Connecting

- You only need a web browser
- The URL to connect to see above - you will be redirected to a router
- All routers run on the same server
- They run on different *ports* - port **9001** is router r01 and so on
- Connect to your router now

Entering commands

- Your router has two modes of operation
- **Terminal Mode** - this is the one you start with.
- The prompt in terminal mode is your router name followed by "#"
- Your router can *autocomplete* commands - just use the *tab* key (type twice for a list of possible completions).
- Or type "?" for a list of possible things you can enter

Lets try that now!

```
r01# sh?
```

Add a 2nd upstream: IPv4

- We already have a peer group for IPv4 upstreams: *upstream*
- We simply add a 2nd member to that group.

```
router bgp 645xx
 neighbor 10.230.x.1 remote-as 64496
 neighbor 10.230.x.1 peer-group upstream
```

Now check again for BGP summary and received prefixes.

Show commands:

- show bgp ipv4 summary
- show bgp ipv4
- show bgp ipv4 neighbor 10.200.x.1 routes
- show bgp ipv4 neighbor 10.200.x.1 received-routes
- show bgp ipv4 neighbor 10.230.x.1 routes
- show bgp ipv4 neighbor 10.230.x.1 received-routes

Add a 2nd upstream: IPv6

- Doing this for IPv6 is just as easy
- Simply add a 2nd member to the IPv6 peer-group: *upstream-v6*

```
router bgp 645xx
 neighbor 2001:db8:230:x::1 remote-as 64496
 neighbor 2001:db8:230:x::1 peer-group upstream-v6
```

Show commands:

- show bgp ipv6 summary
- show bgp ipv6
- show bgp ipv6 neighbor 2001:db8:200:x::1 routes
- show bgp ipv6 neighbor 2001:db8:200:x::1 received-routes
- show bgp ipv6 neighbor 2001:db8:230:x::1 routes
- show bgp ipv6 neighbor 2001:db8:230:x::1 received-routes

Add IPv4-Peering

- We need a new peer group for peering
- And new route-maps

```
route-map peering-in permit 100
route-map peering-out deny 100

router bgp 645xx
 neighbor peering peer-group
 address-family ipv4 unicast
  neighbor peering soft-reconfig inbound
  neighbor peering route-map peering-in in
  neighbor peering route-map peering-out out
  neighbor peering activate
 exit-address-family
 neighbor 80.81.192.1 remote-as 286
 neighbor 80.81.192.1 peer-group peering
```

Show commands:

- show bgp ipv4 summary
- show bgp ipv4
  - which prefixes are "best"? (marked with ">")
  - is it always peering?

Connect: http://go.de-cix.net/webinarlab

Add IPv6 Peering

- Again we need a *different* peergroup for IPv6 peering
- but we use the same route-maps

```
router bgp 645xx
 neighbor peering-v6 peer-group
 address-family ipv6 unicast
  neighbor peering-v6 soft-reconfig inbound
  neighbor peering-v6 route-map peering-in in
  neighbor peering-v6 route-map peering-out out
  neighbor peering-v6 activate
 exit-address-family
 neighbor 2001:7f8::0:1 remote-as 286
 neighbor 2001:7f8::0:1 peer-group peering-v6
```

Show commands:

- show bgp ipv6 summary
- show bgp ipv6

Which prefixes are “best”? Why?

Setting local preference

- We want to prefer peering over upstream
- For this we adjust *Local Preference*
- *Local Preference* is a configured BGP attribute
- Higher value == better

```
route-map peering-in permit 100
 set local-preference 1000

route-map upstream-in permit 100
 set local-preference 10
```

Show commands:

- show bgp ipv4
- show bgp ipv6

Peer with each other - IPv4

- We can now setup peering between all routers
- Just add them to the *peering* peer-group (for IPv4)

```
router bgp 645xx
 neighbor 80.81.192.1yy remote-as 645yy
 neighbor 80.81.192.1yy peer-group peering
 neighbor 80.81.192.1zz remote-as 645zz
 neighbor 80.81.192.1zz peer-group peering
 ...
```

…with yy, zz = 01, 02, 03, …

Show commands:

- show bgp ipv4 summary
- show bgp ipv4

Peer with each other - IPv6

- We can now setup peering between all routers
- For IPv6 add them to the *peering-v6* peer-group

```
router bgp 645xx
 neighbor 2001:7f8::yhex:0:1 remote-as 645yy
 neighbor 2001:7f8::yhex:0:1 peer-group peering-v6
 neighbor 2001:7f8::zhex:0:1 remote-as 645zz
 neighbor 2001:7f8::zhex:0:1 peer-group peering-v6
 ...
```

For IPv6 you need to convert your neighbors AS number into hexadecimal (yhex, zhex):

64501 = fbf5	64504 = fbf8	64507 = fbfb	64510 = fbfe
64502 = fbf6	64505 = fbf9	64508 = fbfc	64511 = fbff
64503 = fbf7	64506 = fbfa	64509 = fbfd	64512 = fc00

Show commands:

- show bgp ipv6 summary
- show bgp ipv6

Inject a prefix into BGP

- For this we must get it into the routing table
- We use a static route for this
- Then must get the prefix into BGP
- For this we use a *network* statement

For IPv4:

```
ip route 192.168.n.0/24 dummy0
router bgp 645xx
 address-family ipv4 unicast
  network 192.168.n.0/24
```

For IPv6:

```
ipv6 route 2001:7f8:99:xx::/48  dummy0
router bgp 645xx
 address-family ipv6 unicast
  network 2001:7f8:99:xx::/48
```

Show commands:

- show bgp ipv4 192.168.n.0/24
- show bgp ipv6 2001:7f8:99:xx::/48

Announcing our prefixes

- We have the prefixes now in BGP but they are not announced
- We have a route-map in place preventing this
- So we must change it and allow *only our* prefixes out.

```
ip prefix-list my-prefixes permit 192.168.n.0/24
ipv6 prefix-list my-prefixes permit 2001:7f8:99:xx::/48

route-map peering-out permit 150
 match ip address prefix-list my-prefixes

route-map peering-out permit 160
 match ipv6 address prefix-list my-prefixes

no route-map peering-out deny 100

route-map upstream-out permit 150
 match ip address prefix-list my-prefixes

route-map upstream-out permit 160
 match ipv6 address prefix-list my-prefixes

no route-map upstream-out deny 100
```

Show commands:

- show bgp ipv4
  (to see what others are announcing)
- show bgp ipv4 neighbor 80.81.192.101 advertised
  (to see what yourself are annoucing)

Experiment 2b - Become multihomed

DE-CIX Academy

Version 2.0w

1 Introduction

In this experiment we add a 2nd upstream provider and also a peering connection to our BGP setup and monitor how BGP attributes are received and how our router selects a best path to destination prefixes.

2 Network Setup

2.1 IPv4

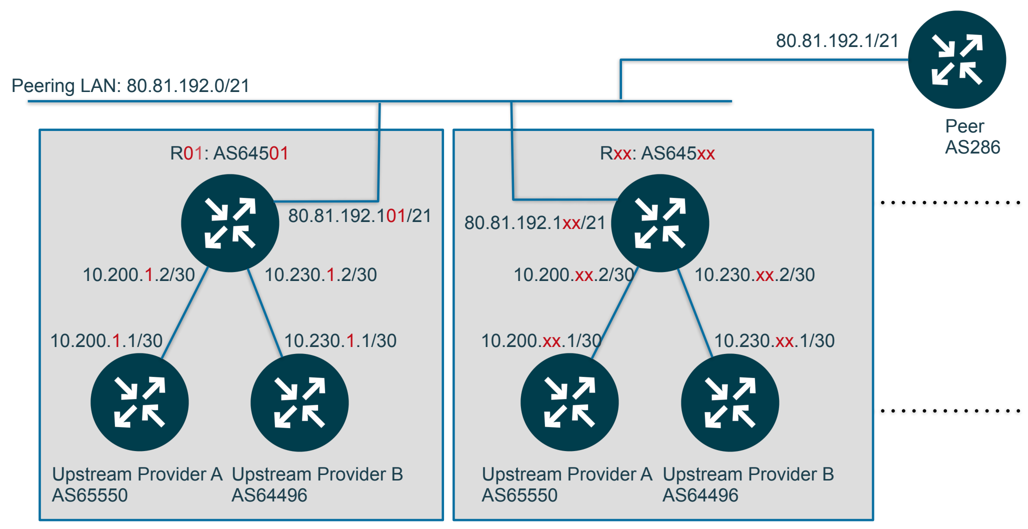

Figure 1: Network Setup for IPv4

Figure 1 shows the network topology and IPv4 addressing for this experiment. Participants are using AS645xx and are connected to AS65550 and AS64496 as upstreams and to a peering LAN.

