# SONIC

## Seriously
## Overpowered
## Network
## Infrastructure
## Code

Silvia Ritsch
Gina Stoffel
Nando Galliard

**Advanced Communication Networks**
**227-0627-00L**

**ETH Zürich**
August 30, 2022

Hinweis:
Dieser Bericht wurde zu gleichen Teilen verfasst von
Nando Galliard, Gina Stoffel und Silvia Ritsch.

# Contents

# 1 Intra-Domain Routing

## 1.1 Question

In this task, we had to enable direct layer-2 connectivity in-between students and in-between staff members, but not between them. If a staff member wants to reach a student they need to go via one of the layer-3 routers.

### 1.1.1 Distributing IP addresses

We used the following IP addresses for the hosts:

| Host: | | IP Address: |
|---|---|---|
| Student | 1 | 70 . 200 . 0 . 16 /24 |
| Student | 2 | 70 . 200 . 0 . 32 /24 |
| Student | 3 | 70 . 200 . 0 . 48 /24 |
| Staff | 1 | 70 . 200 . 1 . 16 /24 |
| Staff | 2 | 70 . 200 . 1 . 32 /24 |
| Staff | 3 | 70 . 200 . 1 . 48 /24 |

Of particular note here is that the staff and student subnet are distinct. To program this we had to set the IP address for which every host listens to. For example, the Student 1 host who is connected to the CERN switch, we used:

```
ip address add 70.200.0.16/24 dev 70-CERN
```

Now if a host encounters a IP that is not within range of the subnet we also needed a default gateway to pass the request along. We set up that all the hosts around CERN and EPFL default to the Geneva router (70.200.0/1.1) and ETHZ defaults to Zürich (70.200.0/1.2). Ex:

```
ip route add default via 70.200.0.1
```

### 1.1.2 Setting up the ports

Since we can't use IP addresses on the switch layer (layer 2) we set up ports for students (PORT 10) and staff (PORT 20).

```
ovs-vsctl set port 70-student_1 tag=10
```

To guarantee connectivity between the subnets, we added trunks on the switches such that they can forward the data to the routers which in turn are able to reach both subnets. Ex:

```
ovs-vsctl set port 70-CERN trunks=10,20
```

### 1.1.3 Setting multiple VLAN's for the routers

We had to enable the routers to send packets to the staff and student VLAN. For the interface GENE-L2.10 we've assigned the IP address 70.200.0.1/24 (which is the default gateway for all users and switches in CERN and EPFL) and for GENE-L2.20 70.200.1.1/24. The router ZURI was configured the same way but with the address 70.200.0/1.2/24. The code used for this configuration is (when logged into the router):

```
router# conf t
router(config)# interface GENE-L2.10
router(config-if)# ip address 70.200.0.1/24
```

### 1.1.4 Tracerout outputs

**EPFL student to staff**

```
root@student_3:~# traceroute 70.200.1.48
traceroute to 70.200.1.48 (70.200.1.48), 30 hops max, 60 byte packets
1  70.200.0.1 (70.200.0.1)  9.347 ms  8.031 ms  11.560 ms
2  70.200.1.48 (70.200.1.48)  15.757 ms  15.729 ms  15.373 ms
```

**ETHZ staff to EPFL student**

```
root@staff_2:~# traceroute 70.200.0.48
traceroute to 70.200.0.48 (70.200.0.48), 30 hops max, 60 byte packets
1  70.200.1.2 (70.200.1.2)  6.673 ms  4.110 ms  6.236 ms
2  70.200.0.48 (70.200.0.48)  29.068 ms * *
```

**EPFL student to ETHZ staff**

```
root@student_3:~# traceroute 70.200.1.32
traceroute to 70.200.1.32 (70.200.1.32), 30 hops max, 60 byte packets
1  70.200.0.1 (70.200.0.1)  7.820 ms  4.943 ms  4.883 ms
2  * 70.200.1.32 (70.200.1.32)  12.577 ms *
```

Since none of the above connections run in the same subnet they all default to their router which is visible in line 1 with 70.200.0/1.1/2, which then sends it to the correct link. Additionally the switches do not show up since they don't have a designated IP and just pass the data along.

