kathara lab

bgp: prefix-filtering with frr

Version	2.1
Author(s)	G. Di Battista, M. Patrignani, M. Pizzonia, F. Ricci, M. Rimondini
E-mail	contact@kathara.org
Web	http://www.kathara.org/
Description	examples of filtering rules; kathara version of a netkit lab

copyright notice

preconditions

- for this lab we assume you have chosen “kathara/frr” as the default image of your Kathará installation
  - execute “kathara settings”
    - select “choose default image”
    - select “kathara/frr”
    - exit from the settings procedure

applying policies

1. announcement filtering
   - send/accept an announcement only if some condition is verified
   - commands:
     - **`prefix-list`** used to filter prefixes
     - **`filter-list`** used to filter as numbers
2. announcement tuning
   - attach to your announcement some information (attributes) that should be considered by the receiver
   - commands:
     - **`route-map`**
     - **`access-list`** used to match prefixes or as-paths in a **`route-map`**

bgp prefix filtering

prefix filtering commands

command syntax

```
neighbor <neighbor-ip> prefix-list <p-list-name> in
```

command syntax

```
neighbor <neighbor-ip> prefix-list <p-list-name> out
```

command syntax

```
ip prefix-list <p-list-name> permit <network/mask>
```

command syntax

```
ip prefix-list <p-list-name> deny <network/mask>
```

prefix filtering: example

frr configuration file

```
router bgp 1
! no bgp ebgp-requires-policy (not needed anymore)
neighbor 193.10.11.2 remote-as 2
neighbor 193.10.11.2 description Router 2 of AS2
!
network 195.11.14.0/24
network 195.11.15.0/24
neighbor 193.10.11.2 prefix-list partialIn in
neighbor 193.10.11.2 prefix-list partialOut out
!
ip prefix-list partialOut seq 5 permit 195.11.14.0/24
ip prefix-list partialIn seq 5 deny 200.1.1.0/24
ip prefix-list partialIn seq 10 permit any
```

only `195.11.14.0/24` is announced to neighbor `193.10.11.2`
all with the exception of `200.1.1.0/24` is accepted from `193.10.11.2`

about `prefix-lists`

- **`prefix-list`** entries are ordered according to a sequence number
  - explicitly assigned by the user; example:
    - `ip prefix-list myPfxList seq 5 permit 10.0.0.0/8`
  - implicitly assigned by zebra; example:
    - `ip prefix-list myPfxList permit 10.0.0.0/8`
      `ip prefix-list myPfxList permit 20.0.0.0/8`
      is automatically turned to:
      `ip prefix-list myPfxList seq 5 permit 10.0.0.0/8`
      `ip prefix-list myPfxList seq 10 permit 20.0.0.0/8`