2.2 IPv6

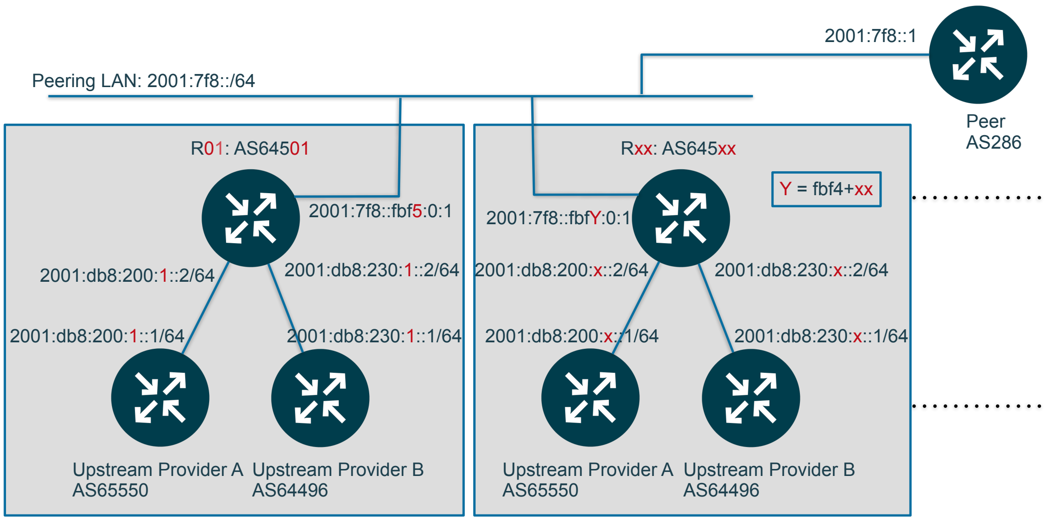

Figure 2: Network Setup for IPv6

Figure 2 shows the network topology and IPv6 addressing for this experiment. Participants are using AS645xx and are connected to AS65550 and AS64496 as upstreams and to a peering LAN. Note that IPv6 addressing in the peering LAN is derived from the AS number - AS64501 converts to fbf5 as part of the IPv6 address (for the fixed peer AS286 this is not possible because of Docker).

3 Setup eBGP

3.1 To set up eBGP you need to:

- configure four peer-groups:
  - two peer-groups for upstreams (v4 and v6)
  - two peer-groups for peering (v4 and v6)
- configure (empty) filters for both (peering-in, peering-out, upstream-in, upstream-out)
- configure neighbors as peer-group members

Information you need:

- Your AS number is 645xx (with xx from 01 to 15)
- Your upstream providers AS numbers are 65550 (with IPv4 address network 10.200.x.1) and 64496 (with 10.230.x.1)

- For peering we are using IP addresses from DE-CIX Frankfurt. Your peer is AS286 with IP 80.81.191.1, your IP address is 80.81.192.1xx/21 (with *xx* being the same as your router number).

For IPv6:

- The AS numbers of your upstream providers are the same as in ipv4
- AS64496 is at 2001:db8:230:xx::1
- AS65550 is at 2001:db8:200:xx::1
- your peer AS286 is at 2001:7f8::1
- your own IPv6 address for peering is 2001:7f8::<*ashex*>:0:1, with <*ashex*> being your AS number in hexadecimal.

