WORLD-WIDE-WEB PROJECT

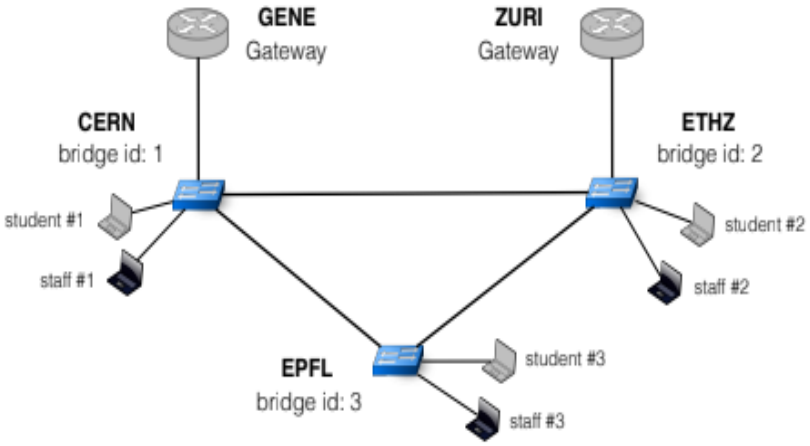
Aggelos Mpallis – csd4689

Dimitris Makrogiannis – csd4676

AS 48

Project Phase One

Question 1.1



This above is our L2 layer topology. We used addresses from the 48.200.0.0/23 range in order to configure it. Below are listed screenshots of the whole configuration, including students, staff and the routers themselves. We differentiated (not really, in abstract) the hosts and routers, setting the 24th bit to 1 for the

routers and 0 for the hosts. We also configured the default gateways for each host, according to projects specifications:

Hosts connected to CERN and EPFL switches -> GENE router.

Hosts connected to ETHZ switch -> ZURI router.

Unfortunately, due to platforms technical issues, real-time connectivity inside the L2 layer is not possible, hence, no “ping” and/or “traceroute” results will be presented for Question 1.1 and Question 1.2

Student 1:

48.200.0.11/23

```
root@student_1:~# ifconfig
48-CERN: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 48.200.0.11  netmask 255.255.254.0  broadcast 0.0.0.0
        ether 2e:eb:58:22:64:8a  txqueuelen 1000  (Ethernet)
        RX packets 528  bytes 37056 (36.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
root@student_1:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         48.200.1.1      0.0.0.0         UG        0 0          0 48-CERN
48.200.0.0      0.0.0.0         255.255.254.0   U         0 0          0 48-CERN
158.48.0.0      0.0.0.0         255.255.0.0     U         0 0          0 ssh
root@student_1:~#
```

Student 2:

48.200.0.21/23

```
root@student_2:~# ifconfig
48-ETHZ: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 48.200.0.21  netmask 255.255.254.0  broadcast 0.0.0.0
        ether f2:30:3c:5a:a1:86  txqueuelen 1000  (Ethernet)
        RX packets 529  bytes 37126 (36.2 KiB)
```

```
root@student_2:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         48.200.1.2      0.0.0.0         UG        0 0          0 48-ETHZ
48.200.0.0      0.0.0.0         255.255.254.0   U         0 0          0 48-ETHZ
158.48.0.0      0.0.0.0         255.255.0.0     U         0 0          0 ssh
root@student_2:~#
```

Student 3:

48.200.0.31/23

```
root@student_3:~# ifconfig
48-EPFL: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 48.200.0.31  netmask 255.255.254.0  broadcast 0.0.0.0
        ether 4a:90:3e:f2:d3:b5  txqueuelen 1000  (Ethernet)
        RX packets 530  bytes 37196 (36.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
root@student_3:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         48.200.1.1      0.0.0.0         UG        0 0          0 48-EPFL
48.200.0.0      0.0.0.0         255.255.254.0   U         0 0          0 48-EPFL
158.48.0.0      0.0.0.0         255.255.0.0     U         0 0          0 ssh
root@student_3:~#
```