## 1.2 Question

We set the routers and the hosts up according to the requirements. The resulting traceroute from the PARI-host to the ATLA-host shows:

```
root@PARI_host:~# traceroute 70.107.0.1
traceroute to 70.107.0.1 (70.107.0.1), 30 hops max, 60 byte packets
1  PARI-host.group70 (70.103.0.2)  1.469 ms  1.978 ms  1.961 ms
2  NEWY-PARI.group70 (70.0.5.2)  4.751 ms MIAM-PARI.group70 (70.0.6.2)
    2.510 ms NEWY-PARI.group70 (70.0.5.2)  4.722 ms
3  ATLA-NEWY.group70 (70.0.11.2)  5.272 ms ATLA-MIAM.group70 (70.0.13.1)
    3.737 ms  3.723 ms
4  host-ATLA.group70 (70.107.0.1)  5.227 ms  4.397 ms  5.198 ms
```

## 1.3 Question

### 1.3.1 Loadbalancing and submarine links

Since our main objective was to minimize the latency we had to make sure that no connection attempt uses two or more submarine cables. To achieve this we attributed the submarine cables a much higher weight making it less incentivised to use those connections. In addition they are categorized in a way that the submarine cables with the highest bandwidth have the lowest weight in comparison.

An additional objective was load balancing between three different start- and endpoints. The most notable here is ZURI - ATLA since the route includes a submarine cable. In the end one just has to find a weight distribution that all load balanced connections have the same total weight. For the example ZURI - ATLA:

| Route 1 | Weight: | Route 2 | Weight: |
|---|---|---|---|
| ZURI - LOND | 3 | ZURI - PARI | 2 |
| LOND - NEWY | 9 | PARI - MIAM | 10 |
| NEWY - ATLA | 1 | MIAM - ATLA | 1 |
| TOTAL | 13 | TOTAL | 13 |