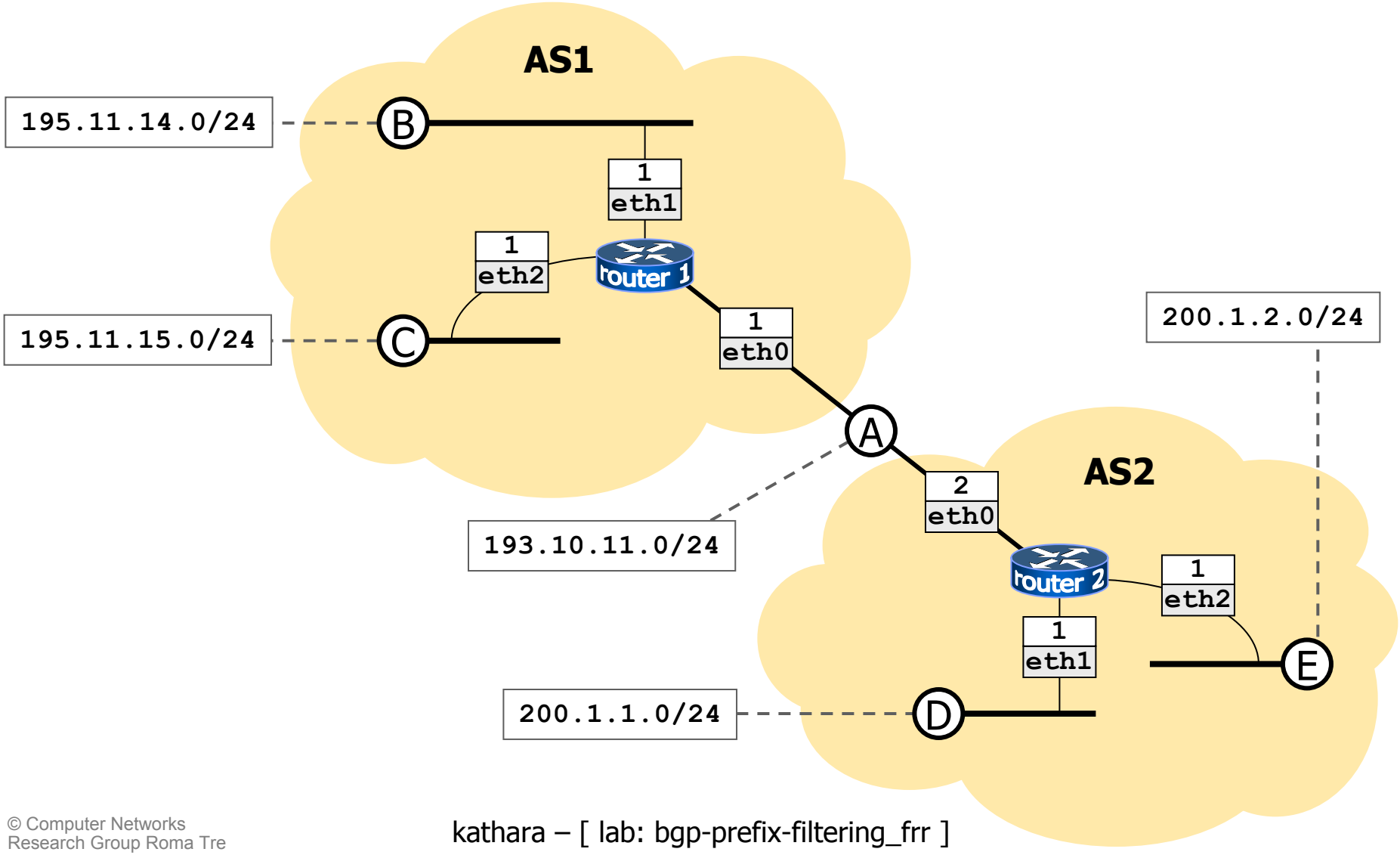
about `prefix-lists`

- the first matching entry is applied; example:
  - `ip prefix-list letThru permit 10.0.0.0/8`
    `ip prefix-list letThru deny any`
    accepts `10.0.0.0/8` only
  - `ip prefix-list throwAway deny any`
    `ip prefix-list throwAway permit 10.0.0.0/8`
    rejects everything

`prefix-list` defaults

- in zebra, **`prefix-lists`** default to **`deny`**; for example:
  - **`ip prefix-list myPrefixList permit 10.0.0.0/8`** filters out everything but **`10.0.0.0/8`**
  - **`ip prefix-list myPrefixList deny 10.0.0.0/8`** filters out everything
- referencing an undefined **`prefix-list`** in a **`neighbor`** statement is equivalent to **`deny`**ing **`any`**thing; for example:
  - **`neighbor 10.0.0.1 prefix-list undefinedPrefixList in`** filters out everything if **`undefinedPrefixList`** is not defined

prefix filtering



prefix filtering

- start the lab

host machine

```
user@localhost:~$ cd kathara-lab_bgp-prefix-filtering_frr
user@localhost:~/kathara-lab_bgp-prefix-filtering_frr$ kathara lstart
```

- check the frr configuration file

router1

```
router1:~# less /etc/frr/frr.conf
```

- check the frr log file

router1

```
router1:~# less /var/log/frr/frr.log
```

prefix filtering

- check the routing table

router1

```
root@router1:/# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
193.10.11.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
195.11.14.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
195.11.15.0     0.0.0.0         255.255.255.0   U     0      0        0 eth2
200.1.2.0       193.10.11.2     255.255.255.0   UG    20     0        0 eth0
root@router1:/# vtysh

Hello, this is FRRouting (version 7.5.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

router1-frr# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup

C>* 193.10.11.0/24 is directly connected, eth0, 00:08:56
C>* 195.11.14.0/24 is directly connected, eth1, 00:08:56
C>* 195.11.15.0/24 is directly connected, eth2, 00:08:56
B>* 200.1.2.0/24 [20/0] via 193.10.11.2, eth0, weight 1, 00:08:54
router1-frr#
```

prefix filtering

- check the frr cli (command line interface)

router1

```
router1-frr# show ip bgp neighbors
BGP neighbor is 193.10.11.2, remote AS 2, local AS 1, external link
 Description: Router 2 of AS2
Hostname: router2
  BGP version 4, remote router ID 200.1.2.1,
                 local router ID 195.11.15.1
  BGP state = Established, up for 00:11:36
  Last read 00:00:36, Last write 00:00:36
  Hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    AddPath:
      IPv4 Unicast: RX advertised IPv4 Unicast and received
    Route refresh: advertised and received(old & new)
    Address Family IPv4 Unicast: advertised and received
...
...
router1-frr# show ip bgp 200.1.1.0
% Network not in table
router1-frr#
```

prefix filtering

- terminate the lab

host machine

```
user@localhost:~/kathara-lab_bgp-prefix-filtering_frr$ kathara lclean
```

bgp attributes

attributes

- a bgp announcement is a “bag” of attributes
- attributes may be
  - “well-known” or optional
    - well-known attributes are understood by any bgp4 speaker
  - mandatory or discretionary
    - mandatory attributes must be present in updates
  - transitive or nontransitive
    - transitive attributes are passed when received
    - nontransitive attributes traverse a single peering

attribute list

- prefix
  - the section of ip space announced
- as-path
  - the sequence of traversed ases
- origin
  - igp (route is interior to the originating as)
  - egp (route learned via the egp protocol)
  - incomplete (route learned in some other way)
- next-hop
  - to be inserted in the routing table
- metric (multi-exit-discriminator)
  - asking another as to prefer lower values of it
- local-pref
  - prefer higher values
- atomic aggregate
- aggregator
- weight
  - cisco proprietary

