim is to improve loadsharing by leaking subprefixes
- Subprefixes marked with **no-export** community
- Router G in AS200 does not announce prefixes with **no-export** community set

No-Peer Community



- Sub-prefixes marked with **no-peer** community are not sent to bi-lateral peers

They are only sent to upstream providers

Summary

Attributes in Action

```
Router1>sh ip bgp
```

```
BGP table version is 28, local router ID is 100.1.15.224
```

```
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
*> 100.1.0.0/20	0.0.0.0	0		32768	i
*>i100.1.16.0/20	100.1.31.224	0	100	0	i
*>i100.1.32.0/19	100.1.63.224	0	100	0	i
...					



BGP Path Selection Algorithm

Why is this the best path?

BGP Path Selection Algorithm

- **Do not consider path if no route to next hop**
- **Do not consider iBGP path if not synchronised**
- **Highest weight (local to router)**
- **Highest local preference (global within AS)**
- **Prefer locally originated route**
- **Shortest AS path**

BGP Path Selection Algorithm (continued)

- **Lowest origin code**

IGP < EGP < incomplete

- **Lowest Multi-Exit Discriminator (MED)**

If `bgp deterministic-med`, order the paths before comparing

If `bgp always-compare-med`, then compare for all paths

otherwise MED only considered if paths are from the same AS (default)

BGP Path Selection Algorithm (continued)

- **Prefer eBGP path over iBGP path**
- **Path with lowest IGP metric to next-hop**
- **For eBGP paths:**
 - If multipath is enabled, install N parallel paths in forwarding table**
 - If router-id is the same, go to next step**
 - If router-id is not the same, select the oldest path**

BGP Path Selection Algorithm (continued)

- **Lowest router-id (originator-id for reflected routes)**
- **Shortest cluster-list**
 - Client **must** be aware of Route Reflector attributes!
- **Lowest neighbour address**



Applying Policy with BGP

How to use the “tools”

Applying Policy with BGP

- **Policy-based on AS path, community or the prefix**
- **Rejecting/accepting selected routes**
- **Set attributes to influence path selection**
- **Tools:**
 - Prefix-list (filters prefixes)**
 - Filter-list (filters ASes)**
 - Route-maps and communities**

Policy Control – Prefix List

- **Per neighbour prefix filter**
incremental configuration
- **Inbound or Outbound**
- **Based upon network numbers (using familiar IPv4 address/mask format)**
- **Using access-lists for filtering prefixes was deprecated long ago**

Strongly discouraged!

Prefix-list Command

- **Syntax:**

```
[no] ip prefix-list <list-name> [seq <seq-value>] permit|deny  
<network>/<len> [ge <ge-value>] [le <le-value>]
```

<network>/<len>: The prefix and its length

ge <ge-value>: "greater than or equal to"

le <le-value>: "less than or equal to"

Both "ge" and "le" are optional. Used to specify the range of the prefix length to be matched for prefixes that are more specific than <network>/<len>

Prefix Lists – Examples

- **Deny default route**

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

- **Permit the prefix 35.0.0.0/8**

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

- **Deny the prefix 172.16.0.0/12**

```
ip prefix-list EG deny 172.16.0.0/12
```

- **In 192/8 allow up to /24**

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

This allows all prefix sizes in the 192.0.0.0/8 address block, apart from /25, /26, /27, /28, /29, /30, /31 and /32.

Prefix Lists – Examples

- **In 192/8 deny /25 and above**

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

This denies all prefix sizes /25, /26, /27, /28, /29, /30, /31 and /32 in the address block 192.0.0.0/8.

It has the same effect as the previous example

- **In 193/8 permit prefixes between /12 and /20**

```
ip prefix-list EG permit 193.0.0.0/8 ge 12 le 20
```

This denies all prefix sizes /8, /9, /10, /11, /21, /22, ... and higher in the address block 193.0.0.0/8.

- **Permit all prefixes**

```
ip prefix-list EG permit 0.0.0.0/0 le 32
```

0.0.0.0 matches all possible addresses, “0 le 32” matches all possible prefix lengths

Policy Control – Prefix List

- **Example Configuration**

```
router bgp 100
  network 105.7.0.0 mask 255.255.0.0
  neighbor 102.10.1.1 remote-as 110
  neighbor 102.10.1.1 prefix-list PEER-IN in
  neighbor 102.10.1.1 prefix-list PEER-OUT out
!
ip prefix-list PEER-IN deny 218.10.0.0/16
ip prefix-list PEER-IN permit 0.0.0.0/0 le 32
ip prefix-list PEER-OUT permit 105.7.0.0/16
ip prefix-list PEER-OUT deny 0.0.0.0/0 le 32
```

Policy Control – Filter List

- Filter routes based on AS path
- Inbound or Outbound
- Example Configuration:

```
router bgp 100
  network 105.7.0.0 mask 255.255.0.0
  neighbor 102.10.1.1 filter-list 5 out
  neighbor 102.10.1.1 filter-list 6 in
!
ip as-path access-list 5 permit ^200$
ip as-path access-list 6 permit ^150$
```


Policy Control – Regular Expressions

- **Like Unix regular expressions**
 - .** Match one character
 - *** Match any number of preceding expression
 - +** Match at least one of preceding expression
 - ^** Beginning of line
 - \$** End of line
 - _** Beginning, end, white-space, brace
 - |** Or
 - ()** brackets to contain expression

Policy Control – Regular Expressions

- **Simple Examples**

.*	match anything
.+	match at least one character
^\$	match routes local to this AS
_1800\$	originated by AS1800
^1800_	received from AS1800
1800	via AS1800
_790_1800_	via AS1800 and AS790
(1800)+	multiple AS1800 in sequence (used to match AS-PATH prepends)
\\(65530\\)	via AS65530 (confederations)

Policy Control – Regular Expressions

- **Not so simple Examples**

^[0-9]+\$

Match AS_PATH length of one

^[0-9]+_[0-9]+\$

Match AS_PATH length of two

^[0-9]*_[0-9]+\$

Match AS_PATH length of one or two

^[0-9]*_[0-9]*\$

**Match AS_PATH length of one or two
(will also match zero)**

^[0-9]+_[0-9]+_[0-9]+\$

Match AS_PATH length of three

(701|1800)

**Match anything which has gone
through AS701 or AS1800**

1849(.+_)12163\$

**Match anything of origin AS12163
and passed through AS1849**

Policy Control – Route Maps

- A route-map is like a “programme” for IOS
- Has “line” numbers, like programmes
- Each line is a separate condition/action
- Concept is basically:
 - if match then do expression and exit*
 - else*
 - if match then do expression and exit*
 - else etc*

Route Maps – Caveats

- **Lines can have multiple set statements but only one match statement**
- **Line with only a set statement**
 - all prefixes are matched and set**
 - any following lines are ignored**
- **Line with a match/set statement and no following lines**
 - only prefixes matching go through**
 - the rest are dropped**

Route Maps – Caveats

- **Example**

omitting the third line below means that prefixes not matching **list-one** or **list-two** are dropped

```
route-map sample permit 10
  match ip address prefix-list list-one
  set local-preference 120
!
route-map sample permit 20
  match ip address prefix-list list-two
  set local-preference 80
!
route-map sample permit 30 ! Don't forget this
```

Policy Control – Route Maps

- **Example Configuration – route map and prefix-lists**

```
router bgp 100
  neighbor 1.1.1.1 route-map infiltrer in
  !
route-map infiltrer permit 10
  match ip address prefix-list HIGH-PREF
  set local-preference 120
  !
route-map infiltrer permit 20
  match ip address prefix-list LOW-PREF
  set local-preference 80
  !
ip prefix-list HIGH-PREF permit 10.0.0.0/8
ip prefix-list LOW-PREF permit 20.0.0.0/8
```

Policy Control – Route Maps

- Example Configuration – route map and filter lists

```
router bgp 100
  neighbor 102.10.1.2 remote-as 200
  neighbor 102.10.1.2 route-map filter-on-as-path in
!
route-map filter-on-as-path permit 10
  match as-path 1
  set local-preference 80
!
route-map filter-on-as-path permit 20
  match as-path 2
  set local-preference 200
!
ip as-path access-list 1 permit _150$
ip as-path access-list 2 permit _210_
```


Policy Control – Route Maps

- **Example configuration of AS-PATH prepend**

```
router bgp 300
  network 105.7.0.0 mask 255.255.0.0
  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
```
- **Use your **own** AS number when prepending**
Otherwise BGP loop detection may cause disconnects

Policy Control – Route Maps

- **Route Map MATCH Articles**

as-path

clns address

clns next-hop

clns route-source

community

interface

ip address

ip next-hop

ip route-source

length

metric

nlri

route-type

tag

Policy Control – Route Maps

- **Route map SET Articles**

as-path

automatic-tag

clns

comm-list

community

dampening

default interface

interface

ip default next-hop

ip next-hop

Policy Control – Route Maps

- **Route map SET Articles**

ip precedence

ip qos-group

ip tos

level

local preference

metric

metric-type

next-hop

nlri multicast

nlri unicast

origin

tag

traffic-index

weight

Policy Control – Matching Communities

- **Example Configuration**