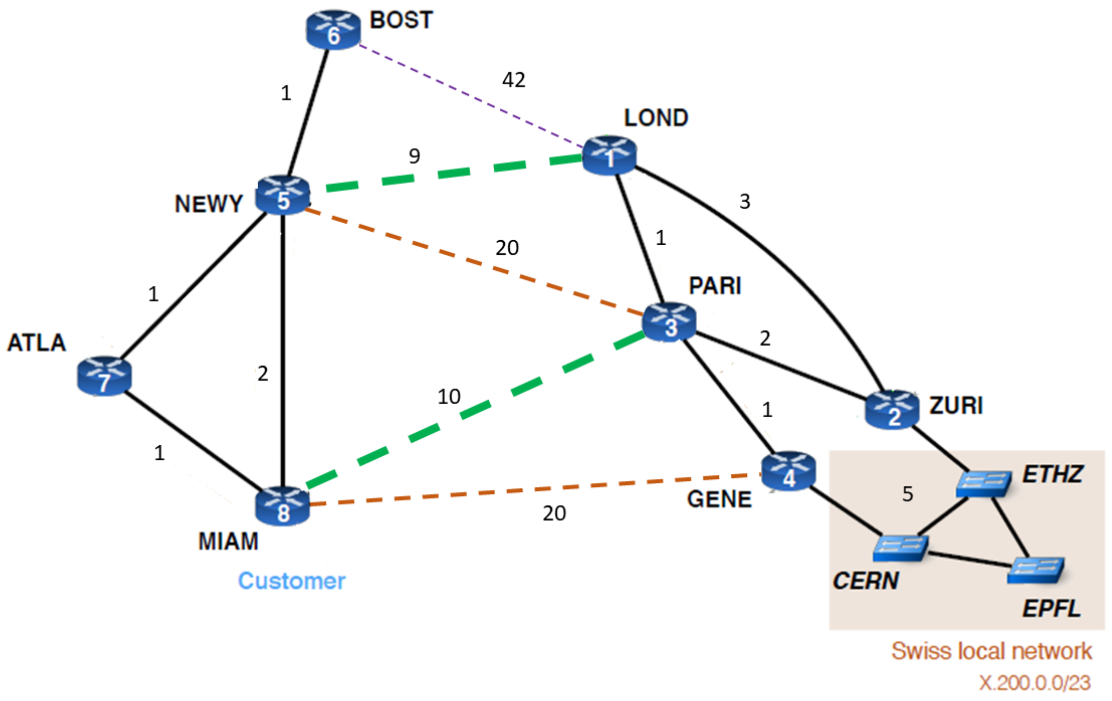

Figure 1: Total weight distribution of cables

### 1.3.2 Tracetroute ATLA host to loopback interface ZURI

```
root@ATLA_host:~# traceroute 70.152.0.1
traceroute to 70.152.0.1 (70.152.0.1), 30 hops max, 60 byte packets
1  ATLA-host.group70 (70.107.0.2)  0.315 ms  0.320 ms  0.046 ms
2  NEWY-ATLA.group70 (70.0.11.1)  0.275 ms  0.281 ms
    MIAM-ATLA.group70 (70.0.13.2)  0.283 ms
3  PARI-MIAM.group70 (70.0.6.1)  0.541 ms  0.523 ms
    LOND-NEWY.group70 (70.0.8.1)  1.301 ms
4  70.152.0.1 (70.152.0.1)  5.097 ms  4.957 ms  4.939 ms
```

We see that the traceroute has two possible options to go to. This is expected since we configured the system in a way that both options are equally appealing. Of special note is that the route over NEWY is slower than the line over MIAM and yet both are considered since we tutored just for bandwidth of the submarine cables.

## 1.4 Question

### 1.4.1 ZURI PARI rerouting

For this problem we've used static routes to enforce the desired paths. We configured the ZURI router to send all traffic for PARI to GENE. Additionally the GENE router has a static route to send all this traffic further to PARI. To obtain further security we assigned the weight 1 between GENE-PARI and the weight 2 between ZURI-PARI. The route GENE-PARI is therefore cheaper and preferred over ZURI-PARI. To assure no transit traffic between ZURI and GENE we added the weight 5 on this path. Traffic therefore prefers to go over PARI with a cost of 3 instead of the direct way with a cost of 5.
In our traceroute the Interface PARI-ZURI was showing up in line 3 from time to time. This was due to the fact that traceroute takes the shortest return path and therefore can take the direct way back to ZURI as this route isn't static.

A general drawback is, that if a link fails, the router will still try to use the static route instead of rerouting the traffic thus failing the connection and not finding a replacement. A further drawback for the network operators is, that the static configuration is rather time consuming and not really scaleable for large networks.

```
root@staff_2:~# traceroute 70.103.0.1
traceroute to 70.103.0.1 (70.103.0.1), 30 hops max, 60 byte packets
 1  70.200.1.2 (70.200.1.2)  3.025 ms  2.955 ms  2.930 ms
 2  70.200.1.1 (70.200.1.1)  5.731 ms  5.711 ms  5.690 ms
 3  PARI-GENE.group70 (70.0.3.1)  5.686 ms  5.664 ms  5.643 ms
 4  host-PARI.group70 (70.103.0.1)  5.907 ms  5.885 ms  5.863 ms
```

## 1.5 Question

### 1.5.1 Internal BGP session

Update-source is used to have a hardware independent IP address. With this solution a BGP connection doesn't fail, when an interface gets interrupted. Therefore the neighbour will look for a path to the loopback address instead of trying to reach a certain interface. Furthermore the "show ip bgp summary" command for ATLA shows:

```
Neighbor        V   AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
70.151.0.1      4   70   14480   14463        0    0    0 01w3d00h            2
70.152.0.1      4   70  139995   14439        0    0    0 01w3d00h            6
70.153.0.1      4   70   14787   14445        0    0    0 01w3d00h           29
70.154.0.1      4   70   14439   14443        0    0    0 01w3d00h            2
70.155.0.1      4   70   14872   14498        0    0    0 01w3d01h          114
70.156.0.1      4   70  492637   14444        0    0    0 01w3d00h            2
70.158.0.1      4   70   14495   14498        0    0    0 01w3d01h            1
179.1.81.2      4   71     273     685        0    0    0 04:29:28            1
```

Here is of not that 70.157.0.1 is missing. This is because the ATLA router itself is 70.157.0.1 and thus not necessary to designate as a neighbor.

# 2 Inter-Domain Routing

## 2.1 Question

### 2.1.1 Next-hop-self

iBGP doesn't change the next hop IP address. If a different AS therefore advertises its prefix to our AS, it could cause reach-ability problems. For example: From our ATLA router we use eBGP to connect to AS 71. Our MIAM router would therefore see the IP address 179.1.81.1/24 of AS 71 as next hop, which MIAM potentially doesn't know how to reach. Next-hop-self solves this problem by switching the next-hop address to its own address. The MIAM router therefore knows it can reach the advertised network from AS 71 by using its link to the ATLA router. So "next-hop-self" is required on every iBGP session which has a connection to a eBGP. In our case this matches to all our iBGP's.
As requested one can see the prefixes of the neighboring ASes of our PARI router in the code snippet below