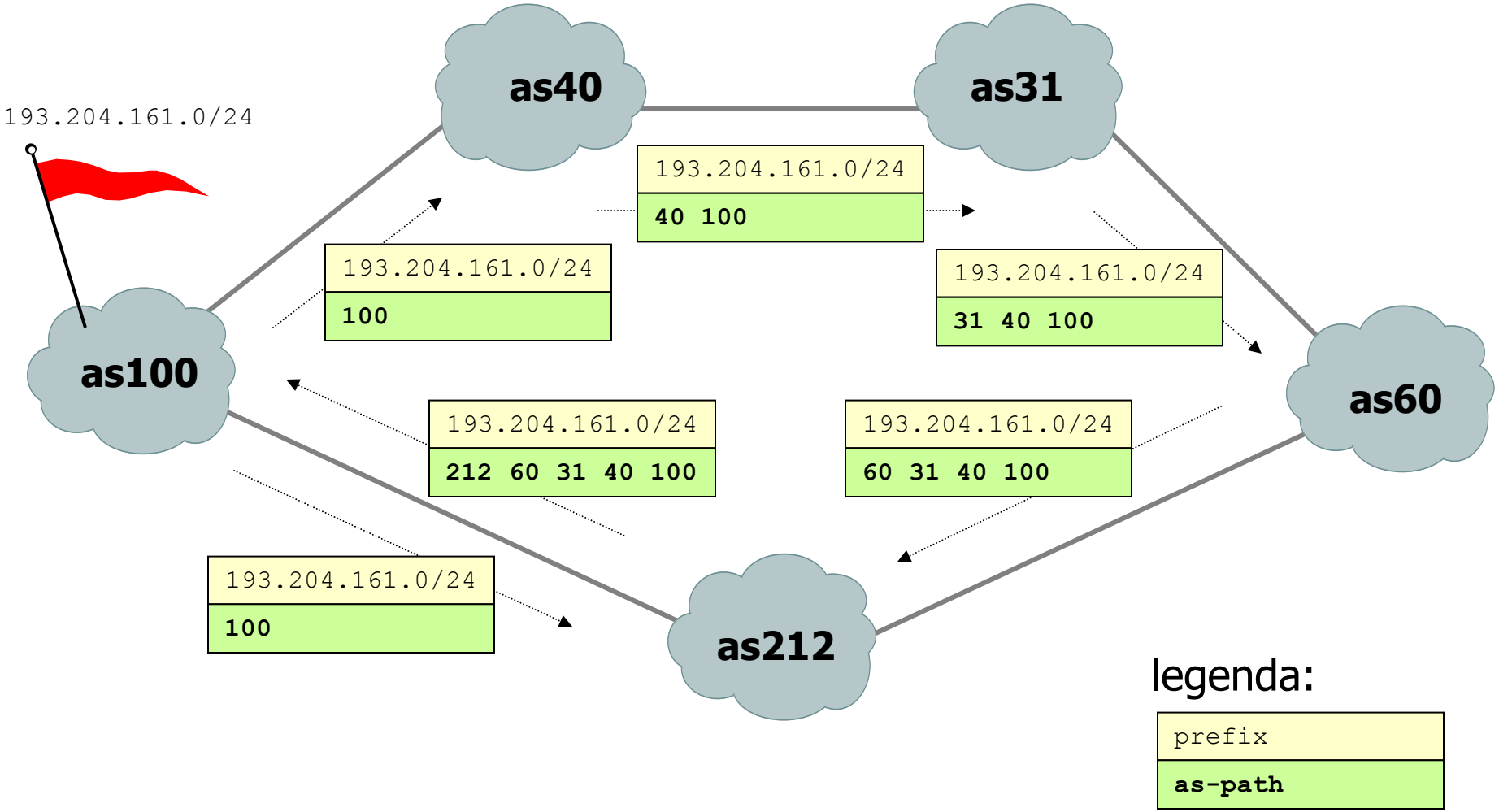
well-known attributes

- mandatory well-known
  - as-path: the sequence of traversed ASes
  - next-hop: to be inserted in the routing table; in i-bgp stays unchanged
  - origin
- discretionary well-known
  - local preference: asking i-bgp peers to prefer higher values of it
  - atomic aggregate

optional attributes

- non transitive optional
  - multi-exit discriminator: asking other ASes to prefer lower values of it
- transitive optional
  - aggregator
  - community

attributes: prefix & as-path



attributes: as-path

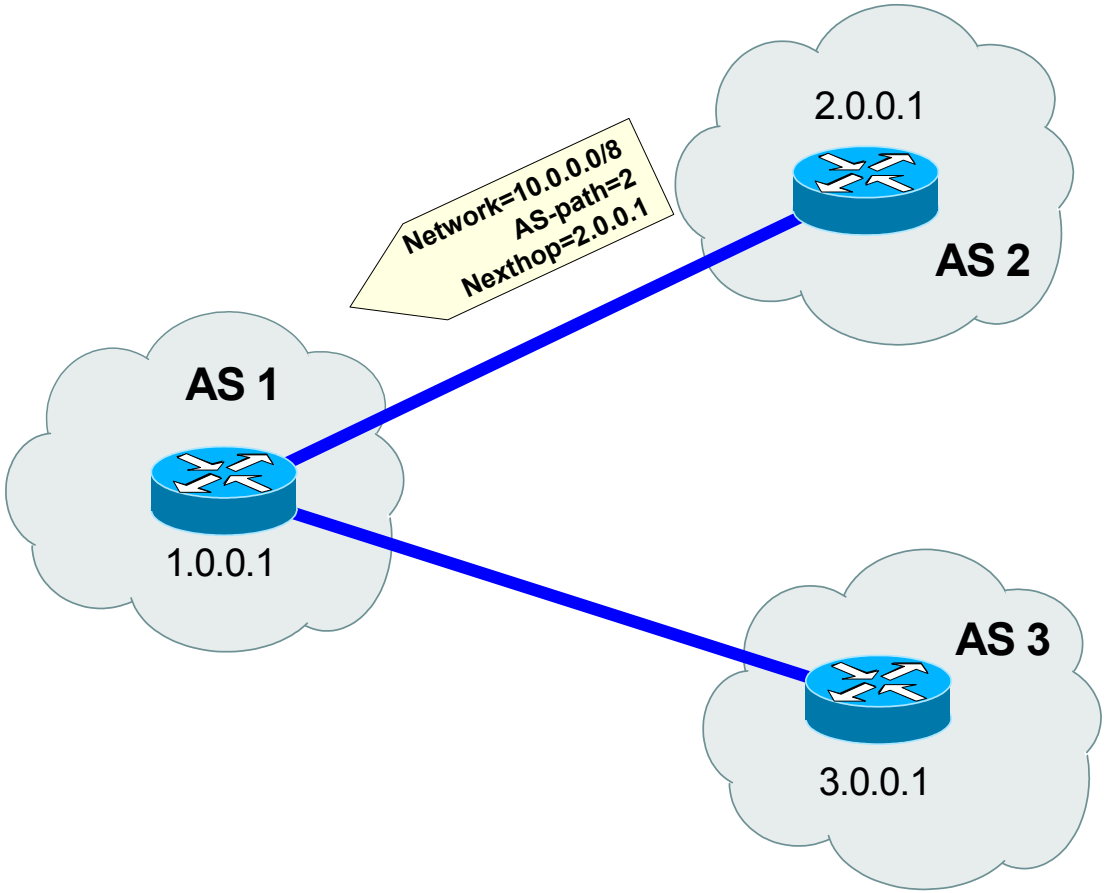
- the as-path is empty when a local route is inserted into the BGP table
- when the announcement goes to a different as (e-BGP) the as number is inserted at the end of the as-path (prepending)
  - a router knows which and how many ases should be traversed to reach the destination
  - loops are avoided
  - policies can be applied
- in i-BGP the as-path does not change

