

kathara lab

bgp: prefix-filtering

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

- All the pages/slides in this presentation, including but not limited to, images, photos, animations, videos, sounds, music, and text (hereby referred to as “material”) are protected by copyright.
- This material, with the exception of some multimedia elements licensed by other organizations, is property of the authors and/or organizations appearing in the first slide.
- This material, or its parts, can be reproduced and used for didactical purposes within universities and schools, provided that this happens for non-profit purposes.
- Information contained in this material cannot be used within network design projects or other products of any kind.
- Any other use is prohibited, unless explicitly authorized by the authors on the basis of an explicit agreement.
- The authors assume no responsibility about this material and provide this material “as is”, with no implicit or explicit warranty about the correctness and completeness of its contents, which may be subject to changes.
- This copyright notice must always be redistributed together with the material, or its portions.

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 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

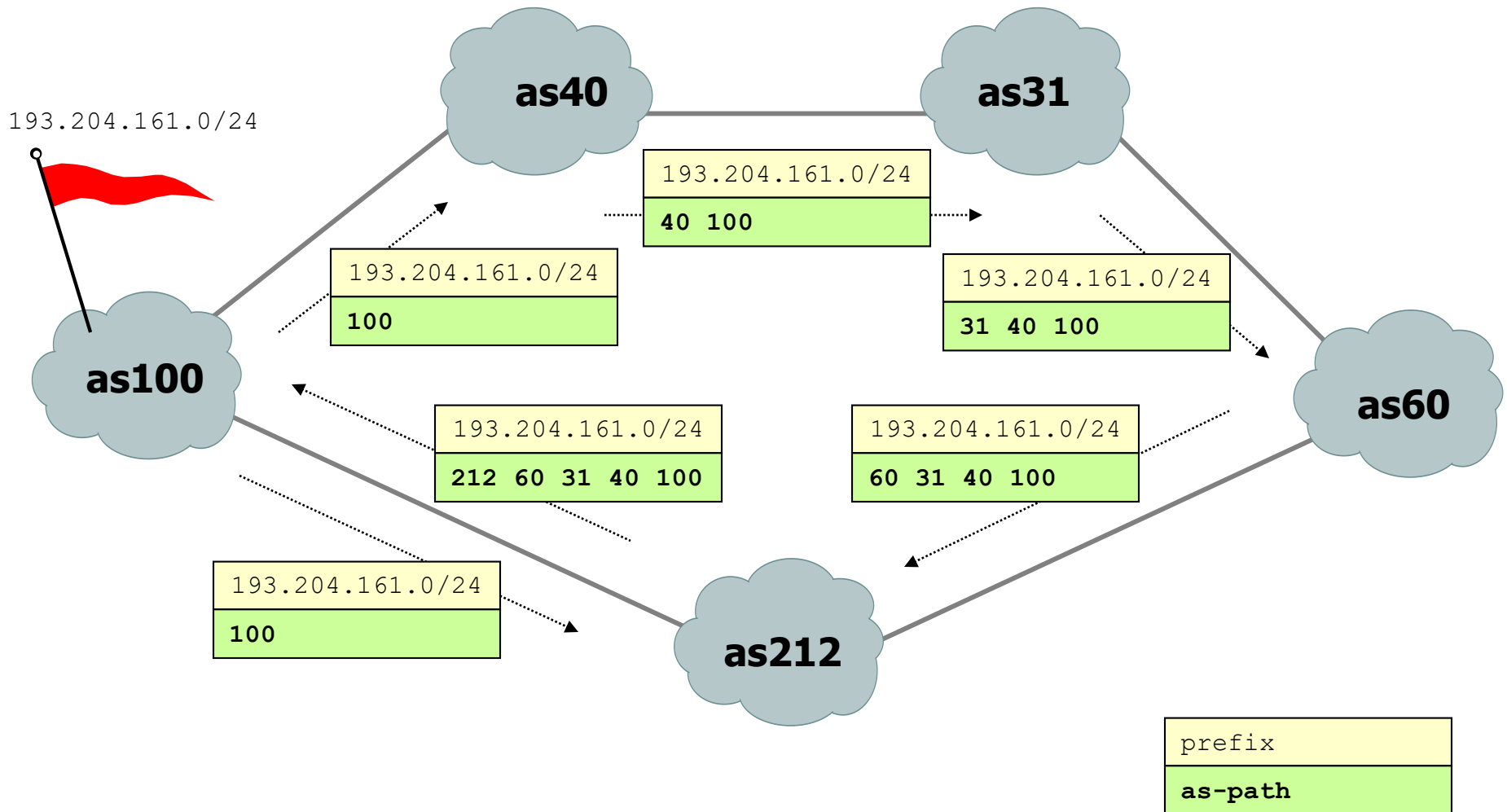
well-known attributes

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

attributes: prefix & as-path



attributes: as-path

- tells the sequence of ases that must be traversed in order to reach the destination prefix
 - also used to prevent loops
- empty for local routes
- does not change in ibgp

route selection

- for each prefix, routers choose one of the received announcements as the “best”
- the decision process is based on the values of bgp attributes and 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 as different prefixes
 - if the next-hop is not reachable via igp, the announcement cannot be selected

attributes: next-hop

- tells where to send packets for a specific ip network
- usually, the next-hop is the router that sends the announcements; exceptions:
 - “shared media” (ethernet, etc..)
 - ibgp announcements of networks learned using ebgp
 - internal routers perform a recursive lookup to understand how to reach the next-hop via igp