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i67.0.0.0/8       70.152.0.1               0    256      0 67 i
* i68.0.0.0/8       70.156.0.1               0    256      0 68 i
*>i                 70.151.0.1               0    256      0 68 i
*> 69.0.0.0/8       179.1.78.1               0    512      0 69 i
*>i71.0.0.0/8       70.157.0.1               0   1024      0 71 i
*                   179.1.78.1                    512      0 69 71 i
*>i72.0.0.0/8       70.154.0.1               0   1024      0 72 i
*                   179.1.78.1                    512      0 69 72 i
```

### 2.1.2 Prefix advertisement

Next, we show a looking glass table from LOND router of group 68. Since only our prefix advertisement is of importance, we only provide a part of the table for better readability. As one can see the prefix of our group 70 is advertised correctly to the neighboring group 68 with a /8 mask.

```
* i70.0.0.0/8       68.158.0.1               0   5000      0 70 i
*>i                 68.154.0.1               0   5000      0 70 i
```

### 2.1.3 Traceroute from PARI-host

Traceroute from PARI-host Group 70 to neighboring PARI-host Group 68

```
    traceroute to 68.103.0.2 (68.103.0.2), 30 hops max, 60 byte packets
1  PARI-host.group70 (70.103.0.2)  1.101 ms  1.179 ms  1.159 ms
2  LOND-PARI.group70 (70.0.4.2)  3.435 ms  3.422 ms  3.406 ms
3  179.1.73.1 (179.1.73.1)  4.670 ms  5.308 ms  5.308 ms
4  PARI-host.group68 (68.103.0.2)  5.661 ms  5.646 ms  5.667 ms
```

## 2.2 Question

### 2.2.1 Route-Map

At first we used the predetermined community values to relay our BGP advertisements to our peers via our IXP. For that we set:

```
route-map IXP_COMMUNITY_MAP_70 permit 1
set community 123:41 123:43 123:45 123:47 123:49 123:51 123:53
    123:62 123:64 123:66 123:68 123:72
```

With questions 3.2 we realised that our solution leads to a problem with the IXP policies and the naming convention. Our Region 3 shouldn't connect to us via the IXP or exchange advertisements via it. In the end we cut the AS's from Region 3 out of the route-map and changed the names of the routemaps:

```
route-map 70.fromCustomer permit 1
match community 1
set community 123:41 123:43 123:45 123:47 123:49 123:51 123:53
!
route-map 70.fromCustomer permit 2
match ip address prefix-list IntraList
set community 123:41 123:43 123:45 123:47 123:49 123:51 123:53
```

In the first line we create a route-map for which we allow input data. Afterwards we apply this route-map to community 1 (70:10) which we discuss further in question 3.1. In the third line we set the community to all the AS's in Region 2 connected via our IXP which will be allowed to receive the advertisement. In the second part we do the same except we send everything that originates in our AS which is accomplished with "match ip address prefix-list IntraList". This we discuss further in 3.1 but in short it makes sure that all advertisements from our own AS70 get out.

### 2.2.2 Looking Glass

Below, one can see a looking glass entry from AS45 NEWY router. One can see under next hop that the connection goes out to the IXP 123 before being sent along to AS70.

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 70.0.0.0/8       180.123.0.70             0     20      0 70 i
```

### 2.2.3 Traceroute

In this section we launched a traceroute from AS45 to our ATLA router. In the forth Hop one can see that the traceroute goes via the IXP.

```
./launch_traceroute.sh 45 70.107.0.1
Hop 1:  45.0.199.1 TTL=0 during transit
Hop 2:  45.0.4.2 TTL=0 during transit
Hop 3:  45.0.8.2 TTL=0 during transit
Hop 4:  180.123.0.70 TTL=0 during transit
Hop 5:  70.0.11.2 TTL=0 during transit
Hop 6:  70.107.0.1 Echo reply (type=0/code=0)
```

# 3 Policy Routing

## 3.1 Question

| | Community-list | Community-values | LP |
|---|---|---|---|
| Customer | 1 | 70:10 | 1024 |
| Peer | 2 | 70:09 | 512 |
| Provider | 3 | 70:08 | 256 |

Since the community-level list is a question of definition we just went for 1 to 3. For the local-preference (LP) it was important that customer >peer >provider, which is achieved with higher LP's for preferred communities. The community value, we chose based on the power of two of the LP.

