Communication Networks Routing Project Report, Group 86

Alkinoos Sarioglou, Johannes Schading, Fabian Stuber

May 6, 2021

1 Q1 Intra-Domain Routing

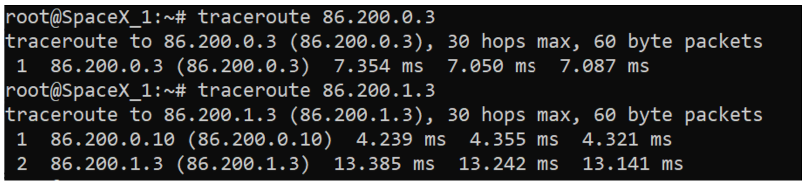
1.1 Q1.1 (Fabian)

The following structure was chosen for the IP Address allocation: The whole datacenter 1 is in the subnet 86.200.0.0/23. To prevent the NASA and SpaceX hosts to communicate on L2, the prefix length for the hosts is chosen 24 bits. All SpaceX hosts or more precisely all hosts in the VLAN10 are in the 86.200.0.0/24 subnet and the NASA hosts (VLAN20) in the 86.200.1.0/24 subnet. With this it is guaranteed that all IP addresses are in the 86.200.0.0/23 subnet but there is no L2 connectivity between the companies. For the New York router host, the last byte is chosen to be 10 and for the Brooklyn router 20. And for the other hosts the last bit is equal to its index. Some examples are shown in table 1. Therefore a traceroute from SpaceX_1 to SpaceX_3 contains

Host	IP-Address
New York VLAN 10	86.200.0.10
Brooklyn VLAN 20	86.200.1.20
SpaceX_1	86.200.0.1
NASA_3	86.200.1.3

Table 1: Example IP Addresses

only one hop as they are connected directly over switches and they are all in the same subnet. A traceroute from SpaceX_1 to NASA_3 contains two hops as the connection is established in layer 3. The SpaceX host realizes that the NASA host is not in the same subnet, therefore it directs the packets to its default gateway on the New York router (86.200.0.10). This router sees the NASA hosts, therefore it directs the packet directly to the corresponding destination.

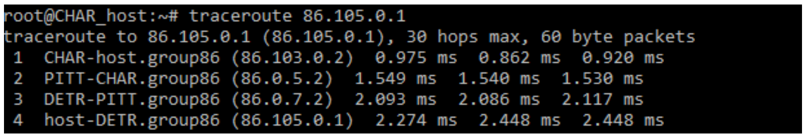


```
root@SpaceX_1:~# traceroute 86.200.0.3
traceroute to 86.200.0.3 (86.200.0.3), 30 hops max, 60 byte packets
 1  86.200.0.3 (86.200.0.3)  7.354 ms  7.050 ms  7.087 ms
root@SpaceX_1:~# traceroute 86.200.1.3
traceroute to 86.200.1.3 (86.200.1.3), 30 hops max, 60 byte packets
 1  86.200.0.10 (86.200.0.10)  4.239 ms  4.355 ms  4.321 ms
 2  86.200.1.3 (86.200.1.3)  13.385 ms  13.242 ms  13.141 ms
```

Figure 1: Traceroute from SpaceX_1 to SpaceX_3 and NASA_3

1.2 Q1.2 (Johannes)

We have configured the IP addresses in our network appropriate to the suggested L3 topology on GitLab. The *show ip route ospf* command shows that all the routers and necessary hosts are connected. The following Figure 2 shows the IPv4 traceroute from the CHAR host (86.103.0.1) to DETR host (86.105.0.1).



```
root@CHAR_host:~# traceroute 86.105.0.1
traceroute to 86.105.0.1 (86.105.0.1), 30 hops max, 60 byte packets
 1  CHAR-host.group86 (86.103.0.2)  0.975 ms  0.862 ms  0.920 ms
 2  PITT-CHAR.group86 (86.0.5.2)  1.549 ms  1.540 ms  1.530 ms
 3  DETR-PITT.group86 (86.0.7.2)  2.093 ms  2.086 ms  2.117 ms
 4  host-DETR.group86 (86.105.0.1)  2.274 ms  2.448 ms  2.448 ms
```

Figure 2: Traceroute from CHAR host to DETR host

1.3 Q1.3 (Johannes)

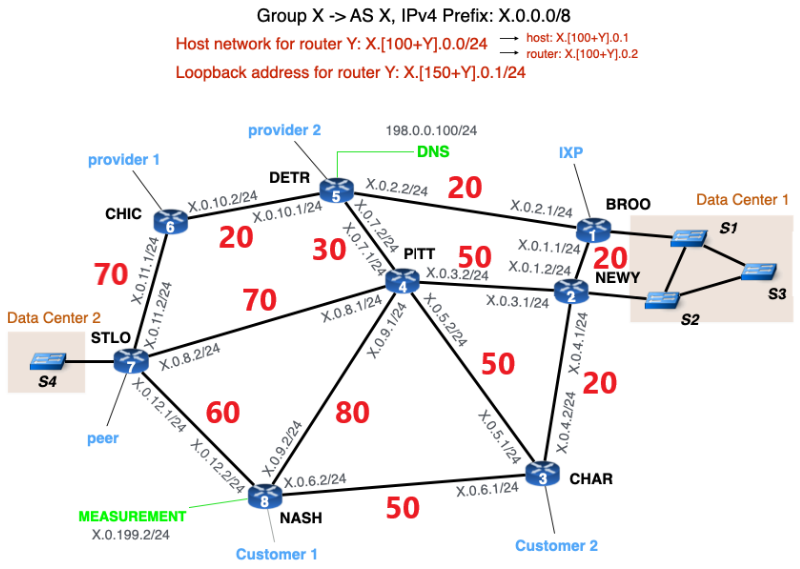
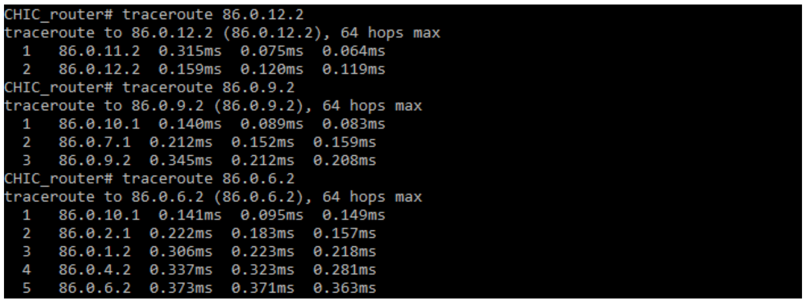


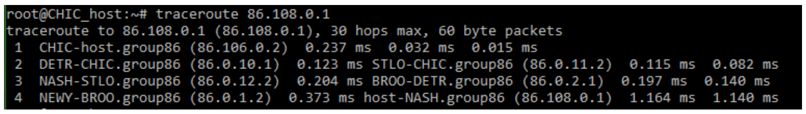
Figure 3: Assigned weights of the links for intra domain routing

Figure 3 shows our assigned essential weights of the links for intra domain routing. They are chosen such that the CHIC router sends packets to the NASH router as desired, which is verified by the traceroutes to all the interfaces of the NASH router shown in Figure 4. There is also the required traceroute in Figure 5 from CHIC host (86.106.0.1) to NASH host (86.108.0.1).



```
CHIC_router# traceroute 86.0.12.2
traceroute to 86.0.12.2 (86.0.12.2), 64 hops max
  1   86.0.11.2  0.315ms  0.075ms  0.064ms
  2   86.0.12.2  0.159ms  0.120ms  0.119ms
CHIC_router# traceroute 86.0.9.2
traceroute to 86.0.9.2 (86.0.9.2), 64 hops max
  1   86.0.10.1  0.140ms  0.089ms  0.083ms
  2   86.0.7.1  0.212ms  0.152ms  0.159ms
  3   86.0.9.2  0.345ms  0.212ms  0.208ms
CHIC_router# traceroute 86.0.6.2
traceroute to 86.0.6.2 (86.0.6.2), 64 hops max
  1   86.0.10.1  0.141ms  0.095ms  0.149ms
  2   86.0.2.1  0.222ms  0.183ms  0.157ms
  3   86.0.1.2  0.306ms  0.223ms  0.218ms
  4   86.0.4.2  0.337ms  0.323ms  0.281ms
  5   86.0.6.2  0.373ms  0.371ms  0.363ms
```