3.2 To configure eBGP in config mode:

neighbor <name> peer-group starts a peer group named <*name*>

neighbor <name> route-map <rmname> in/out filters all prefixes in or out through route-map <*rmname*>. If the route-map is not defined (or if there is a typo in the name) it blocks everyhting in or out.

neighbor <name> soft-reconfiguration inbound allows you to see (via *show bgp ipv4/ipv6 neighbor <address> received-routes*) what the router receives before the incoming route-map is processed.

neighbor <name> send-community both tells BGP to send communities, needed later.

neighbor <ip-address> remote-as <asnumber> sets up an neighbor with IP <*ip-address*> and AS <*asnumber*>. This command becomes active as soon as it is entered, so if you want to enter more config items for this peer, you either be quick or you need to shut it down until config is complete.

neighbor <ip-address> shutdown shuts down the BGP connection to this neibbor. Useful when you want to take some time configuring.

The remote AS is not configured in the peer-group but in the peer-group member, as you might want to re-use the same peer-group for more then one upstream provider.

3.3 Commands to check your eBGP session:

show bgp ipv4/ipv6 summary give you information about your peers

show bgp ipv4/ipv6 neighbor <ip-address> gives you detailed information about BGP neighbor with IP <*ip-address*>. If you ommit the IP lists all neigbors very detailed (long output).

show bgp ipv4/ipv6 lists all prefixes in BGP. Be aware that once you are on a real router connected to the real internet the output can be very longer

show bgp ipv4/ipv6 <prefix> gives you detailed BGP information about IP <*prefix*>

show bgp ipv4/ipv6 neighbor <ip-address> routes shows all valid routes received from a specific neighbor (after the filters).

show bgp ipv4/ipv6 neighbor <ip-address> received-routes shows all routes received from a specific neighbor (before filters are applied). Useful for checking if your filters work.

show bgp ipv4/ipv6 neighbor <ip-address> advertised-routes shows all prefixes advertised to a specific neighbor.

3.4 Tasks:

- Define two peer groups for eBGP upstream neighbors (if not already done) - one for IPv4 and one for IPv6
- Set up an eBGP session to AS65550 using IPv4
- Set up an eBGP session to AS64496 using IPv4
- Check if the sessions come up
- Look if you are receiving anything
- define (empty) route-maps for in and out for upstream
- Check again for received prefixes
- Define a peer group for eBGP peering
- Set up an eBGP session to AS286
- define (empty) route-maps for in and out for peering
- Try all commands listed above and look for anything unusual.
- do the same for IPv6

Answer the following questions:

- Are there any prefixes sent which your router does not accept?
- Why?
- Which routes to a given prefix are "best"?
- What criteria is used to make a prefix "best"?

4 Setup peering with all other participants

4.1 To set up peering you need to:

- create peer-groups *peering* and *peering-v6*(if you have not already done so)
- configure neighbors to all other participants using these peer-groups.

4.2 Information you need:

- AS numbers are 645xx (with xx from 01 to 20)
- IPv4 addresses are 80.81.192.1xx (with xx same as in AS)
- IPv6 addresses are 2001:7f8::YYYY:0:1 - with YYYY being the AS number. (AS64501 == fbf5).

5 Optimize Routing

5.1 Routing Policy

We want to implement the following very simple routing policy in this experiment:

- Prefer peering over upstream

5.2 Config statements you need

route-map <name> permit/deny <number> starts a route-map statement for a named route-map. Each route-map is referenced by *<name>* and statements are processed in order, starting with the lowest *<number>*. Inside each statement there can be zero to multiple *match* and *set* clauses. If the statement is evaluated true (all match conditions are true) permit or deny is executed. If not (matches evaluate false) the next statement in order is processed. If there are no statements left, there is an implicit deny at the end.

match <condition> An empty/missing *match* is always true. If there are multiples *match* statements, they all must be true for the entry to be true.

set <parameter> <value> If all matches are true (or if there are no match statements), all *set* statements are executed, the route-map is terminated and either permit or deny is executed (depending on the value in the statement header).

set local-preference *value* this sets the local preference of prefixes to value. It is only executed if there is either no *match* statement or if *all* match statements evaluate true.

5.3 Implementation

We are using *local preference* to influence the routers best prefix selection. Local preference is the very first decision parameter in the best path selection algorithm.

Where to set *local preference*? This is what route maps are for. We add statements to the already existing (but empty) incoming route maps.