Staff 1:

48.200.0.12/23

```
root@staff_1:~# ifconfig
48-CERN: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 48.200.0.12  netmask 255.255.254.0  broadcast 0.0.0.0
        ether be:d0:a6:fd:2f:f7  txqueuelen 1000  (Ethernet)
        RX packets 529  bytes 37126 (36.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
root@staff_1:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         48.200.1.1      0.0.0.0         UG        0 0          0 48-CERN
48.200.0.0      0.0.0.0         255.255.254.0   U         0 0          0 48-CERN
158.48.0.0      0.0.0.0         255.255.0.0     U         0 0          0 ssh
root@staff_1:~#
```

Staff 2:

48.200.0.22/23

```
root@staff_2:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         48.200.1.2      0.0.0.0         UG        0 0          0 48-ETHZ
48.200.0.0      0.0.0.0         255.255.254.0   U         0 0          0 48-ETHZ
158.48.0.0      0.0.0.0         255.255.0.0     U         0 0          0 ssh
root@staff_2:~#
```

```
root@staff_2:~# ifconfig
48-ETHZ: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 48.200.0.22  netmask 255.255.254.0  broadcast 0.0.0.0
        ether f6:03:a3:6f:dd:4e  txqueuelen 1000  (Ethernet)
        RX packets 532  bytes 37336 (36.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
```

Staff 3:48.200.0.32/23

```
root@staff_3:~# ifconfig
48-EPFL: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 48.200.0.32  netmask 255.255.254.0  broadcast 0.0.0.0
        ether a2:98:3e:c1:b3:8c  txqueuelen 1000  (Ethernet)
        RX packets 529  bytes 37126 (36.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
root@staff_3:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         48.200.1.1      0.0.0.0         UG        0 0          0 48-EPFL
48.200.0.0      0.0.0.0         255.255.254.0   U         0 0          0 48-EPFL
158.48.0.0      0.0.0.0         255.255.0.0     U         0 0          0 ssh
root@staff_3:~#
```

GENE router:

48.200.1.1/23

```
GENE_router# show inter brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
GENE-L2         up      default         48.200.1.1/23
GENE-L2.10      up      default
GENE-L2.20      up      default
GENE-L2.30      up      default
ext_50_LOND     up      default
lo              up      default
port_MIAM       up      default         48.0.9.1/24
port_PARI       up      default         48.0.3.2/24
ssh             up      default         158.48.13.1/16

GENE_router#
```

ZURI router:

48.200.1.2/23

```
ZURI_router# show inter brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
ZURI-L2         up      default         48.200.1.2/23
ZURI-L2.10      up      default
ZURI-L2.20      up      default
ZURI-L2.30      up      default
ext_45_ATLA     up      default
lo              up      default         48.152.0.1/24
measurement_48  up      default         48.0.199.1/24
port_LOND       up      default         48.0.2.1/24
port_PARI       up      default         48.0.1.1/24
ssh             up      default         158.48.11.1/16

ZURI_router#
```

Question 1.2

In this question, we were tasked with setting up virtual networks, in order to separate the students from the staff, with VLAN 10 (L2.10 interfaces) belonging to staff and VLAN 20 (L2.20 interfaces) belonging to students.

As we are going to see, the whole configuration of our L2 layer changed.

We chose the network 48.200.0.0/24 for the staff and 48.200.1.0/24 for the students.

The mask changed from /23 to /24 for all IPs, and that is because now we must keep static the 24th bit of our addresses, with 0 being for the staff and 1 being for the students, so that a clear distinction between the two networks is present.