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
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
8. lowest igp metric
9. lowest bgp router-id

lowest
priority

announcement 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

zebra configuration file

```
router bgp 1
network 195.11.14.0/24
network 195.11.15.0/24
neighbor 193.10.11.2 remote-as 2
neighbor 193.10.11.2 description Router 2 of AS2
neighbor 193.10.11.2 prefix-list partialOut out
neighbor 193.10.11.2 prefix-list partialIn in
!
ip prefix-list partialOut permit 195.11.14.0/24
!
ip prefix-list partialIn deny 200.1.1.0/24
ip prefix-list partialIn 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:

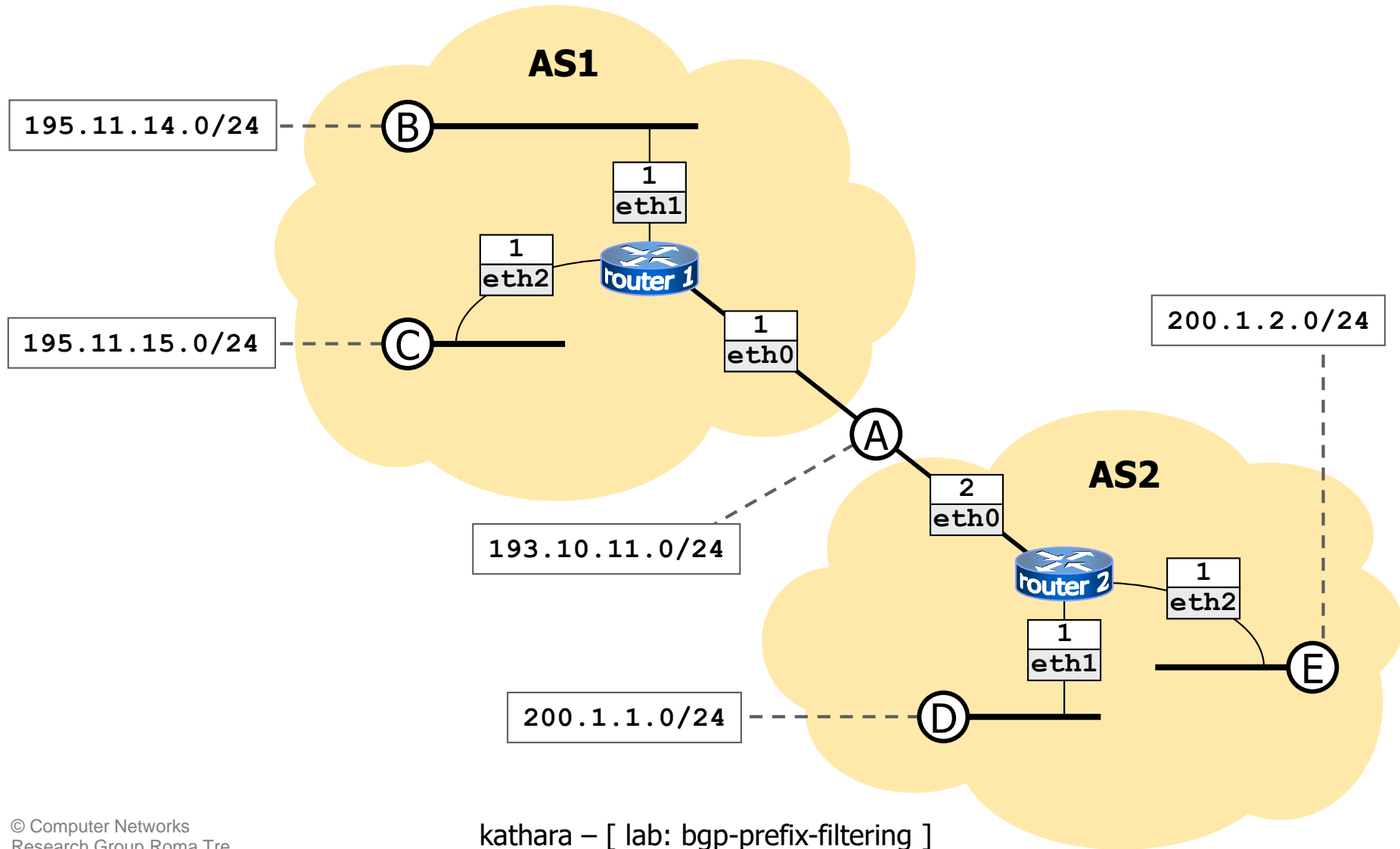
- `ip prefix-list letThru permit 10.0.0.0/8`
`ip prefix-list letThru deny any`
accepts 10.0.0.0/8 only
- `ip prefix-list throwAway deny any`
`ip prefix-list throwAway permit 10.0.0.0/8`
rejects everything

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 **denying anything**; 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
user@localhost:~/kathara-lab_bgp-prefix-filtering$ ./start
```