attributes: nexthop

- where to send packets for a specific ip network
- usually, the nexthop is the router that sends the announcements
  - exceptions:
    - "*shared media*" (ethernet, etc..)
    - i-BGP announcements of networks learned using e-BGP

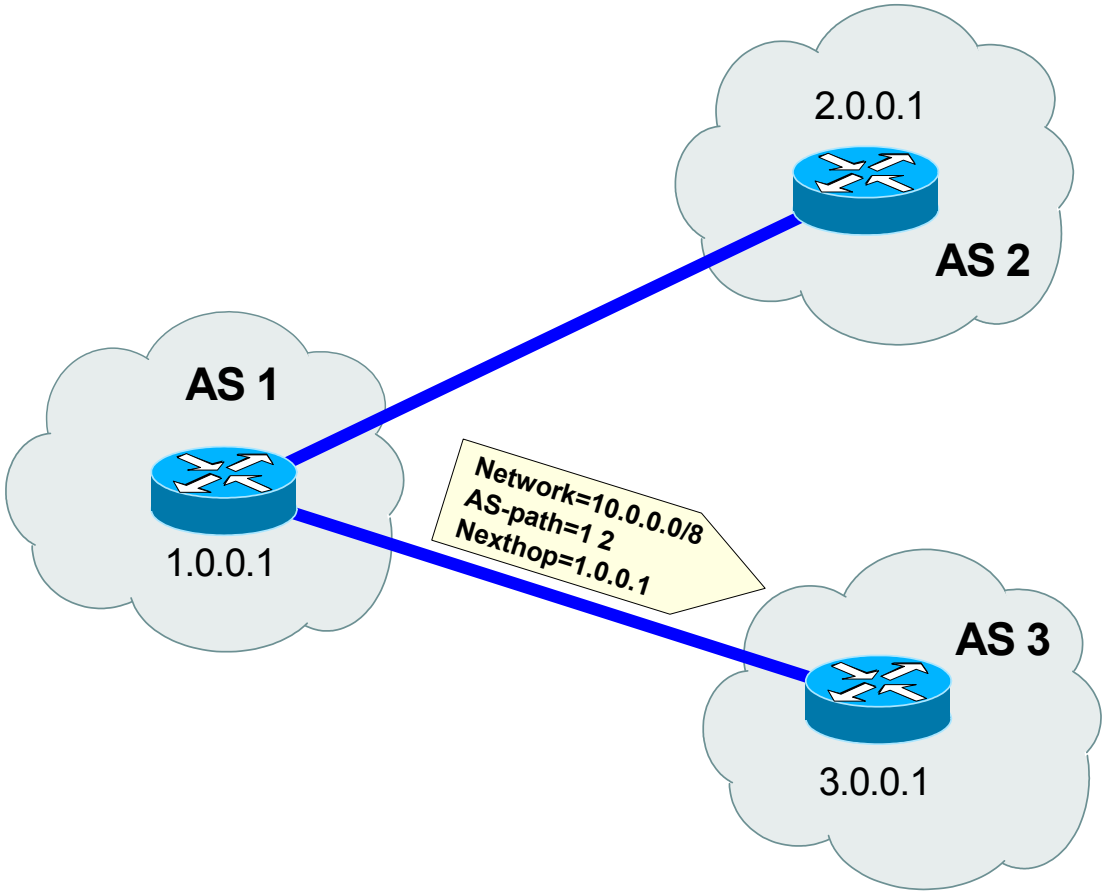
attributes: nexthop

- as 2 router sends an announcement to as 1 specifying 2.0.0.1 as the nexthop



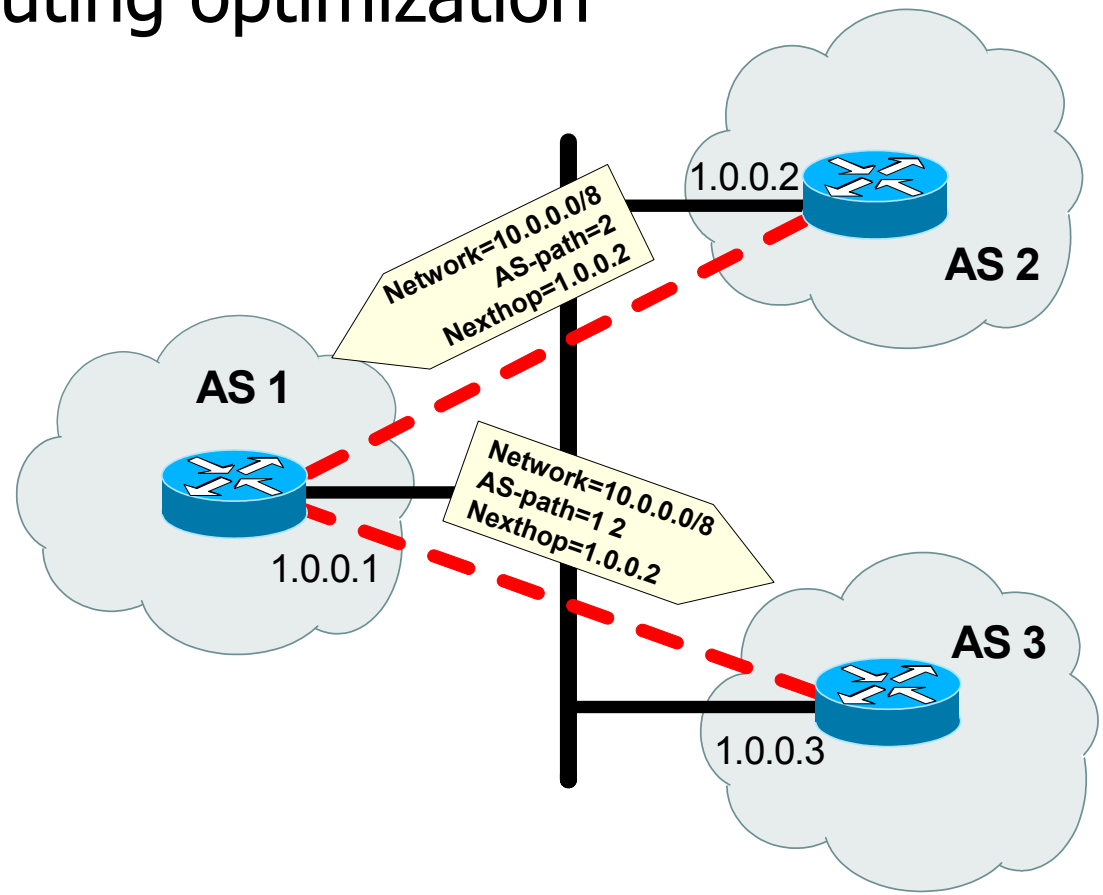
attributes: nexthop