New hosts' IPs:

- Staff 1: 48.200.0.1/24
- Student 1: 48.200.1.1/24
- Staff 2: 48.200.0.2/24
- Student 2: 48.200.1.2/24
- Staff 3: 48.200.0.3/24
- Student 3: 48.200.1.3/24

New GENE and ZURI configurations:

GENE

```
GENE_router# show inter brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
GENE-L2         up      default
GENE-L2.10      up      default         48.200.0.10/24
GENE-L2.20      up      default         48.200.1.10/24
GENE-L2.30      up      default
ext_50_LOND     up      default
lo              up      default
port_MIAM       up      default         48.0.9.1/24
port_PARI       up      default         48.0.3.2/24
ssh             up      default         158.48.13.1/16

GENE_router#
```

ZURI

```
ZURI_router# show inter brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
ZURI-L2         up      default
ZURI-L2.10      up      default         48.200.0.20/24
ZURI-L2.20      up      default         48.200.1.20/24
ZURI-L2.30      up      default
ZURI-LW         down    default
ext_45_ATLA     up      default
lo              up      default         48.152.0.1/24
measurement_48  up      default         48.0.199.1/24
port_LOND       up      default         48.0.2.1/24
port_PARI       up      default         48.0.1.1/24
ssh             up      default         158.48.11.1/16

ZURI_router#
```

was created by misstype, never configured

New CERN, EPFL and ETHZ configurations:

CERN

```
root@CERN:~# ovs-vsctl show
c8e93c82-616b-4874-95d8-775a460ca4d2
    Bridge "br0"
        fail_mode: standalone
        Port "48-staff_1"
            tag: 10
            Interface "48-staff_1"
        Port GENErouter
            Interface GENErouter
        Port "br0"
            Interface "br0"
                type: internal
        Port "48-EPFL"
            trunks: [10, 20]
            Interface "48-EPFL"
        Port "48-ETHZ"
            trunks: [10, 20]
            Interface "48-ETHZ"
        Port "48-student_1"
            tag: 20
            Interface "48-student_1"
        Port "48-vpn_1"
            Interface "48-vpn_1"
    ovs_version: "2.6.2"
root@CERN:~#
```

EPFL

```
root@EPFL:~# ovs-vsctl show
e8c462a0-7a75-4d5c-a601-8498ad91f094
    Bridge "br0"
        fail_mode: standalone
        Port "48-student_3"
            tag: 20
            Interface "48-student_3"
        Port "48-ETHZ"
            trunks: [10, 20]
            Interface "48-ETHZ"
        Port "48-vpn_3"
            Interface "48-vpn_3"
        Port "br0"
            Interface "br0"
                type: internal
        Port "48-CERN"
            trunks: [10, 20]
            Interface "48-CERN"
        Port "48-staff_3"
            tag: 10
            Interface "48-staff_3"
    ovs_version: "2.6.2"
root@EPFL:~#
```

ETHZ

```
root@ETHZ:~# ovs-vsctl show
74f986b5-dac1-4c80-aac3-8a5d1e09b1d0
    Bridge "br0"
        fail_mode: standalone
        Port ZURIrouter
            Interface ZURIrouter
        Port "48-CERN"
            trunks: [10, 20]
            Interface "48-CERN"
        Port "48-EPFL"
            trunks: [10, 20]
            Interface "48-EPFL"
        Port "48-student_2"
            tag: 20
            Interface "48-student_2"
        Port "48-staff_2"
            tag: 10
            Interface "48-staff_2"
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.6.2"
root@ETHZ:~#
```

With student 1 and 3 having as their default gateway 48.200.1.10, which is the IP of the GENE-L2.20 interface.

Staff 1 and 3 have their default gateway set to 48.200.0.10, which is the IP of the GENE-L2.10 interface.