- check the bgpd configuration file

▼ router1

```
router1:~# less /etc/zebra/bgpd.conf
```

- check the bgpd log file

▼ router1

```
router1:~# less /var/log/zebra/bgpd.log
```

prefix filtering

■ check the routing table

```
router1
```

```
router1:~# route
Kernel IP routing table
Destination      Gateway          Genmask          Flags  Metric  Ref    Use  Iface
200.1.2.0        193.10.11.2     255.255.255.0    UG      0        0      0  eth0
193.10.11.0      *               255.255.255.0    U        0        0      0  eth0
195.11.14.0      *               255.255.255.0    U        0        0      0  eth1
195.11.15.0      *               255.255.255.0    U        0        0      0  eth2

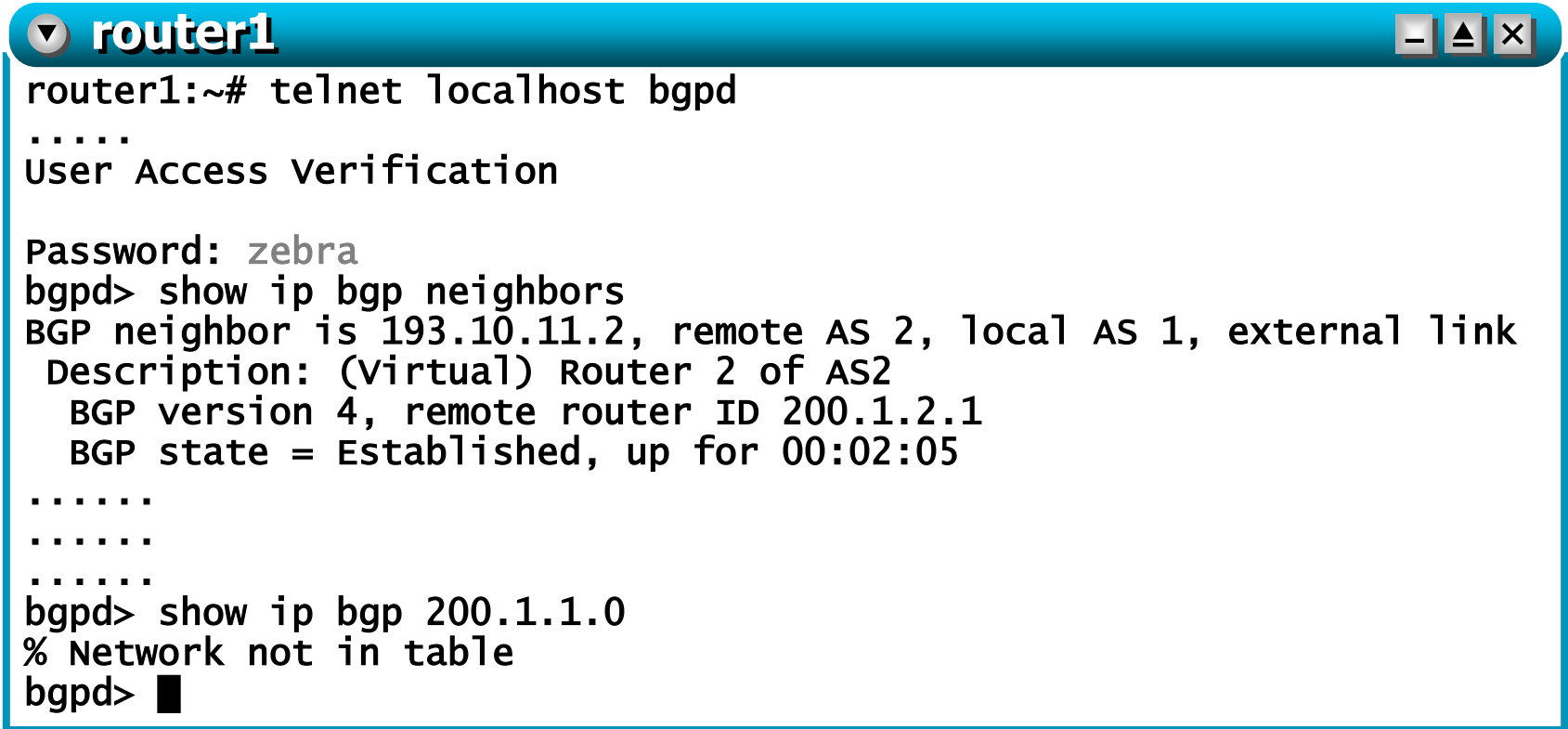
router1:~# telnet localhost zebra
Trying 127.0.0.1...
Connected to router1.
Escape character is '^]'.
.....
User Access Verification

Password: zebra
Router> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
      B - BGP, > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
C>* 193.10.11.0/24 is directly connected, eth0
C>* 195.11.14.0/24 is directly connected, eth1
C>* 195.11.15.0/24 is directly connected, eth2
B>* 200.1.2.0/24 [20/0] via 193.10.11.2, eth0, 00:11:20
Router> █
```

prefix filtering

- check the bgpd cli (command line interface)



```
router1:~# telnet localhost bgpd
.....
User Access Verification

Password: zebra
bgpd> show ip bgp neighbors
BGP neighbor is 193.10.11.2, remote AS 2, local AS 1, external link
  Description: (Virtual) Router 2 of AS2
    BGP version 4, remote router ID 200.1.2.1
    BGP state = Established, up for 00:02:05
.....
.....
.....
bgpd> show ip bgp 200.1.1.0
% Network not in table
bgpd> █
```

prefix filtering

- terminate the lab



A terminal window with a red title bar labeled "host machine". The terminal shows a command prompt "user@localhost:~/kathara-lab_bgp-prefix-filtering\$" followed by the command "1crash". A black cursor is visible at the end of the command.

```
user@localhost:~/kathara-lab_bgp-prefix-filtering$ 1crash
```


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

—zebra 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

—zebra 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
```