### 3.1.1 In out route-map

```
 1   address-family ipv4 unicast
 2       ...
 3       neighbor 179.1.70.1 route-map 70.toProvider in
 4       neighbor 179.1.70.1 route-map 70.fromCustomer out
 5   exit-address-family
 6   !
 7   ...
 8   !
 9   ip prefix-list IntraList seq 5 permit 70.0.0.0/8
10   !
11   bgp community-list 1 permit 70:10
12   !
13   route-map 70.fromCustomer permit 10
14           match community 1
15   !
16   route-map 70.fromCustomer permit 5
17     match ip address prefix-list IntraList
18   !
19   route-map 70.toProvider permit 10
20       set community 70:8
21       set local-preference 256
```

We are looking at the ZURI router which is connected to a provider, namely AS67.
We tag the connection going to one of our providers as 70.toProvider, which can be seen in line 3. The connection coming over the Intranet from one of our customers, we filter with a 70.fromCustomer tag. This can be seen in line 4. In hindsight this was not the best naming convention, since we sometimes mixed up the names. A connection that goes out (to) can also come in (from), so it is not directly understandable.

It is not enough to just say the incoming advertisements is a provider. We specified in line 19 to 20 a community and a local-preference which designate all the incoming advertisements as a provider.

We then need to filter the outgoing advertisements since we want our customer to have every possible option. But to our providers and peers we only want to show our customer connections.

This we achieve with line 4, 14 and 11 in which we give the permit for 70:10. 70 for our AS and 10 as our predefined customer community value.
In contrast, for a customer connected router, we need to allow ALL outgoing advertisements. We simply change our bgp community-list 1 to incorporate every community value:

```
22    bgp community-list 1 permit 70:10
23    bgp community-list 1 permit 70:8
24    bgp community-list 1 permit 70:9
```

That way we send every possible external advertisement along. But not our internal ones as of yet. The last thing we need to make sure is that every connection from our AS70 subnet is allowed to leave over every connection. We achieve this with an additional 70.fromCustomer permit in line 16 to 17 where the IntraList defined in line 9 incorporates every 70.0.0.0/8.

### 3.1.2 Looking glass of peer

By taking a look from the looking glass of AS69 PARI, we see one of our customers AS71 advertised through us.

```
*  71.0.0.0/8       179.1.78.2                        800       0 70 71 i
```

Yet if we look at one of our peers, AS43 for example, we see that we don't advertise them as it is requested of a peer to peer relationship.

```
*>i43.0.0.0/8       69.152.0.1                        600       0 68 41 43 i
```

### 3.1.3 Traceroute from a peer to a customer

Since we are a tier 2 AS and directly connected to a stub AS we can't run a traceroute from a customer to a peer. For the AS such as ours the question got changed to a traceroute from a peer to a customer. We went with the same pair again of AS43 and AS71, with the loopback address of the ZURI router as destination.

```
./launch_traceroute.sh 43 71.152.0.1
Hop 1:  43.0.199.1 TTL=0 during transit
Hop 2:  43.0.4.2 TTL=0 during transit
Hop 3:  179.0.20.1 TTL=0 during transit
Hop 4:  180.123.0.70 TTL=0 during transit
Hop 5:  70.0.11.2 TTL=0 during transit
Hop 6:  71.152.0.1 Echo reply (type=0/code=0)
```

## 3.2 Question

### 3.2.1 Route-map at NEWY

In the first seven lines we defined a community of all the AS's that are connected with us through IXP 123. We allow all of these peers to reach us through IXP 123. On the other hand we set the communities in line 14 and 18 such that we only allow incoming traffic through these peers from region 2. By only allowing incoming traffic through IXP 123 from one of these ASes (43,45,49,51), we implicitly deny traffic through IXP 123 from ASes in our region, namely region 3.