Figure 4: Different traceroutes from CHIC router to NASH router

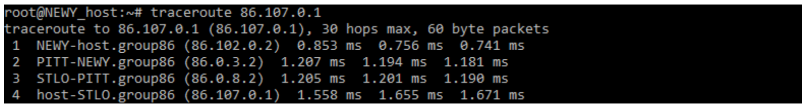


```
root@CHIC_host:~# traceroute 86.108.0.1
traceroute to 86.108.0.1 (86.108.0.1), 30 hops max, 60 byte packets
 1  CHIC-host.group86 (86.106.0.2)  0.237 ms  0.032 ms  0.015 ms
 2  DETR-CHIC.group86 (86.0.10.1)  0.123 ms STLO-CHIC.group86 (86.0.11.2)  0.115 ms  0.082 ms
 3  NASH-STLO.group86 (86.0.12.2)  0.204 ms BROO-DETR.group86 (86.0.2.1)  0.197 ms  0.140 ms
 4  NEWY-BROO.group86 (86.0.1.2)  0.373 ms host-NASH.group86 (86.108.0.1)  1.164 ms  1.140 ms
```

Figure 5: Traceroute from CHIC host to NASH host

Moreover, the PITT router forwards to STLO or NEWY without detour in order to provide the direct paths for the communication between the Data Center 1 and Data Center 2. This is verified by the traceroute in Figure

6 from the NEWY host (86.102.0.1) to the STLO host (86.107.0.1).



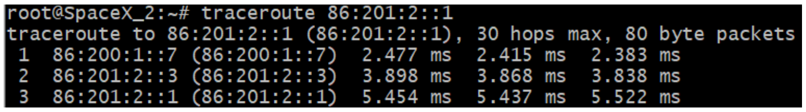
```
root@NEWY_host:~# traceroute 86.107.0.1
traceroute to 86.107.0.1 (86.107.0.1), 30 hops max, 60 byte packets
 1  NEWY-host.group86 (86.102.0.2)  0.853 ms  0.756 ms  0.741 ms
 2  PITT-NEWY.group86 (86.0.3.2)  1.207 ms  1.194 ms  1.181 ms
 3  STLO-PITT.group86 (86.0.8.2)  1.205 ms  1.201 ms  1.190 ms
 4  host-STLO.group86 (86.107.0.1)  1.558 ms  1.655 ms  1.671 ms
```

Figure 6: Traceroute from NEWY host to STLO host

The result of the traceroutes corresponds to what we expected. This is due to the fact that we chose the weights of the path STLO-PITT-NEWY and some links connected to PITT to be large enough such that they are too expensive for routes from CHIC to NASH to be used and the alternative paths are cheap enough. So every path comprising at least a part of the path STLO-PITT-NEWY is more expensive than at least one alternative path avoiding it. But the links are still available in general, hence, they function only as a backup path.

1.4 Q1.4 (Alkinoos)

Figure 7 shows the IPv6 traceroute from the SpaceX_2 host (86:200:1::2/48) to the NASA_4 host (86:201:2::1/48). The host of SpaceX_2 (86:200:1::2/48) is trying to communicate with an IP address outside of its subnet (86:201:2::1/48

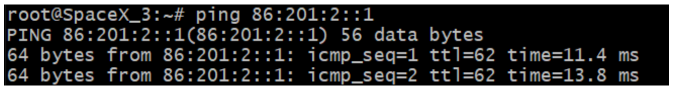


```
root@SpaceX_2:~# traceroute 86:201:2::1
traceroute to 86:201:2::1 (86:201:2::1), 30 hops max, 80 byte packets
 1  86:200:1::7 (86:200:1::7)  2.477 ms  2.415 ms  2.383 ms
 2  86:201:2::3 (86:201:2::3)  3.898 ms  3.868 ms  3.838 ms
 3  86:201:2::1 (86:201:2::1)  5.454 ms  5.437 ms  5.522 ms
```

Figure 7: Traceroute from SpaceX_2 to NASA_4

of NASA_4). Therefore, it will use the default gateway to reach the destination. The default gateway is the NEWY router for VLAN10 corresponding to the VLAN of SpaceX. As it can be seen from the output of the traceroute, the packet reaches this interface (86:200:1::7/48) and then it is forwarded through the network to the STLO router. The STLO router receives the IPv4 packet and removes the IPv4 header. Then, it sends the packet in the appropriate interface which corresponds to VLAN20, which is the VLAN of NASA. From the traceroute, it can be observed that the packet reaches this interface (86:201:2::3/48) and then it is forwarded to the NASA_4 host (86:201:2::1/48) through the switch S4.

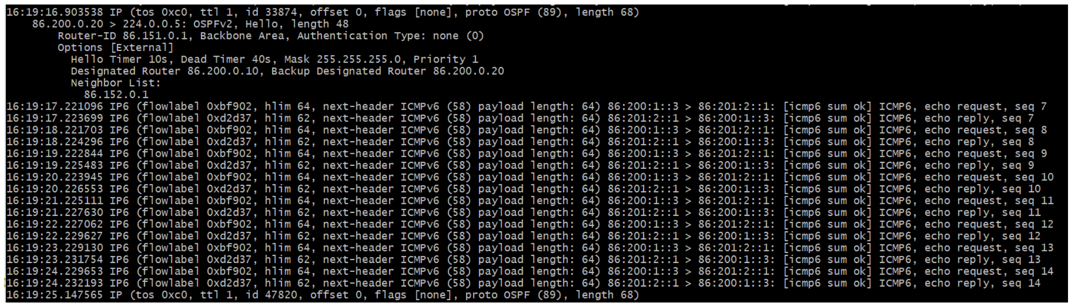
Following this, a ping from SpaceX_3 to NASA_4 is run. The output of the ping command from SpaceX_3 (86:200:1::3/48) to the IPv6 address of NASA_4 (86:201:2::1/48) is shown in Figure 8.



```
root@SpaceX_3:~# ping 86:201:2::1
PING 86:201:2::1(86:201:2::1) 56 data bytes
64 bytes from 86:201:2::1: icmp_seq=1 ttl=62 time=11.4 ms
64 bytes from 86:201:2::1: icmp_seq=2 ttl=62 time=13.8 ms
```