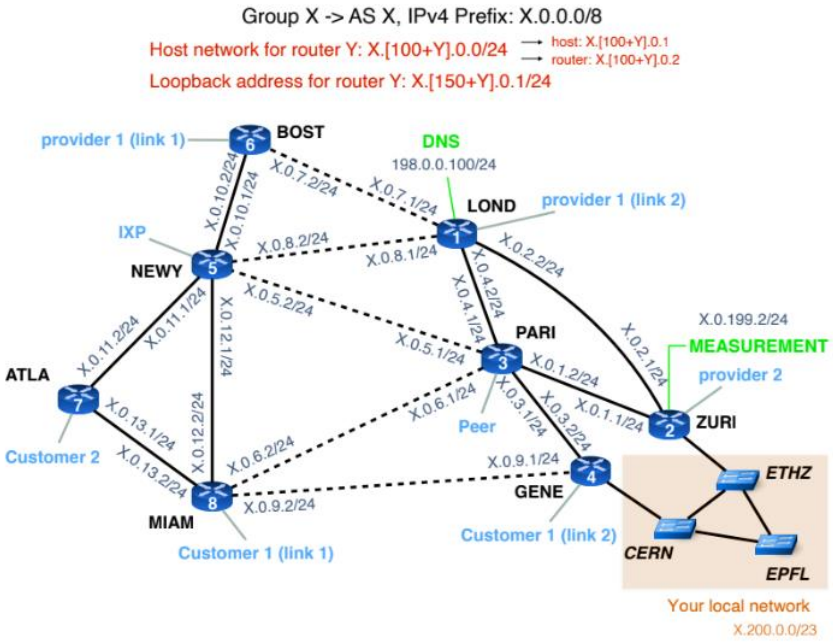
Lastly, staff and student 2 have their default gateways set to 48.200.0.20(ZURI-L2.10 interface) and 48.200.1.20(ZURI-L2.20 interface) respectively.

As already mentioned, no ping/traceroute commands are operational, so, no results can be shown.

Although, if we did a traceroute from student 3 to staff 3 (both connected to EPFL switch), we would observe a hop to GENE router, since, they are not connected through the L2 layer anymore, the traffic needs to be sent to the router first, in order to determine where it needs to be headed (all this happens because we successfully separated the staff from the students).

If we also did a traceroute from student 3 (EPFL) to student 2 (ETHZ), we would not observe a hop, since they are directly connected through L2 (again, due to the VLAN configurations), and traceroute does not show the switches it “hops” over.

Question 1.3



Here, we are tasked with configuring the whole network.

First, we assign IPs to each router (BOST, LOND, etc.) and their respective ports, then we assign IPs and default gateways to each router's host and lastly, connect each router with its "neighbors" (via the IPs shown on the links between them).

Since there is connectivity between routers and their hosts, we can now start "advertising" each network via the OSPF protocol.

After this is done, the connection between all routers/host is now complete.