Line 10 permits all customers line 13 tags outgoing traffic as if part of customer tag, you can be sent to one of the communities customer tag is 70:10 line 21 in combination with 1-7 filters incoming traffic

```
1   ip prefix-list IXP_NEIGHBOR_AS seq 1 permit 180.123.0.41/32
2   ip prefix-list IXP_NEIGHBOR_AS seq 2 permit 180.123.0.43/32
3   ip prefix-list IXP_NEIGHBOR_AS seq 3 permit 180.123.0.45/32
4   ip prefix-list IXP_NEIGHBOR_AS seq 4 permit 180.123.0.47/32
5   ip prefix-list IXP_NEIGHBOR_AS seq 5 permit 180.123.0.49/32
6   ip prefix-list IXP_NEIGHBOR_AS seq 6 permit 180.123.0.51/32
7   ip prefix-list IXP_NEIGHBOR_AS seq 7 permit 180.123.0.53/32
8   ip prefix-list IntraList seq 5 permit 70.0.0.0/8
9   !
10  bgp community-list 1 permit 70:10
11  !
12  route-map 70.fromCustomer permit 1
13  match community 1
14  set community 123:41 123:43 123:45 123:47 123:49 123:51 123:53
15  !
16  route-map 70.fromCustomer permit 2
17  match ip address prefix-list IntraList
18  set community 123:41 123:43 123:45 123:47 123:49 123:51 123:53
19  !
20  route-map 70.toPeer permit 1
21  match ip next-hop prefix-list IXP_NEIGHBOR_AS
22  set community 70:9
23  set local-preference 512
```

### 3.2.2 Advertisement of stub AS

Below is an excerpt from the show ip bgp output of the NEWY router. We see that the next hop of our stub AS72 does not advertise through our common IXP 123 but through one of our local routers in the AS70, GENE with 70.154.0.1 which is directly connected to the customer to be precise.

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i72.0.0.0/8       70.154.0.1               0   1024      0 72 i
```

To give an example of how it should not look we can see here that besides the local router we also get a possible next hop over 180.123.0.72 which is not desirable.

```
   Network          Next Hop            Metric LocPrf Weight Path
*  72.0.0.0/8       180.123.0.72             0    512      0 72 i
*>i                 70.154.0.1               0   1024      0 72 i
```

### 3.2.3 Looking glass output

A looking glass code snippet from the ZURI router of stub AS72, which is also connected to IXP 123 is being provided below. In the project questions the looking glass of NEWY was desired yet for server load reasons the stubs got shrunk to just include ZURI and GENE. We are directly connected to a AS72 ZURI router which is why we chose this looking glass.

```
     Network          Next Hop            Metric LocPrf Weight Path
1    *> 70.0.0.0/8       179.1.80.1               0     20      0 70 i
2    *                   179.1.77.1                     20      0 69 70 i
```

We see that no next hop attempt is made over the IXP to reach us. We only get advertised directly or over AS69 since we are in a peer to peer relationship with them and AS69 is in a provider to customer relationship with AS 72.
Again a looking glass code snippet, this time from AS45 which is in a different region but also connected to IXP 123 through its NEWY router.

```
*> 70.0.0.0/8       180.123.0.70             0     20      0 70 i
```

## 3.3 Question

Since we are connected to two providers, namely Group 68 and Group 67, with Group 67 being connected to our ZURI router, we want all traffic coming from providers to reach us through Group 67.
Therefore, we prepended our AS70 once when advertising our network from BOST and LOND to our provider AS68. Now it should be more efficient for provider AS68 to send traffic, destined to our AS70, to AS67 first. With AS67 having a more attractive connection to our AS. However, we can only influence routing decisions of AS68 but not control them. Since AS68 has a provider2customer relationship with us and a peer2peer relationship to AS67 it might still prefer to send traffic directly to us instead over 67. This way our provider AS68 makes money, whereas when it would send the traffic to AS67 it wouldn't gain anything from it. The advertisements of our AS70 from our two providers are shown below. One can see that our prepending worked, but AS68 still prefers to send traffic directly. We also tried this with prepending AS70 five times, but no different results would show. Therefore for simplicity we sticked to prepending it once.
This is also the drawback mentioned in the task description. But since the routing decision is made by AS68 and its local preferences, we don't think there's a feasible way around this.

AS 68 PARI router

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i70.0.0.0/8       68.158.0.1               0   5000      0 70 70 i
*                   179.1.71.1                    500      0 67 70 i
```