Figure 8: Ping from SpaceX_3 to NASA_4

Then, the output is captured from the interface of VLAN10 (corresponding to the VLAN of SpaceX) on the NEWY router by executing the *tcpdump -i NEWY-L2.10 -n -v* command. The result can be seen in Figure 9.



```
16:19:16.903538 IP (tos 0xc0, ttl 1, id 33874, offset 0, flags [none], proto OSPF (89), length 68)
    86.200.0.20 > 224.0.0.5: OSPFv2, Hello, length 48
        Router-ID 86.151.0.1, Backbone Area, Authentication Type: none (0)
        Options [External]
          Hello Timer 10s, Dead Timer 40s, Mask 255.255.255.0, Priority 1
          Designated Router 86.200.0.10, Backup Designated Router 86.200.0.20
          Neighbor List:
            86.152.0.1
16:19:17.221096 IP6 (flowlabel 0xbf902, hlim 64, next-header ICMPv6 (58) payload length: 64) 86:200:1::3 > 86:201:2::1: [icmp6 sum ok] ICMP6, echo request, seq 7
16:19:17.223699 IP6 (flowlabel 0xd2d37, hlim 62, next-header ICMPv6 (58) payload length: 64) 86:201:2::1 > 86:200:1::3: [icmp6 sum ok] ICMP6, echo reply, seq 7
16:19:18.221703 IP6 (flowlabel 0xbf902, hlim 64, next-header ICMPv6 (58) payload length: 64) 86:200:1::3 > 86:201:2::1: [icmp6 sum ok] ICMP6, echo request, seq 8
16:19:18.224296 IP6 (flowlabel 0xd2d37, hlim 62, next-header ICMPv6 (58) payload length: 64) 86:201:2::1 > 86:200:1::3: [icmp6 sum ok] ICMP6, echo reply, seq 8
16:19:19.222844 IP6 (flowlabel 0xbf902, hlim 64, next-header ICMPv6 (58) payload length: 64) 86:200:1::3 > 86:201:2::1: [icmp6 sum ok] ICMP6, echo request, seq 9
16:19:19.225483 IP6 (flowlabel 0xd2d37, hlim 62, next-header ICMPv6 (58) payload length: 64) 86:201:2::1 > 86:200:1::3: [icmp6 sum ok] ICMP6, echo reply, seq 9
16:19:20.223945 IP6 (flowlabel 0xbf902, hlim 64, next-header ICMPv6 (58) payload length: 64) 86:200:1::3 > 86:201:2::1: [icmp6 sum ok] ICMP6, echo request, seq 10
16:19:20.226553 IP6 (flowlabel 0xd2d37, hlim 62, next-header ICMPv6 (58) payload length: 64) 86:201:2::1 > 86:200:1::3: [icmp6 sum ok] ICMP6, echo reply, seq 10
16:19:21.225111 IP6 (flowlabel 0xbf902, hlim 64, next-header ICMPv6 (58) payload length: 64) 86:200:1::3 > 86:201:2::1: [icmp6 sum ok] ICMP6, echo request, seq 11
16:19:21.227630 IP6 (flowlabel 0xd2d37, hlim 62, next-header ICMPv6 (58) payload length: 64) 86:201:2::1 > 86:200:1::3: [icmp6 sum ok] ICMP6, echo reply, seq 11
16:19:22.227062 IP6 (flowlabel 0xbf902, hlim 64, next-header ICMPv6 (58) payload length: 64) 86:200:1::3 > 86:201:2::1: [icmp6 sum ok] ICMP6, echo request, seq 12
16:19:22.229627 IP6 (flowlabel 0xd2d37, hlim 62, next-header ICMPv6 (58) payload length: 64) 86:201:2::1 > 86:200:1::3: [icmp6 sum ok] ICMP6, echo reply, seq 12
16:19:23.229130 IP6 (flowlabel 0xbf902, hlim 64, next-header ICMPv6 (58) payload length: 64) 86:200:1::3 > 86:201:2::1: [icmp6 sum ok] ICMP6, echo request, seq 13
16:19:23.231754 IP6 (flowlabel 0xd2d37, hlim 62, next-header ICMPv6 (58) payload length: 64) 86:201:2::1 > 86:200:1::3: [icmp6 sum ok] ICMP6, echo reply, seq 13
16:19:24.229653 IP6 (flowlabel 0xbf902, hlim 64, next-header ICMPv6 (58) payload length: 64) 86:200:1::3 > 86:201:2::1: [icmp6 sum ok] ICMP6, echo request, seq 14
16:19:24.232193 IP6 (flowlabel 0xd2d37, hlim 62, next-header ICMPv6 (58) payload length: 64) 86:201:2::1 > 86:200:1::3: [icmp6 sum ok] ICMP6, echo reply, seq 14
16:19:25.147565 IP (tos 0xc0, ttl 1, id 47820, offset 0, flags [none], proto OSPF (89), length 68)
```

Figure 9: tcpdump output from the NEWY-L2.10 interface of NEWY router

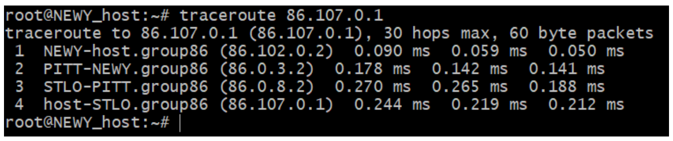
From the output of the tcpdump it is obvious that the SpaceX_3 host (86:200:1::3/48) performs TCP echo requests to the NASA_4 host (86:201:2::1/48) and the NASA_4 host replies back for all the sequences of pings. Therefore, it is demonstrated that the 6in4 tunnel from NEWY to STLO functions correctly.

1.5 Q1.5 (Alkinoos)

In order to instruct the NEWY and STLO routers to send all the traffic between them through the PITT router only, static routing needs to be performed.
First of all, the NEWY router was configured manually to send all the traffic directed to the loopback address of the STLO router (86.157.0.1/24) through the interface of the PITT router (86.0.3.2). This was achieved by running the command: *ip route 86.157.0.1/24 86.0.3.2*. Following this the STLO router was also configured statically to send all the traffic directed to the loopback address of the NEWY router (86.152.0.1/24) through its interface with the PITT router (86.0.8.1). The above was accomplished by executing the following command: *ip route 86.152.0.1/24 86.0.8.1*.

Configuring the routers statically can be risky in many situations, because static routing is not failure tolerant. For instance, if the link between either NEWY-PITT or PITT-STLO fails, then the traffic cannot be redirected to a different route in order to reach the destination. In that case, the hosts from DC1 and DC2 would not be able to communicate at all, which is worse than communicating but at a slower rate through another route. The hosts would have to wait for the broken link to be repaired or for the reconfiguration of the routers NEWY and STLO, before they can communicate again.

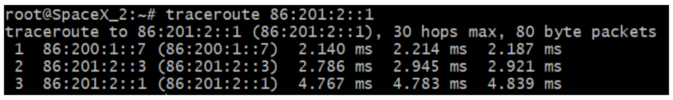
Then, the success of the static routing is tested by running a traceroute from the NEWY-host (86.102.0.1/24) to the STLO-host (86.107.0.1/24). The result of the traceroute command is seen on Figure 10. It can be seen that the traffic between the hosts is indeed directed through the PITT router.



```
root@NEWY_host:~# traceroute 86.107.0.1
traceroute to 86.107.0.1 (86.107.0.1), 30 hops max, 60 byte packets
 1  NEWY-host.group86 (86.102.0.2)  0.090 ms  0.059 ms  0.050 ms
 2  PITT-NEWY.group86 (86.0.3.2)  0.178 ms  0.142 ms  0.141 ms
 3  STLO-PITT.group86 (86.0.8.2)  0.270 ms  0.265 ms  0.188 ms
 4  host-STLO.group86 (86.107.0.1)  0.244 ms  0.219 ms  0.212 ms
root@NEWY_host:~# |
```

Figure 10: Traceroute output from the NEWY-host to the STLO-host

Following this, a traceroute between the SpaceX_3 host of DC1 and the NASA_4 host of DC2 is executed and the result can be observed on Figure 11. It is obvious that the communication between the hosts was successful.



```
root@SpaceX_2:~# traceroute 86:201:2::1
traceroute to 86:201:2::1 (86:201:2::1), 30 hops max, 80 byte packets
 1  86:200:1::7 (86:200:1::7)  2.140 ms  2.214 ms  2.187 ms
 2  86:201:2::3 (86:201:2::3)  2.786 ms  2.945 ms  2.921 ms
 3  86:201:2::1 (86:201:2::1)  4.767 ms  4.783 ms  4.839 ms
```