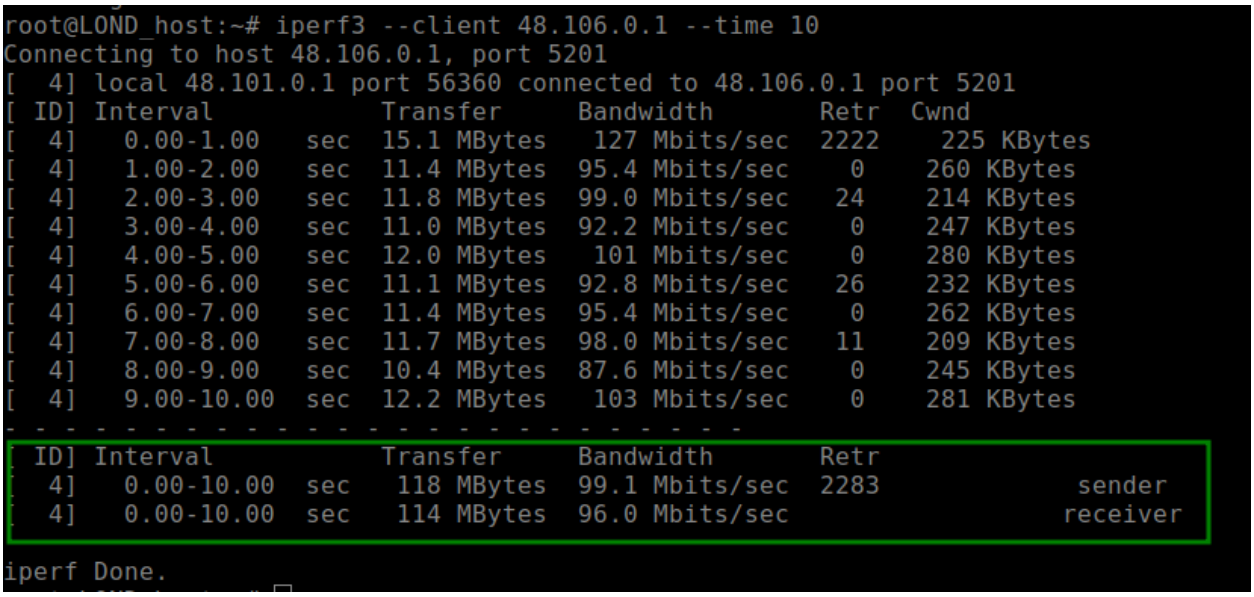
Here is an example of a traceroute from PARI host to LOND host:

```
File  Edit  View  Search  Terminal  Help
root@PARI_host:~# traceroute 48.107.0.1
traceroute to 48.107.0.1 (48.107.0.1), 30 hops max, 60 byte packets
 1  PARI-host.group48 (48.103.0.2)  0.434 ms  0.282 ms  0.372 ms
 2  NEWY-PARI.group48 (48.0.5.2)  2.221 ms MIAM-PARI.group48 (48.0.6.2)  0.245 ms NEWY-PARI.gr
oup48 (48.0.5.2)  2.078 ms
 3  ATLA-NEWY.group48 (48.0.11.2)  2.407 ms  2.330 ms  2.318 ms
 4  host-ATLA.group48 (48.107.0.1)  2.479 ms  1.664 ms  1.670 ms
root@PARI_host:~#
```

Question 1.4

So, to find our network's configuration, we had to run some speed metrics using iperf3.

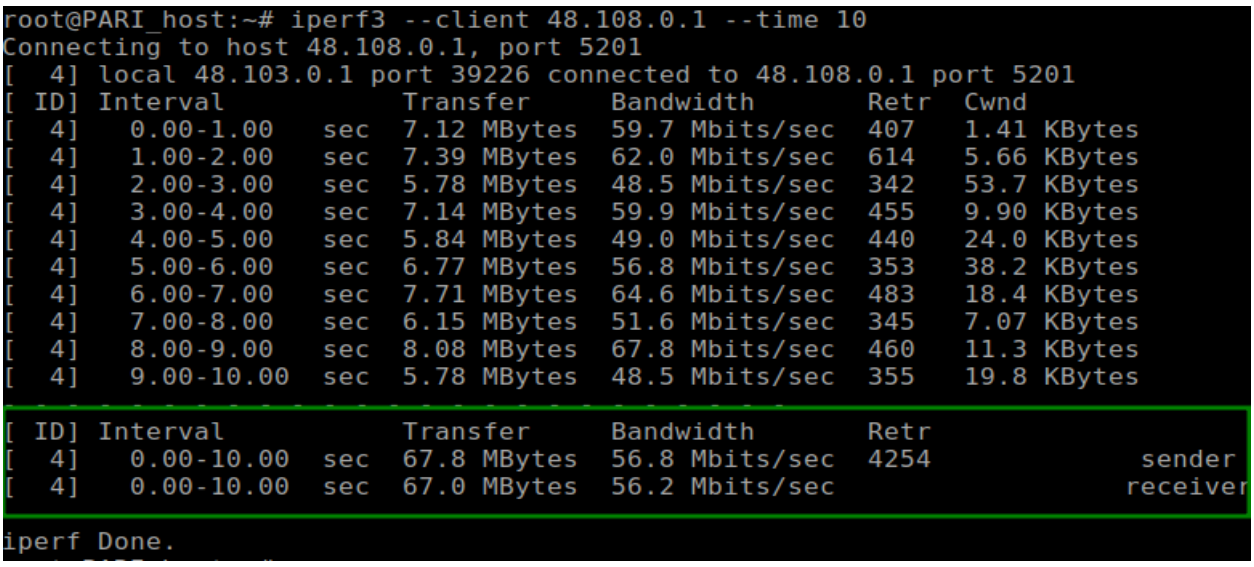
Metric from BOST host to LOND host:



```
root@LOND_host:~# iperf3 --client 48.106.0.1 --time 10
Connecting to host 48.106.0.1, port 5201
[  4] local 48.101.0.1 port 56360 connected to 48.106.0.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  15.1 MBytes   127 Mbits/sec  2222    225 KBytes
[  4]   1.00-2.00   sec  11.4 MBytes  95.4 Mbits/sec    0    260 KBytes
[  4]   2.00-3.00   sec  11.8 MBytes  99.0 Mbits/sec   24    214 KBytes
[  4]   3.00-4.00   sec  11.0 MBytes  92.2 Mbits/sec    0    247 KBytes
[  4]   4.00-5.00   sec  12.0 MBytes   101 Mbits/sec    0    280 KBytes
[  4]   5.00-6.00   sec  11.1 MBytes  92.8 Mbits/sec   26    232 KBytes
[  4]   6.00-7.00   sec  11.4 MBytes  95.4 Mbits/sec    0    262 KBytes
[  4]   7.00-8.00   sec  11.7 MBytes  98.0 Mbits/sec   11    209 KBytes
[  4]   8.00-9.00   sec  10.4 MBytes  87.6 Mbits/sec    0    245 KBytes
[  4]   9.00-10.00  sec  12.2 MBytes   103 Mbits/sec    0    281 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec   118 MBytes  99.1 Mbits/sec  2283             sender
[  4]   0.00-10.00  sec   114 MBytes  96.0 Mbits/sec                   receiver

iperf Done.
```