AS 67 PARI router

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i70.0.0.0/8       67.157.0.1               0    300      0 70 i
*                   179.1.71.2                    200      0 68 70 70 i
```

## 3.4 Question

To further influence inbound traffic, making the route from AS68 over BOST more attractive than over LOND, we prepended our AS70 once more when advertising to AS68 from LOND. Now BOST artificially prolongs the AS path with one AS70 prepend and LOND prolongs it with two AS70 prepends. Again we also tried this with a factor five and ten, but no different results showed, except for longer advertisements.
Here we actually accomplished that the GENE router of AS68 prefers to route traffic over the MIAM router, which is connected to our AS70 BOST router. Still we cannot guarantee for that route, since the routing decisions are still made by AS68, but our efforts to redirect traffic worked in this case. This can be seen from the two code snippets below. In the first one we see that AS68 GENE router prefers to send traffic to its MIAM router instead of directly sending it to AS70 LOND router. In the second snippet one can see that the AS68 MIAM router directly sends traffic to our AS70 although we prolonged the AS path.

AS 68 GENE router (connected to AS 70 LOND)

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i70.0.0.0/8       68.158.0.1               0   5000      0 70 70 i
*                   179.1.73.2               0   5000      0 70 70 70 i
```

AS 68 MIAM Router (connected to AS 70 BOST)

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 70.0.0.0/8       179.1.72.2               0   5000      0 70 70 i
```

## 3.5 Question

While solving other Policy Routing questions, we began to see hijacked prefixes not configured by us. View from the looking glass:

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i70.0.0.0/8       68.158.0.1               0   5000      0 70 70 i
*                   179.1.71.1                    500      0 67 70 i
*>i70.108.0.0/25    68.151.0.1                     50      0 66 64 62 102 i
* i                 68.152.0.1                     50      0 65 63 61 102 i
* i                 68.156.0.1                     50      0 66 64 62 102 i
*>i70.108.0.128/25  68.151.0.1                     50      0 66 64 62 102 i
* i                 68.152.0.1                     50      0 65 63 61 102 i
* i                 68.156.0.1                     50      0 66 64 62 102 i
```

The hijacked prefix space is 70.108.0.0/24 but since they use a more precise (larger) prefix with 25 and two different prefix spaces than us they get preferred. To mitigate this we one-up them. We split the prefix space in four on all routers with:

```
address-family ipv4 unicast
    network 70.108.0.0/26
    network 70.108.0.64/26
    network 70.108.0.128/26
    network 70.108.0.192/26
```

This way we are now more precise and get preferential treatment on every router. This of course is a race to the bottom and an actual malicious party could just one up us again until we end at the prefix /32 where nothing goes anymore and your router log is really specified and huge. Because of that it isn't the most elegant solution and we suspect this is a problem that network security personnel have to solve to this day.

With this it is not yet done though we need to make sure that the MIAM router doesn't just send it out to the attacker again and creates an oscillation. This we do simply by providing a static route in the MIAM router to the host with:

```
ip route 70.108.0.0/25 host
```

Notice how we send the whole subnet /25 along even though we just need one /32. This is because if we would add an additional interface to the host we don't have to reconfigure the router.
Just to verify the functioning route we sent a traceroute with the measurement container to one of the compromised IP's from the other end of the internet (AS109). We can clearly see that it takes the right way to the host.

```
./launch_traceroute.sh 109 70.103.0.1
Hop 1:  109.0.199.1 TTL=0 during transit
Hop 2:  179.2.30.1 TTL=0 during transit
Hop 3:  108.0.11.1 TTL=0 during transit
Hop 4:  180.125.0.87 TTL=0 during transit
Hop 5:  87.0.10.2 TTL=0 during transit
Hop 6:  87.0.7.1 TTL=0 during transit
Hop 7:  87.0.4.1 TTL=0 during transit
Hop 8:  87.0.2.1 TTL=0 during transit
Hop 9:  86.0.13.1 TTL=0 during transit
Hop 10:  e86.0.8.2 TTL=0 during transit
Hop 11:  67.0.10.1 TTL=0 during transit
Hop 12:  67.0.11.2 TTL=0 during transit
Hop 13:  179.1.70.2 TTL=0 during transit
Hop 14:  70.200.1.1 TTL=0 during transit
Hop 15:  70.0.4.1 TTL=0 during transit
Hop 16:  70.103.0.1 Echo reply (type=0/code=0)
```

## 3.6 Question

We were able to see the Swiss local network but not anything else, probably since we weren't advertising our IP through the routers and were only able to see the direct connections over the switches. We had to advertise the 70.200.30.0/24 prefix space over ospf with:

```
router ospf
    network 70.200.30.0/24 area 0
```

for both ZURI and GENE router. And give the interfaces GENE-L2.30 and ZURI-L2.30 their correct IP. Additionally we would need to add a default gateway to the GENE router but at this point our rasp gave out and we gave in.