Figure 11: Traceroute from SpaceX_3 to NASA_4

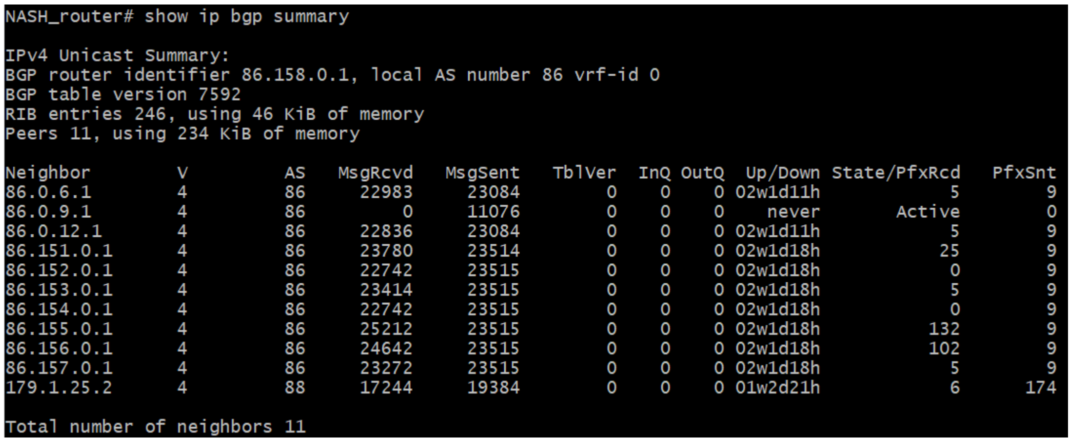
As configured by the steps followed in Question 1.3, the network could also redirect other traffic through the links NEWY-PITT and STLO-PITT. This will occur in cases where one of the other links in the network has failed, so the dynamic routing algorithm will have to use the backup links NEWY-PITT and STLO-PITT. Therefore, under these specific circumstances there might also be traffic which is not directly connected to the two Data Centers. For instance, if the link PITT-NASH has failed or is overloaded, and traffic is sent from CHIC to NASH then the dynamic routing algorithm will redirect the traffic through the link PITT-STLO (back-up link), in order for the data to reach router NASH.

2 Q2 Inter-Domain Routing

2.1 Q2.1 (Alkinoos)

In this part, the iBGP sessions between all the routers in the network are configured by using the *neighbor IP_addr remote-as 86* command with the IP_addr being the loopback address of the peer router. Following this, it is also important to use the command *neighbor IP_addr update-source lo*, which will instruct the source router to always

use its loopback address as its source address when communicating over the iBGP session with the peer router. This step is crucial in order to ensure that the iBGP session between the two peer routers is always functional even if the physical interface between the two routers is down. Following this approach, the source router is using its loopback address, which will always be functional, as it is a virtual address and the iBGP communication will still be possible by reaching the destination peer router through a different path. As a result, the iBGP sessions between any routers will not be affected by links which might be torn down for a period of time.
All the iBGP sessions of the NASH router with all the other routers in the local network can be observed by running the *show ip bgp summary* command. The result can be seen in Figure 12:



```
NASH_router# show ip bgp summary

IPv4 Unicast Summary:
BGP router identifier 86.158.0.1, local AS number 86 vrf-id 0
BGP table version 7592
RIB entries 246, using 46 KiB of memory
Peers 11, using 234 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt
86.0.6.1        4         86     22983     23084        0    0    0 02w1d11h            5        9
86.0.9.1        4         86         0     11076        0    0    0    never       Active        0
86.0.12.1       4         86     22836     23084        0    0    0 02w1d11h            5        9
86.151.0.1      4         86     23780     23514        0    0    0 02w1d18h           25        9
86.152.0.1      4         86     22742     23515        0    0    0 02w1d18h            0        9
86.153.0.1      4         86     23414     23515        0    0    0 02w1d18h            5        9
86.154.0.1      4         86     22742     23515        0    0    0 02w1d18h            0        9
86.155.0.1      4         86     25212     23515        0    0    0 02w1d18h          132        9
86.156.0.1      4         86     24642     23515        0    0    0 02w1d18h          102        9
86.157.0.1      4         86     23272     23515        0    0    0 02w1d18h            5        9
179.1.25.2      4         88     17244     19384        0    0    0 01w2d21h            6      174

Total number of neighbors 11
```

Figure 12: IP BGP summary of router NASH

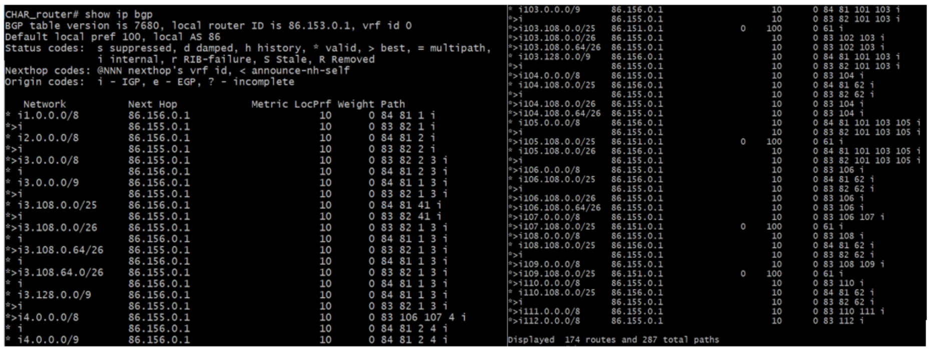
2.2 Q2.2 (Alkinoos)

Following the iBPG sessions, the eBGP sessions can be established with the routers of the neighbouring ASs and the IXP. The steps followed to establish an eBGP session (for example the eBGP session from the DETR router of our AS (179.1.18.2/24) to the CHAR router of AS83 (179.1.18.1/24)) were:

1. Configure router (*conf t*)
2. Configure static route to 86.0.0.0/8 and set next hop to Null0 so that it can be advertised by the router to its neighbours (*ip route 86.0.0.0/8 Null0*)
3. Add the IP address in the interface ext_83_CHAR (*ip address 179.1.18.2/24*)
4. Set up a route-map that permits all of the advertised and received prefixes (*route-map ACCEPT_ALL permit 10*)
5. Configure the BGP session of the DETR router (router bgp 86) and establish an eBGP session with the CHAR router of AS83 (*neighbor 179.1.18.1 remote-as 83*)
6. Apply the route-map on the BGP session (*neighbor 179.1.18.1 route-map ACCEPT_ALL in*(repeat with *out*))
7. Advertise the prefix on the BGP session (*network 86.0.0.0/24*)
8. Run the BGP next-hop-self command for all the iBGP sessions of the DETR router by using the loopback addresses of all the routers. Due to the fact that the iBGP sessions are configured in a full mesh, the next-hop-self configuration needs to be performed in between all the routers in the AS.
   Therefore, for each one of the iBGP sessions it is specified that the next-hop should be the DETR router:
   neighbor 86.151.0.1 remote-as 86, neighbor 86.151.0.1 next-hop-self *//BROO*
   neighbor 86.152.0.1 remote-as 86, neighbor 86.152.0.1 next-hop-self *//NEWY*
   neighbor 86.153.0.1 remote-as 86, neighbor 86.153.0.1 next-hop-self *//CHAR*
   neighbor 86.154.0.1 remote-as 86, neighbor 86.154.0.1 next-hop-self *//PITT*
   neighbor 86.156.0.1 remote-as 86, neighbor 86.156.0.1 next-hop-self *//CHIC*
   neighbor 86.157.0.1 remote-as 86, neighbor 86.157.0.1 next-hop-self *//STLO*

neighbor 86.158.0.1 remote-as 86, neighbor 86.158.0.1 next-hop-self *//NASH*

The next-hop-self command is necessary when advertising eBGP information over iBGP sessions. For every router with an eBGP connection in our AS it is crucial to also configure all the iBGP sessions to the other routers as well so that they can use it as a next-hop in case they receive an eBGP advertisement from it through the iBGP connection. That is because the routers inside the AS do not recognise IP addresses outside of the AS and therefore any eBGP advertisement with unknown IP address would be discarded. However, if we explicitly specify the next-hop for such an advertisement as being the router that the advertisement came from, then they know they can use this router to get to the advertised IP address.
In the example above, the DETR router holds an eBGP connection to the CHAR router of AS83 and the IP addresses from this router will be advertised to all of the other routers inside our AS through iBGP. Therefore, with the commands shown above, in the iBGP sessions of the DETR router with these routers it is specified that these routers should use DETR as the next-hop for any advertisements they receive from it. That way they know how to reach all the IP addresses of the other AS advertised through the eBGP connection.
The configuration of the iBGP and the eBGP connections for the CHAR router can be seen in Figure 13:

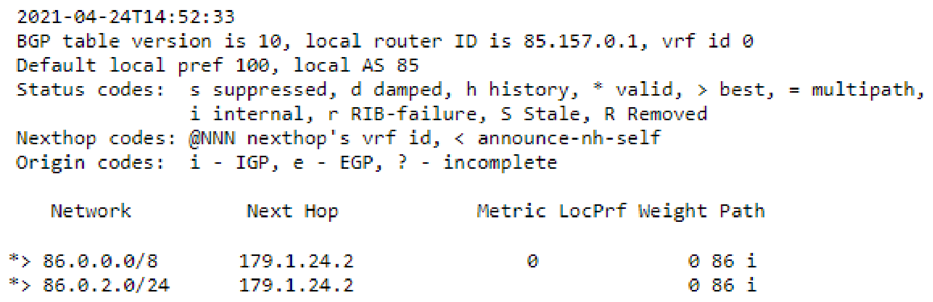


```
CHAR_router# show ip bgp
BGP table version is 7680, local router ID is 86.153.0.1, vrf id 0
Default local pref 100, local AS 86
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i1.0.0.0/8        86.156.0.1                    10      0 84 81 1 i
*>i                 86.155.0.1                    10      0 83 82 1 i
* i2.0.0.0/8        86.156.0.1                    10      0 84 81 2 i
*>i                 86.155.0.1                    10      0 83 82 2 i
*>i3.0.0.0/8        86.155.0.1                    10      0 83 82 2 3 i
* i                 86.156.0.1                    10      0 84 81 2 3 i
* i3.0.0.0/9        86.156.0.1                    10      0 84 81 1 3 i
*>i                 86.155.0.1                    10      0 83 82 1 3 i
* i3.108.0.0/25     86.156.0.1                    10      0 84 81 41 i
*>i                 86.155.0.1                    10      0 83 82 41 i
*>i3.108.0.0/26     86.155.0.1                    10      0 83 82 1 3 i
* i                 86.156.0.1                    10      0 84 81 1 3 i
*>i3.108.0.64/26    86.155.0.1                    10      0 83 82 1 3 i
* i                 86.156.0.1                    10      0 84 81 1 3 i
*>i3.108.64.0/26    86.155.0.1                    10      0 83 82 1 3 i
* i                 86.156.0.1                    10      0 84 81 1 3 i
* i3.128.0.0/9      86.156.0.1                    10      0 84 81 1 3 i
*>i                 86.155.0.1                    10      0 83 82 1 3 i
*>i4.0.0.0/8        86.155.0.1                    10      0 83 106 107 4 i
* i                 86.156.0.1                    10      0 84 81 2 4 i
* i4.0.0.0/9        86.156.0.1                    10      0 84 81 2 4 i
```

```
* i103.0.0.0/9      86.156.0.1                    10      0 84 81 101 103 i
*>i                 86.155.0.1                    10      0 83 82 101 103 i
*>i103.108.0.0/25   86.151.0.1               0   100      0 61 i
*>i103.108.0.0/26   86.155.0.1                    10      0 83 102 103 i
*>i103.108.0.64/26  86.155.0.1                    10      0 83 102 103 i
* i103.128.0.0/9    86.156.0.1                    10      0 84 81 101 103 i
*>i                 86.155.0.1                    10      0 83 82 101 103 i
*>i104.0.0.0/8      86.155.0.1                    10      0 83 104 i
* i104.108.0.0/25   86.156.0.1                    10      0 84 81 62 i
*>i                 86.155.0.1                    10      0 83 82 62 i
*>i104.108.0.0/26   86.155.0.1                    10      0 83 104 i
*>i104.108.0.64/26  86.155.0.1                    10      0 83 104 i
* i105.0.0.0/8      86.156.0.1                    10      0 84 81 101 103 105 i
*>i                 86.155.0.1                    10      0 83 82 101 103 105 i
*>i105.108.0.0/25   86.151.0.1               0   100      0 61 i
* i105.108.0.0/26   86.156.0.1                    10      0 84 81 101 103 105 i
*>i                 86.155.0.1                    10      0 83 82 101 103 105 i
*>i106.0.0.0/8      86.155.0.1                    10      0 83 106 i
* i106.108.0.0/25   86.156.0.1                    10      0 84 81 62 i
*>i                 86.155.0.1                    10      0 83 82 62 i
*>i106.108.0.0/26   86.155.0.1                    10      0 83 106 i
*>i106.108.0.64/26  86.155.0.1                    10      0 83 106 i
*>i107.0.0.0/8      86.155.0.1                    10      0 83 106 107 i
*>i107.108.0.0/25   86.151.0.1               0   100      0 61 i
*>i108.0.0.0/8      86.155.0.1                    10      0 83 108 i
* i108.108.0.0/25   86.156.0.1                    10      0 84 81 62 i
*>i                 86.155.0.1                    10      0 83 82 62 i
*>i109.0.0.0/8      86.155.0.1                    10      0 83 108 109 i
*>i109.108.0.0/25   86.151.0.1               0   100      0 61 i
*>i110.0.0.0/8      86.155.0.1                    10      0 83 110 i
* i110.108.0.0/25   86.156.0.1                    10      0 84 81 62 i
*>i                 86.155.0.1                    10      0 83 82 62 i
*>i111.0.0.0/8      86.155.0.1                    10      0 83 110 111 i
*>i112.0.0.0/8      86.155.0.1                    10      0 83 112 i

Displayed  174 routes and 287 total paths
```

Figure 13: CHAR router - show ip bgp

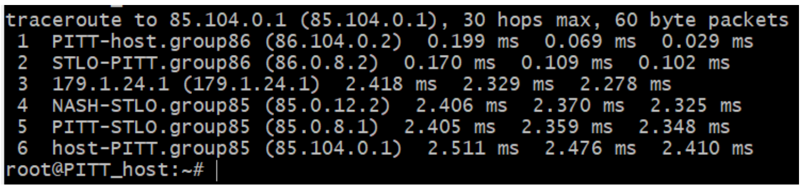
Then, a further test is performed to ensure that the neighboring ASs receive the 86.0.0.0/8 prefix advertised by our routers to the eBGP sessions. For that purpose, the looking glass of router STLO of AS85 is observed. This router is connected to the STLO router of our AS with a peer2peer connection. As it can be seen in Figure 14, the prefix 86.0.0.0/8 has been advertised through the eBGP session and this AS can now reach our AS through their STLO router.



```
2021-04-24T14:52:33
BGP table version is 10, local router ID is 85.157.0.1, vrf id 0
Default local pref 100, local AS 85
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path

*> 86.0.0.0/8       179.1.24.2               0             0 86 i
*> 86.0.2.0/24      179.1.24.2                             0 86 i
```