As we can see, we have a high bandwidth, ~98Mbits/sec.

Metric from MIAM host to PARI host:



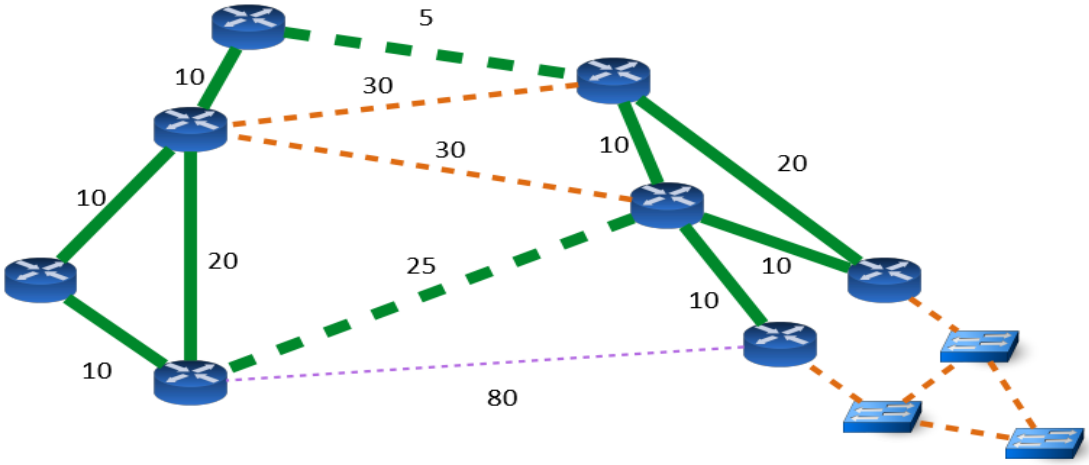
```
root@PARI_host:~# iperf3 --client 48.108.0.1 --time 10
Connecting to host 48.108.0.1, port 5201
[  4] local 48.103.0.1 port 39226 connected to 48.108.0.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  7.12 MBytes  59.7 Mbits/sec  407   1.41 KBytes
[  4]   1.00-2.00   sec  7.39 MBytes  62.0 Mbits/sec  614   5.66 KBytes
[  4]   2.00-3.00   sec  5.78 MBytes  48.5 Mbits/sec  342   53.7 KBytes
[  4]   3.00-4.00   sec  7.14 MBytes  59.9 Mbits/sec  455   9.90 KBytes
[  4]   4.00-5.00   sec  5.84 MBytes  49.0 Mbits/sec  440   24.0 KBytes
[  4]   5.00-6.00   sec  6.77 MBytes  56.8 Mbits/sec  353   38.2 KBytes
[  4]   6.00-7.00   sec  7.71 MBytes  64.6 Mbits/sec  483   18.4 KBytes
[  4]   7.00-8.00   sec  6.15 MBytes  51.6 Mbits/sec  345   7.07 KBytes
[  4]   8.00-9.00   sec  8.08 MBytes  67.8 Mbits/sec  460   11.3 KBytes
[  4]   9.00-10.00  sec  5.78 MBytes  48.5 Mbits/sec  355   19.8 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  67.8 MBytes  56.8 Mbits/sec  4254             sender
[  4]   0.00-10.00  sec  67.0 MBytes  56.2 Mbits/sec                   receiver

iperf Done.
```