- as 1 router changes the nexthop



attributes: nexthop

- shared segment: nexthop stays unchanged
  - routing optimization

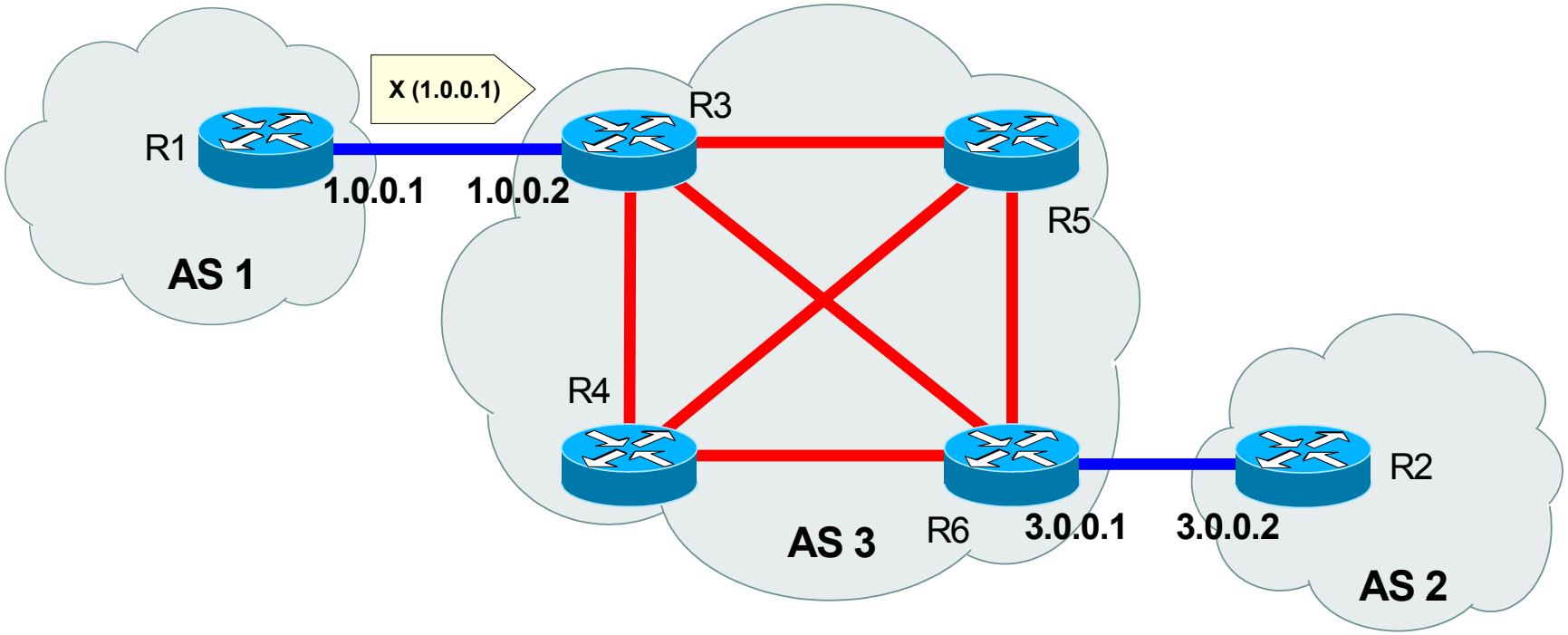


attributes: nexthop

- if a route learned from e-BGP is propagated using i-BGP, then the nexthop remains the same
  - nexthop equal to the address of the remote peer
- internal routers perform a "recursive lookup" for understanding how to reach the nexthop
  - the routers should know, via igp, how to reach the nexthop