Figure 14: Looking glass of router STLO - group 85

Finally, a traceroute is performed between the PITT-host of our AS and the PITT-host of AS85. In Figure 15, it is obvious that the communication between the hosts was successful.



```
traceroute to 85.104.0.1 (85.104.0.1), 30 hops max, 60 byte packets
 1  PITT-host.group86 (86.104.0.2)  0.199 ms  0.069 ms  0.029 ms
 2  STLO-PITT.group86 (86.0.8.2)  0.170 ms  0.109 ms  0.102 ms
 3  179.1.24.1 (179.1.24.1)  2.418 ms  2.329 ms  2.278 ms
 4  NASH-STLO.group85 (85.0.12.2)  2.406 ms  2.370 ms  2.325 ms
 5  PITT-STLO.group85 (85.0.8.1)  2.405 ms  2.359 ms  2.348 ms
 6  host-PITT.group85 (85.104.0.1)  2.511 ms  2.476 ms  2.410 ms
root@PITT_host:~# |
```

Figure 15: Traceroute to the PITT-host of AS85

3 Q3 Policy Routing

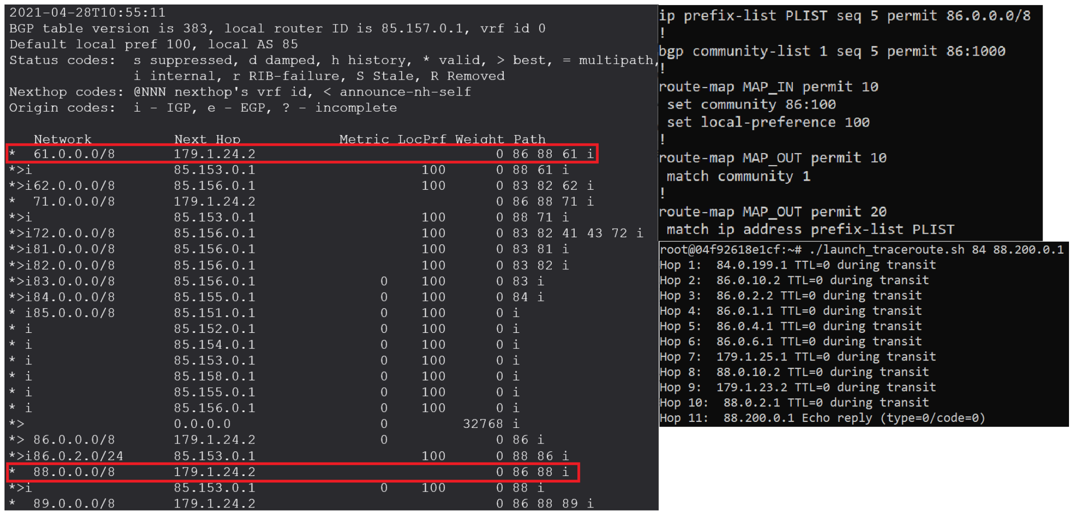
3.1 Q3.1 (Fabian)

As it can be seen in Figure 16 the community value is corresponding to the local preference value. For a connection to a provider, this means the router is a customer, the local preference is set to 10 and therefore the community value is 86:10, for a peer connection the preference is 100 and the community value 86:100 and finally for a connection to a customer, meaning the router being the provider a local preference of 1000 and a community value of 86:1000 is assigned. The route-map of the STLO router is shown in Figure 17.2. There it can be seen that on incoming

Receiving router	Role	BGP community	local preference
CHIC	customer	86:10	10
DETR	customer	86:10	10
STLO	peer	86:100	100
BROO	peer	86:100	100
NASH	provider	86:1000	1000
CHAR	provider	86:1000	1000

Figure 16: BGP communities

routes the community value and the local preference is assigned. The outgoing routes are only allowed if either their community value corresponds to a customer, as only customers should be advertised to peers, or if the prefix is from the own network, namely 86.0.0.0/8. The route-map for a provider is nearly the same, only the community value and the local preference of the incoming routes is different. The same for a customer route but additionally for those routes the output route-map accepts all routes, as all reachable routes should be advertised to the customer. A list of the BGP routes of group 85 (Figure 17.1), which is our direct peer shows, that out customer is seen but our peer is only seen through a wrongly configured customer. And finally a traceroute from our provider, namely Group 84 to a host in our customers network, namely 88.200.0.1, shows a perfectly working inter-domain routing architecture (Figure 17.3).



```
2021-04-28T10:55:11
BGP table version is 383, local router ID is 85.157.0.1, vrf id 0
Default local pref 100, local AS 85
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  61.0.0.0/8       179.1.24.2                             0 86 88 61 i
*>i                 85.153.0.1                    100      0 88 61 i
*>i62.0.0.0/8       85.156.0.1                    100      0 83 82 62 i
*  71.0.0.0/8       179.1.24.2                             0 86 88 71 i
*>i                 85.153.0.1                    100      0 88 71 i
*>i72.0.0.0/8       85.156.0.1                    100      0 83 82 41 43 72 i
*>i81.0.0.0/8       85.156.0.1                    100      0 83 81 i
*>i82.0.0.0/8       85.156.0.1                    100      0 83 82 i
*>i83.0.0.0/8       85.156.0.1               0    100      0 83 i
*>i84.0.0.0/8       85.155.0.1               0    100      0 84 i
* i85.0.0.0/8       85.151.0.1               0    100      0 i
* i                 85.152.0.1               0    100      0 i
* i                 85.154.0.1               0    100      0 i
* i                 85.153.0.1               0    100      0 i
* i                 85.158.0.1               0    100      0 i
* i                 85.155.0.1               0    100      0 i
* i                 85.156.0.1               0    100      0 i
*>                  0.0.0.0                  0         32768 i
*> 86.0.0.0/8       179.1.24.2               0             0 86 i
*>i86.0.2.0/24      85.153.0.1                    100      0 88 86 i
*  88.0.0.0/8       179.1.24.2                             0 86 88 i
*>i                 85.153.0.1               0    100      0 88 i
*  89.0.0.0/8       179.1.24.2                             0 86 88 89 i
```

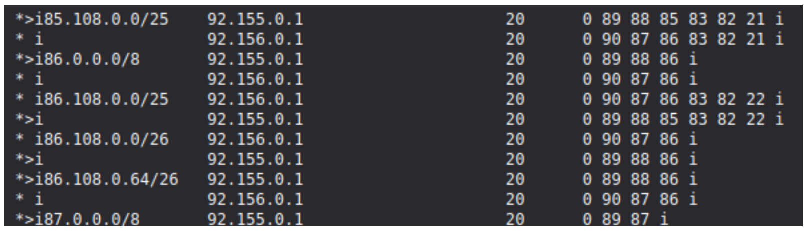
```
ip prefix-list PLIST seq 5 permit 86.0.0.0/8
!
bgp community-list 1 seq 5 permit 86:1000
!
route-map MAP_IN permit 10
 set community 86:100
 set local-preference 100
!
route-map MAP_OUT permit 10
 match community 1
!
route-map MAP_OUT permit 20
 match ip address prefix-list PLIST
```

```
root@04f92618e1cf:~# ./launch_traceroute.sh 84 88.200.0.1
Hop 1:  84.0.199.1 TTL=0 during transit
Hop 2:  86.0.10.2 TTL=0 during transit
Hop 3:  86.0.2.2 TTL=0 during transit
Hop 4:  86.0.1.1 TTL=0 during transit
Hop 5:  86.0.4.1 TTL=0 during transit
Hop 6:  86.0.6.1 TTL=0 during transit
Hop 7:  179.1.25.1 TTL=0 during transit
Hop 8:  88.0.10.2 TTL=0 during transit
Hop 9:  179.1.23.2 TTL=0 during transit
Hop 10:  88.0.2.1 TTL=0 during transit
Hop 11:  88.200.0.1 Echo reply (type=0/code=0)
```

Figure 17: Screenshots for Question 3.1

3.2 Q3.2 (Johannes)

The "MAP-IN" for incoming routes and "MAP-OUT" for outgoing routes in the BROO router have to be modified, because it is connected to the IXP 124. We need to modify the eBGP sesssion between the peer to peer connection of the ASs connected to the IXP 124 as well. Figure 18 shows the relevant part of the looking glass of the stub AS 92 in our group.