5.4 Tasks

- Set the local preference of all prefixes received via peering to 1000
- Set local preference of all prefixes receive via upstream to 10

6 Solutions

6.1 Configure two IPv4 upstreams

```
router bgp 645xx
 bgp ebgp-requires-policy
 no bgp default ipv4-unicast
 neighbor upstream peer-group
 neighbor 10.200.xx.1 remote-as 65550
 neighbor 10.200.xx.1 peer-group upstream
 neighbor 10.230.xx.1 remote-as 64496
 neighbor 10.230.xx.1 peer-group upstream
 address-family ipv4 unicast
  neighbor upstream soft-reconfiguration inbound
  neighbor upstream route-map upstream-in in
  neighbor upstream route-map upstream-out out
  neighbor upstream activate
 exit-address-family
!
route-map upstream-out deny 100
!
route-map upstream-in permit 100
!
```

6.2 Configure two IPv6 upstreams

```
router bgp 645xx
 bgp ebgp-requires-policy
 no bgp default ipv4-unicast
 neighbor upstream-v6 peer-group
 neighbor 2001:db8:200:xx::1 remote-as 65550
 neighbor 2001:db8:200:xx::1 peer-group upstream-v6
 neighbor 2001:db8:230:xx::1 remote-as 64496
 neighbor 2001:db8:230:xx::1 peer-group upstream-v6
 address-family ipv6 unicast
  neighbor upstream-v6 activate
  neighbor upstream-v6 soft-reconfiguration inbound
  neighbor upstream-v6 route-map upstream-in in
  neighbor upstream-v6 route-map upstream-out out
 exit-address-family
 !
 route-map upstream-out deny 100
 !
 route-map upstream-in permit 100
 !
```

6.3 Configure Peering (IPv4 and IPv6)

```
router bgp 645xx
 bgp ebgp-requires-policy
 no bgp default ipv4-unicast
 neighbor peering peer-group
 neighbor peering-v6 peer-group
 neighbor 80.81.192.1 remote-as 286
 neighbor 80.81.192.1 peer-group peering
 neighbor 80.81.192.1yy remote-as 645yy
 neighbor 80.81.192.1yy peer-group peering
 ...
 neighbor 2001:7f8::1 remote-as 286
 neighbor 2001:7f8::1 peer-group peering-v6
 neighbor 2001:7f8::fbf5:0:1 remote-as 64501
 neighbor 2001:7f8::fbf5:0:1 peer-group peering-v6
 neighbor 2001:7f8::fbf6:0:1 remote-as 64502
 neighbor 2001:7f8::fbf6:0:1 peer-group peering-v6
 ...
 address-family ipv4 unicast
  neighbor peering activate
  neighbor peering soft-reconfiguration inbound
  neighbor peering route-map peering-in in
  neighbor peering route-map peering-out out
 exit-address-family
 address-family ipv6 unicast
  neighbor peering-v6 activate
  neighbor peering-v6 soft-reconfiguration inbound
  neighbor peering-v6 route-map peering-in in
  neighbor peering-v6 route-map peering-out out
 exit-address-family
!
route-map peering-in permit 100
!
route-map peering-out deny 100
!
```

6.4 Announce an IPv4 and IPv6 prefix

Be careful - this does not work on Cisco routers. While FRRouting needs separate route-map statements for IPv4 and IPv6, Cisco wants both *match* clauses in one route-map statement.

```
ip route 192.168.nn.0/24 dummy0
ipv6 route 2001:db7:nnnn::/48 dummy0
!
router bgp 645xx
 address-family ipv4 unicast
  network 192.168.nn.0/24
 address-family ipv6 unicast
  network 2001:db7:nnnn::/48
!
ip prefix-list my-prefixes permit 192.168.nn.0/24
!
ipv6 prefix-list my-prefixes permit 2001:db7:nnnn::/48
!
route-map peering-out permit 50
 match ip address prefix-list my-prefixes
```

```
!
route-map peering-out permit 60
 match ipv6 address prefix-list my-prefixes
!
route-map upstream-out permit 50
 match ip address prefix-list my-prefixes
!
route-map upstream-out permit 60
 match ipv6 address prefix-list my-prefixes
```

6.5 Prefer peering over upstream

We set the local preference in the incoming route map:

```
  route-map upstream-in permit 100
   set local-preference 10
  !
  route-map peering-in permit 100
   set local-preference 1000
```