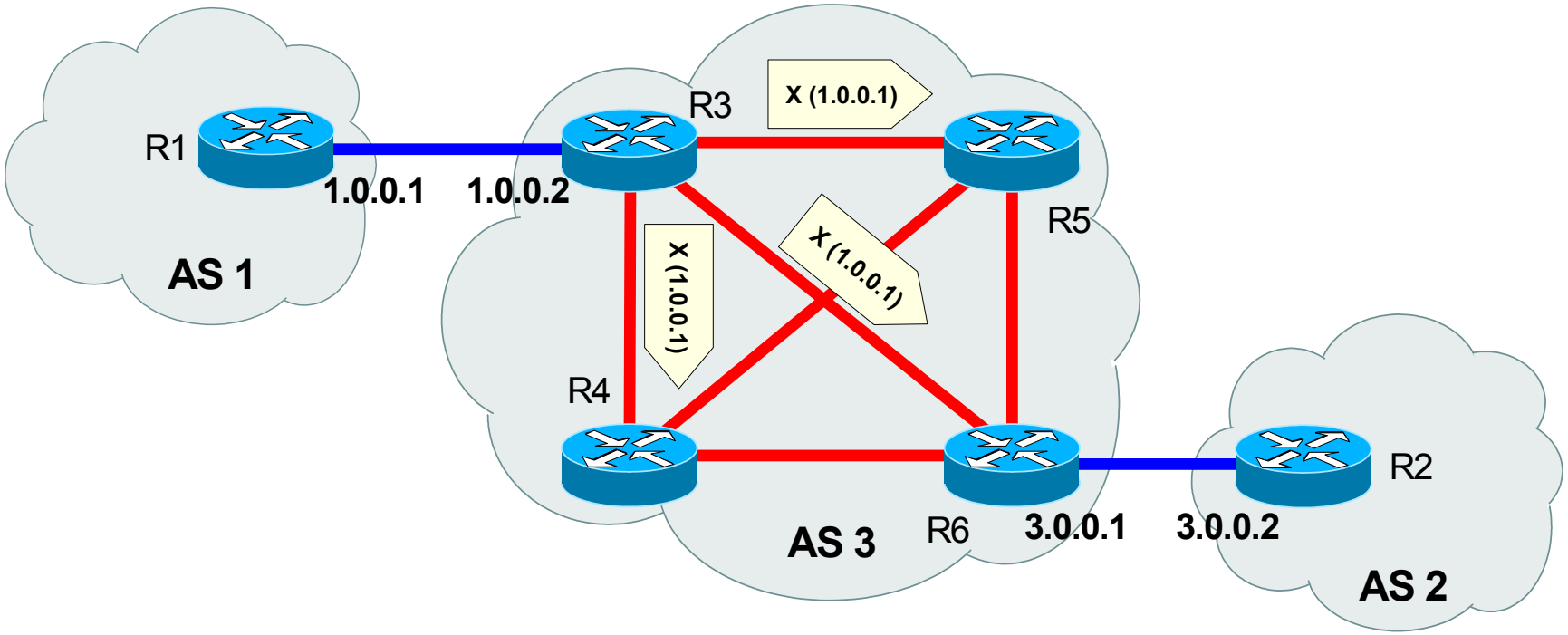
attributes: nexthop

- R1 announces network X with nexthop 1.0.0.1



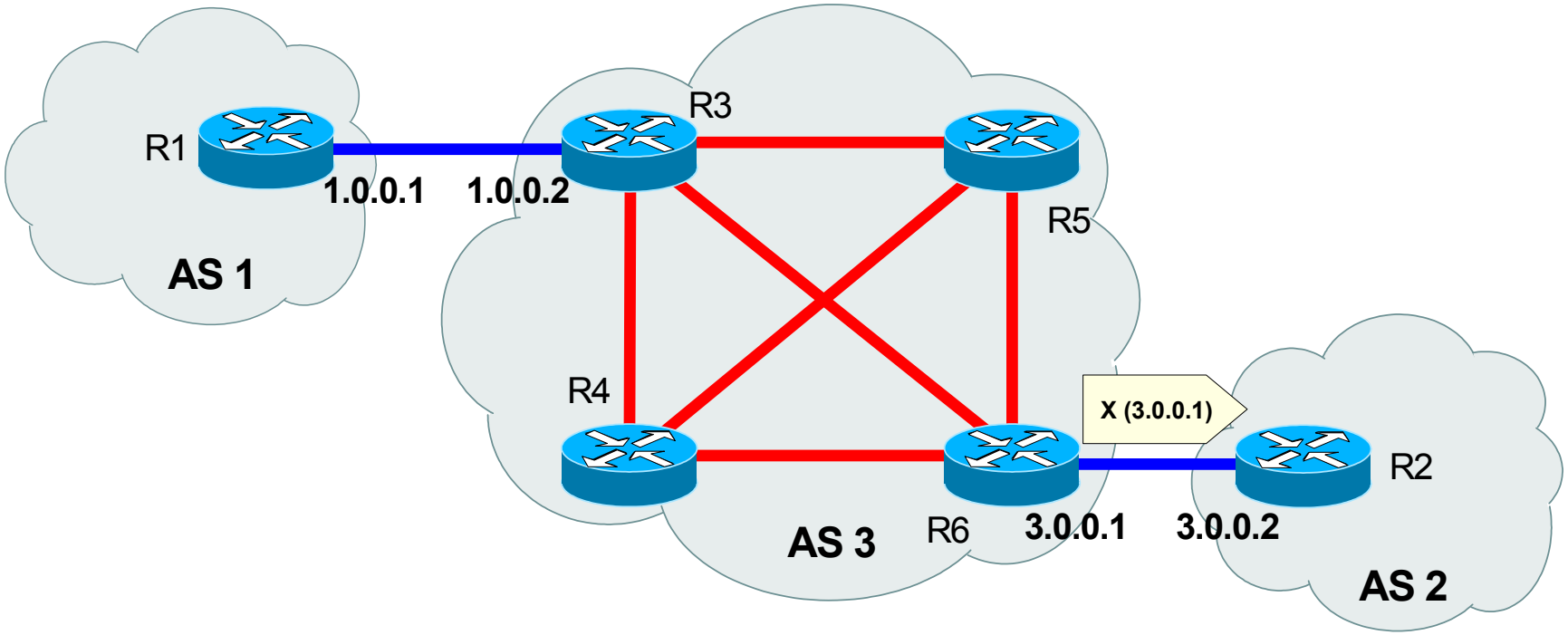
attributes: nexthop

- the announcement is propagated with i-BGP to R4, R5, and R6
  - the nexthop is unchanged



attributes: nexthop

- the nexthop changes when the announcement goes to a different as



attributes: origin

- igp
  - declared as internal by the starting AS: “network” command
- egp
  - injected into BGP by EGP: backward compatibility
- incomplete
  - generated redistributing an igp protocol

attributes: aggregator

- conveys the IP address of the router or BGP speaker generating the aggregate route
  - useful for debugging purposes

selection of the “best route” to a prefix