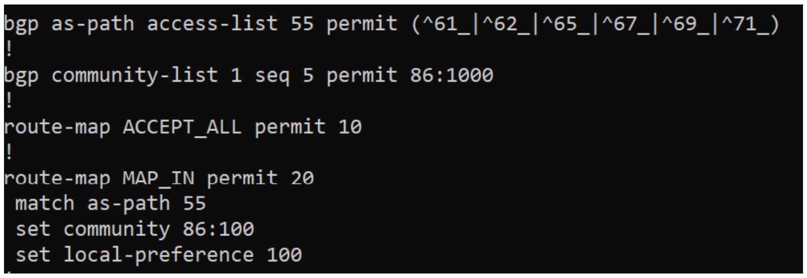


```
*>i85.108.0.0/25    92.155.0.1              20      0 89 88 85 83 82 21 i
* i                 92.156.0.1              20      0 90 87 86 83 82 21 i
*>i86.0.0.0/8       92.155.0.1              20      0 89 88 86 i
* i                 92.156.0.1              20      0 90 87 86 i
* i86.108.0.0/25    92.156.0.1              20      0 90 87 86 83 82 22 i
*>i                 92.155.0.1              20      0 89 88 85 83 82 22 i
* i86.108.0.0/26    92.156.0.1              20      0 90 87 86 i
*>i                 92.155.0.1              20      0 89 88 86 i
*>i86.108.0.64/26   92.155.0.1              20      0 89 88 86 i
* i                 92.156.0.1              20      0 90 87 86 i
*>i87.0.0.0/8       92.155.0.1              20      0 89 87 i
```

Figure 18: Part of looking glass of stub AS 92

3.2.1 MAP-IN

The route-map for incoming routes has to deny all the routes our local network receives from the neighboring networks, i.e. in group 4, and which are connected and sent via IXP 124. We only want to receive routs via the providers and customers directly. The filtering is implemented in the input route-map of the BROO router, which has an interface to IXP 124. Figure 19 shows the necessary lines of code.

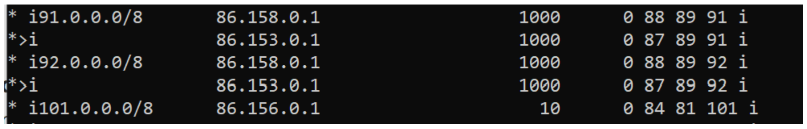


```
bgp as-path access-list 55 permit (^61_|^62_|^65_|^67_|^69_|^71_)
!
bgp community-list 1 seq 5 permit 86:1000
!
route-map ACCEPT_ALL permit 10
!
route-map MAP_IN permit 20
 match as-path 55
 set community 86:100
 set local-preference 100
```

Figure 19: Code of input route-map to sort out necessary routes

It means, incoming routes will be denied, if the first element of the as-path is one of our group. We do not want to receive routes of them via the IXP 124 and therefore, via the BROO router.

The result is shown in Figure 20. One can see that all the routes coming from the ASs in group 3 which are connected to IXP 124 will never reach our local network 86.0.0.0/8 directly via the IXP 124 but via another AS in our neighborhood. Therefore, the filter works.

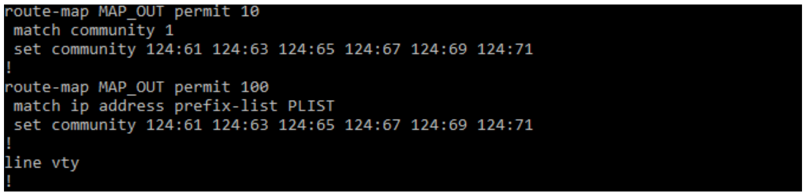


```
* i91.0.0.0/8       86.158.0.1              1000      0 88 89 91 i
*>i                 86.153.0.1              1000      0 87 89 91 i
* i92.0.0.0/8       86.158.0.1              1000      0 88 89 92 i
*>i                 86.153.0.1              1000      0 87 89 92 i
* i101.0.0.0/8      86.156.0.1                10      0 84 81 101 i
```

Figure 20: Part of looking glass of our AS 86

3.2.2 MAP-OUT

The route-map for outgoing advertisements needs to set specific community values. The routes will be sent to IXP 124. Therefore we need to set the community values to 124:n with n the number of the peers of IXP 124 in group 3 in order to advertise to them via the peer2peer connection. The implementation is shown by the following lines of code in Figure 21.

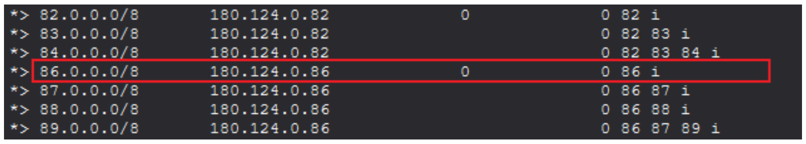


```
route-map MAP_OUT permit 10
 match community 1
 set community 124:61 124:63 124:65 124:67 124:69 124:71
!
route-map MAP_OUT permit 100
 match ip address prefix-list PLIST
 set community 124:61 124:63 124:65 124:67 124:69 124:71
!
line vty
!
```

Figure 21: Code of output route-map to add the necessary community values

With this code we append a list of community values to the existing list.

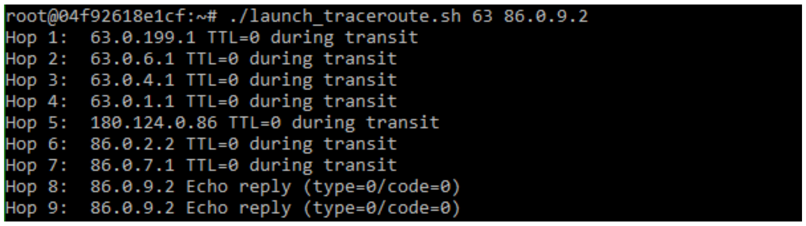
Now one can observe, that the ASs connected to the same IXP 124 in the other group 3 will receive our advertisements directly, since our AS 86 is the leading as-path element. This behaviour is shown in the example of the looking glass of AS 63 in Figure 22.



```
*> 82.0.0.0/8       180.124.0.82            0             0 82 i
*> 83.0.0.0/8       180.124.0.82                          0 82 83 i
*> 84.0.0.0/8       180.124.0.82                          0 82 83 84 i
*> 86.0.0.0/8       180.124.0.86            0             0 86 i
*> 87.0.0.0/8       180.124.0.86                          0 86 87 i
*> 88.0.0.0/8       180.124.0.86                          0 86 88 i
*> 89.0.0.0/8       180.124.0.86                          0 86 87 89 i
```

Figure 22: Part of looking glass of AS 63

Finally there is the traceroute from AS 63 to our NASH router interface 86.0.9.2 in Figure 23. The route hops directly to our AS without passing other ASs, as expected.