```
router bgp 100
  neighbor 102.10.1.2 remote-as 200
  neighbor 102.10.1.2 route-map filter-on-community in
!
route-map filter-on-community permit 10
  match community 1
  set local-preference 50
!
route-map filter-on-community permit 20
  match community 2 exact-match
  set local-preference 200
!
ip community-list 1 permit 150:3 200:5
ip community-list 2 permit 88:6
```

Policy Control – Setting Communities

- **Example Configuration**

```
router bgp 100
  network 105.7.0.0 mask 255.255.0.0
  neighbor 102.10.1.1 remote-as 200
  neighbor 102.10.1.1 send-community
  neighbor 102.10.1.1 route-map set-community out
!
route-map set-community permit 10
  match ip address prefix-list NO-ANNOUNCE
  set community no-export
!
route-map set-community permit 20
  match ip address prefix-list AGGREGATE
!
ip prefix-list NO-ANNOUNCE permit 105.7.0.0/16 ge 17
ip prefix-list AGGREGATE permit 105.7.0.0/16
```

Aggregation Policies

- **Suppress Map**

Used to suppress selected more-specific prefixes (e.g. defined through a route-map) in the absence of the **summary-only** keyword.

- **Unsuppress Map**

Used to unsuppress selected more-specific prefixes per BGP peering when the **summary-only** keyword is in use.

Aggregation Policies – Suppress Map

- Example

```
router bgp 100
  network 102.10.10.0
  network 102.10.11.0
  network 102.10.12.0
  network 102.10.33.0
  network 102.10.34.0
  aggregate-address 102.10.0.0 255.255.0.0 suppress-map block-net
  neighbor 102.5.7.2 remote-as 200
!
route-map block-net permit 10
  match ip address prefix-list SUPPRESS
!
ip prefix-list SUPPRESS permit 102.10.8.0/21 le 32
ip prefix-list SUPPRESS deny 0.0.0.0/0 le 32
!
```


Aggregation Policies – Suppress Map

- **show ip bgp** on the local router

```
router1#sh ip bgp
```

```
BGP table version is 11, local router ID is 102.5.7.1
```

```
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
*> 102.10.0.0/16	0.0.0.0			32768	i
s> 102.10.10.0	0.0.0.0	0		32768	i
s> 102.10.11.0	0.0.0.0	0		32768	i
s> 102.10.12.0	0.0.0.0	0		32768	i
*> 102.10.33.0	0.0.0.0	0		32768	i
*> 102.10.34.0	0.0.0.0	0		32768	i

Aggregation Policies – Suppress Map

- **show ip bgp** on the remote router

```
router2#sh ip bgp
```

```
BGP table version is 90, local router ID is 102.5.7.2
```

```
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
*> 102.10.0.0/16	102.5.7.1			0 100	i
*> 102.10.33.0	102.5.7.1	0		0 100	i
*> 102.10.34.0	102.5.7.1	0		0 100	i

Aggregation Policies – Unsuppress Map

- Example

```
router bgp 100
  network 102.10.10.0
  network 102.10.11.0
  network 102.10.12.0
  network 102.10.33.0
  network 102.10.34.0
  aggregate-address 102.10.0.0 255.255.0.0 summary-only
  neighbor 102.5.7.2 remote-as 200
  neighbor 102.5.7.2 unsuppress-map leak-net
!
route-map leak-net permit 10
  match ip address prefix-list LEAK
!
ip prefix-list LEAK permit 102.10.8.0/21 le 32
ip prefix-list LEAK deny 0.0.0.0/0 le 32
!
```

Aggregation Policies – Unsuppress Map

- **show ip bgp** on the local router

```
router1#sh ip bgp
```

```
BGP table version is 11, local router ID is 102.5.7.1
```

```
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
*> 102.10.0.0/16	0.0.0.0			32768	i
s> 102.10.10.0	0.0.0.0	0		32768	i
s> 102.10.11.0	0.0.0.0	0		32768	i
s> 102.10.12.0	0.0.0.0	0		32768	i
s> 102.10.33.0	0.0.0.0	0		32768	i
s> 102.10.34.0	0.0.0.0	0		32768	i

Aggregation Policies – Unsuppress Map

- **show ip bgp** on the remote router

```
router2#sh ip bgp
```

```
BGP table version is 90, local router ID is 102.5.7.2
```

```
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
*> 102.10.0.0/16	102.5.7.1			0 100	i
*> 102.10.10.0	102.5.7.1	0		0 100	i
*> 102.10.11.0	102.5.7.1	0		0 100	i
*> 102.10.12.0	102.5.7.1	0		0 100	i

Aggregation Policies – Aggregate Address

- **Summary-only used**

all subprefixes suppressed

unsuppress-map to selectively leak subprefixes

bgp per neighbour configuration

- **Absence of summary-only**

no subprefixes suppressed

suppress-map to selectively suppress subprefixes

bgp global configuration



BGP Attributes and Policy Control

ISP/IXP Workshops