- each router, for each prefix, chooses one of the received announcements as the “best”
- the decision process is fully deterministic
  - no random choice is applied
- only the best routes are (possibly) announced to peers
- selection criteria:
  - more specific and less specific prefixes are considered different prefixes
    - both are injected into the routing table
  - if the next-hop is not reachable (it does not match a line of the routing table of the router) the announcement can not be selected
  - the selection is based both on the values of the attributes and on the constraints imposed by the administrator (e.g. weight)

bgp decision process (at a router)

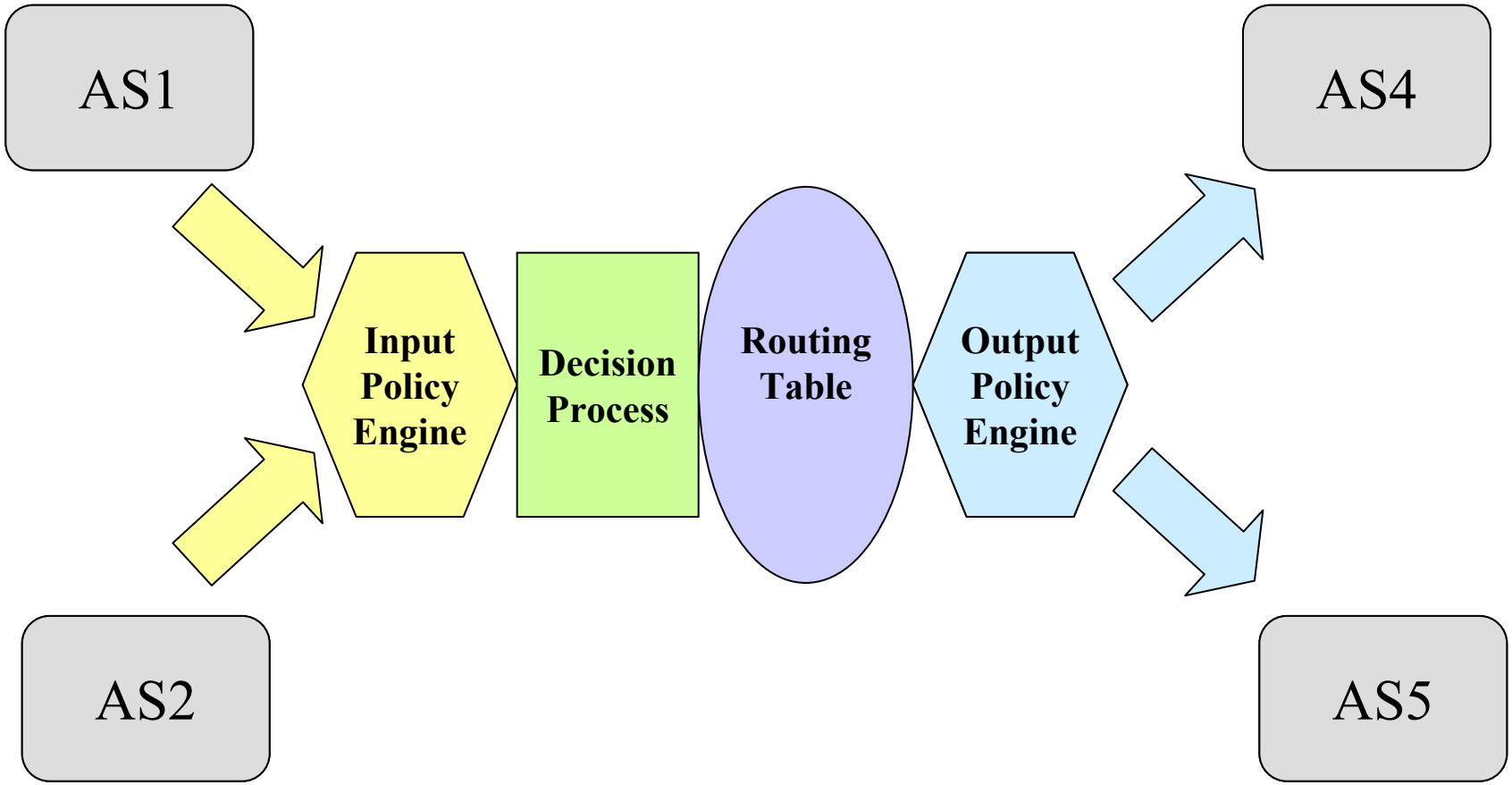
for each network prefix, select the route with:

highest priority

1. largest weight (cisco proprietary)
2. largest local preference
3. locally originated (by the router itself)
4. shortest as-path length
5. lowest origin (igp<egp<incomplete)
6. lowest multi-exit-discriminator
   (only comparable for the same neighboring as)
7. prefer ebgp over ibgp (hot potato routing)
8. lowest igp metric (to next-hop)
9. lowest router-id (of announcing peer)

lowest priority

BGP decision process and architecture



as-path filtering

as-path filtering commands

command syntax

```
neighbor <neighbor-ip> filter-list <acl-name> in
```

command syntax

```
neighbor <neighbor-ip> filter-list <acl-name> out
```

command syntax

```
ip as-path access-list <acl-name> permit <regexp>
```

command syntax

```
ip as-path access-list <acl-name> deny <regexp>
```

as-path filtering commands

- *regexp* may contain the following characters:

`.`	matches any single character
`\`	escapes special characters
`[]`	matches a range of characters
`^`	matches the beginning of a string
`$`	matches the end of a string
`?`	matches zero or one occurrence of a pattern
`*`	matches zero or more occurrences of a pattern
`+`	matches one or more occurrences of a pattern
`()`	groups characters to form a pattern
`|`	matches one of the patterns on either side
`_`	a shortcut for `[,{} ]|^|$`

as-path filtering example

frr configuration file

```
router bgp 100
network 100.1.1.0/24
neighbor 222.2.2.2 remote-as 200
neighbor 222.2.2.2 filter-list myACL in
!
ip as-path access-list myACL permit ^200_300
```

- accept from as 200 only the routes received via as 300

announcement tuning

attribute setting commands

command syntax

```
neighbor <neighbor-ip> route-map <r-map-name> in
```

command syntax

```
neighbor <neighbor-ip> route-map <r-map-name> out
```

command syntax

```
route-map <r-map-name> permit <seq-number>
  match <announce-property>
  set <attribute-setting>
  ...
```

command syntax

```
route-map <r-map-name> deny <seq-number>
  match <announce-property>
  set <attribute-setting>
  ...
```

about `route-maps`

- **`route-maps`** may consist of multiple statements
  - statements are processed in the order established by sequence numbers
  - for each received/sent announcement, only one statement is applied
    - the first one without a **`match`** condition
    - the first one that matches the announcement attributes (prefix, as-path, etc.)
  - announcements that are not matched by any statement, or that are matched by a **`deny`** statement are simply filtered out
    - **`set`** commands in a **`route-map deny`** are useless
- referencing an undefined **`route-map`** in a **`neighbor`** statement results in filtering out everything

all `match` commands

- match as-path
- match community
- match extcommunity
- match ip address
- match ip next-hop
- match ipv6 address
- match metric
- match origin

all `set` commands

- set aggregator as
- set as-path prepend
- set atomic-aggregate
- set comm-list
- set community
- set extcommunity
- set ip next-hop
- set ipv6 next-hop
- set local-preference
- set metric
- set origin
- set originator-id
- set weight

address **match** conditions

- **match ip address** can be used in conjunction with **access-lists** or **prefix-lists**

command syntax

```
match ip address <acl-name>
```

command syntax

```
match ip address prefix-list <prefix-list-name>
```

command syntax

```
access-list <acl-name> permit <network/mask>
```

command syntax

```
access-list <acl-name> deny <network/mask>
```

about `access-lists`

- an alternative construction to filter prefixes
- the **`as-path access-list`** variant allows to filter based on as-paths
- **`access-lists`** are identified by a name or an integer
  - the integer determines the type of filtering applied
    - 1-99: standard access list (filter from specific IPs)
    - 100-199: extended access list (filter by protocol and/or source/destination IP)

about `access-lists`

- no sequence numbers, still the first matching entry applies; example:
  - `access-list permissiveAcl permit any`
    `access-list permissiveAcl deny any`
    allows everything
  - `access-list restrictiveAcl deny any`
    `access-list restrictiveAcl permit any`
    discards everything
- same for `as-path access-lists`; example:
  - `ip as-path access-list noWay deny .*`
    `ip as-path access-list noWay permit ^100_200`
    discards everything

`access-list` defaults

- in zebra, **`access-lists`** default to **`deny`**
- by default, **`access-lists`** match a prefix as well as all its more specifics; for example:
  - **`access-list myList permit 193.100.0.0/16`**
    also matches **`193.100.5.0/24`**, **`193.100.192.0/25`**, etc.
  - **`access-list permissiveList permit 0.0.0.0/0`**
    matches everything(!)
- this behavior can be changed by using **`exact-match`**
- referencing an undefined **`access-list`** (e.g., in a **`filter-list`** statement) results in filtering out everything

attribute setting example

frr configuration file

```
router bgp 100
network 100.1.1.0/24
neighbor 222.2.2.2 remote-as 200
neighbor 222.2.2.2 route-map myRouteMap in
!
route-map myRouteMap permit 10
  match ip address myAccessList
  set metric 5
  set local-preference 25
!
route-map myRouteMap permit 20
  set metric 2
!
access-list myAccessList permit 193.204.0.0/16
```