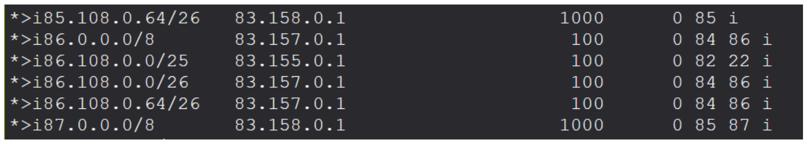


```
root@04f92618e1cf:~# ./launch_traceroute.sh 63 86.0.9.2
Hop 1:  63.0.199.1 TTL=0 during transit
Hop 2:  63.0.6.1 TTL=0 during transit
Hop 3:  63.0.4.1 TTL=0 during transit
Hop 4:  63.0.1.1 TTL=0 during transit
Hop 5:  180.124.0.86 TTL=0 during transit
Hop 6:  86.0.2.2 TTL=0 during transit
Hop 7:  86.0.7.1 TTL=0 during transit
Hop 8:  86.0.9.2 Echo reply (type=0/code=0)
Hop 9:  86.0.9.2 Echo reply (type=0/code=0)
```

Figure 23: Traceroute from AS 63 to NASH router in our AS 86

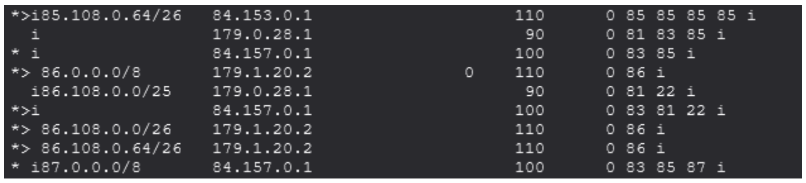
3.3 Q3.3 (Fabian)

To influence the inbound traffic, a rather aggressive method was used. The advertisements of the DETR router were stopped by removing any output route-map, which results in denying all outgoing routes. Therefore all traffic will be guided through the CHIC router. This can be seen in Figures 24 and 25, as AS 83 sends its traffic through 84. The big problem about this solution is, that if something goes wrong with the connection to the provider connected to the CHIC router, connectivity to a huge part of the mini-Internet would be lost as no provider would be available. Another solution would be using the as-path, namely appending three times the AS number 86 on the advertisements of the DETR router instead of once, which leads to a longer path if a route goes through the DETR router. The drawback of this method is, that the traffic from AS 83 through the DETR router, as we are their customer. But all the traffic that does come from a provider of our providers would enter through the CHIC router.



```
*>i85.108.0.64/26   83.158.0.1                   1000      0 85 i
*>i86.0.0.0/8       83.157.0.1                    100      0 84 86 i
*>i86.108.0.0/25    83.155.0.1                    100      0 82 22 i
*>i86.108.0.0/26    83.157.0.1                    100      0 84 86 i
*>i86.108.0.64/26   83.157.0.1                    100      0 84 86 i
*>i87.0.0.0/8       83.158.0.1                   1000      0 85 87 i
```

Figure 24: Routing Table of AS 83

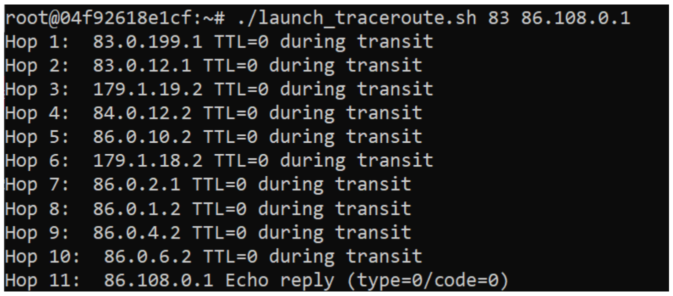


```
*>i85.108.0.64/26   84.153.0.1                          110       0 85 85 85 85 i
  i                 179.0.28.1                           90       0 81 83 85 i
* i                 84.157.0.1                          100       0 83 85 i
*> 86.0.0.0/8       179.1.20.2                     0    110       0 86 i
  i86.108.0.0/25    179.0.28.1                           90       0 81 22 i
*>i                 84.157.0.1                          100       0 83 81 22 i
*> 86.108.0.0/26    179.1.20.2                          110       0 86 i
*> 86.108.0.64/26   179.1.20.2                          110       0 86 i
* i87.0.0.0/8       84.157.0.1                          100       0 83 85 87 i
```

Figure 25: Routing Table of AS 84

3.4 Q3.4 (Fabian)

The hijacked prefix space was found through the looking glass. There was the 86.108.0.0/25 prefix advertised by some AS and the origin of the AS path was AS22. Therefore all traffic within this subnet would have been redirected there, as we only advertised the 86.0.0.0/8 prefix. To attract the traffic back, we began to advertise the two prefixes 86.108.0.0/26 and 86.108.0.64/26 from the NASH router, such that the traffic will be flowing to our AS following the longest prefix match rule. In Figure 26 it can be seen that a traceroute actually goes to the correct location. This countermeasure is only effective for this very specific hijack attack, as we advertise only prefixes that are exactly one bit longer than the hijacked prefix.

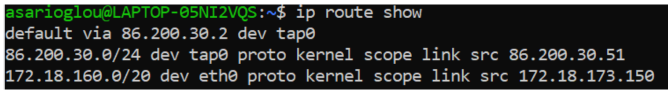


```
root@04f92618e1cf:~# ./launch_traceroute.sh 83 86.108.0.1
Hop 1:  83.0.199.1 TTL=0 during transit
Hop 2:  83.0.12.1 TTL=0 during transit
Hop 3:  179.1.19.2 TTL=0 during transit
Hop 4:  84.0.12.2 TTL=0 during transit
Hop 5:  86.0.10.2 TTL=0 during transit
Hop 6:  179.1.18.2 TTL=0 during transit
Hop 7:  86.0.2.1 TTL=0 during transit
Hop 8:  86.0.1.2 TTL=0 during transit
Hop 9:  86.0.4.2 TTL=0 during transit
Hop 10:  86.0.6.2 TTL=0 during transit
Hop 11:  86.108.0.1 Echo reply (type=0/code=0)
```

Figure 26: Traceroute from AS 83 to the host 86.108.0.1

3.5 Q3.5 - Bonus Question (Alkinoos)

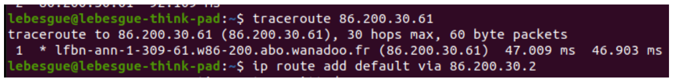
As soon as the VPN connection is established the host is assigned the IP address 86.200.30.51/24. As a result the prefix space that is reachable by this host is 86.200.30.0/24 as seen by an *ip route show* command in Figure 27. Following this, an IPv6 address is configured for the local device (86:200:3::1/48) and the default gateway both for



```
asarioglou@LAPTOP-05NI2VQS:~$ ip route show
default via 86.200.30.2 dev tap0
86.200.30.0/24 dev tap0 proto kernel scope link src 86.200.30.51
172.18.160.0/20 dev eth0 proto kernel scope link src 172.18.173.150
```

Figure 27: IP Route for Local Device

the IPv4 and IPv6 networks. The default gateway for the new host is set to be the NEWY router (86.200.30.2 (IPv4) and 86:200:3::7 (IPv6) for VLAN 30) as for the other hosts in DC1. The default gateway ensures that the traffic from the VPN client will be redirected to the mini-Internet and not to an actual server in the real Internet. In Figure 28, it can be seen that if the default gateway is not changed the ping to an IP address inside the 86.200.30.0/24 subnet will find a server in the real Internet.



```
lebesgue@lebesgue-think-pad:~$ traceroute 86.200.30.61
traceroute to 86.200.30.61 (86.200.30.61), 30 hops max, 60 byte packets
 1  * lfbn-ann-1-309-61.w86-200.abo.wanadoo.fr (86.200.30.61)  47.009 ms  46.903 ms
lebesgue@lebesgue-think-pad:~$ ip route add default via 86.200.30.2
```

Figure 28: Faulty traceroute to 86.200.30.61

That is why we set the default gateways inside the mini_Internet to one of the interfaces of the NEWY or BROO routers. A way to verify that the traffic is indeed redirected to the mini Internet is to run a traceroute to 86.108.0.1 instead of a ping and observe the path taken.