Here also, we have a high bandwidth connection, ~67Mbits/sec.

Since our two high bandwidth connections are these, our network is of Configuration 1.

Below is the Configuration 1, accompanied by the custom OSPF weights we set, in order to balance traffic between ATLA-NEWY-MIAM and LOND-PARI-ZURI, as well as the other weight adjustments we had to do (all the defaults weights were 10).

Configuration 1



We also did some traceroutes to test if the adjustments were correct.

Here is a traceroute from ATLA host to ZURI loopback inter before the load balancing:

```
root@ATLA_host:~# traceroute 48.152.0.1
traceroute to 48.152.0.1 (48.152.0.1), 30 hops max, 60 byte packets
 1  ATLA-host.group48 (48.107.0.2)  0.530 ms  0.395 ms  0.417 ms
 2  MIAM-ATLA.group48 (48.0.13.2)  0.769 ms NEWY-ATLA.group48 (48.0.11.1)  0.765 ms MIAM-ATLA.group48 (48.0.13.2)  1.022 ms
 3  PARI-MIAM.group48 (48.0.6.1)  1.006 ms PARI-NEWY.group48 (48.0.5.1)  3.196 ms  2.739 ms
 4  48.152.0.1 (48.152.0.1)  2.239 ms  2.217 ms  2.092 ms
```

And here is the after:

```
root@ATLA_host:~# traceroute 48.152.0.1
traceroute to 48.152.0.1 (48.152.0.1), 30 hops max, 60 byte packets
 1  ATLA-host.group48 (48.107.0.2)  0.520 ms  0.337 ms  0.331 ms
 2  NEWY-ATLA.group48 (48.0.11.1)  0.712 ms MIAM-ATLA.group48 (48.0.13.2)  0.788 ms  1.013 ms
 3  BOST-NEWY.group48 (48.0.10.2)  0.751 ms PARI-MIAM.group48 (48.0.6.1)  1.367 ms  1.161 ms
 4  LOND-BOST.group48 (48.0.7.1)  20.828 ms  20.744 ms  20.719 ms
 5  48.152.0.1 (48.152.0.1)  11.843 ms  12.070 ms  11.670 ms
root@ATLA_host:~# traceroute 48.152.0.1
traceroute to 48.152.0.1 (48.152.0.1), 30 hops max, 60 byte packets
 1  ATLA-host.group48 (48.107.0.2)  4.091 ms  1.742 ms  1.774 ms
 2  NEWY-ATLA.group48 (48.0.11.1)  2.191 ms MIAM-ATLA.group48 (48.0.13.2)  2.271 ms  2.251 ms
 3  PARI-MIAM.group48 (48.0.6.1)  2.424 ms BOST-NEWY.group48 (48.0.10.2)  2.335 ms PARI-MIAM.group48 (48.0.6.1)  2.720 ms
 4  48.152.0.1 (48.152.0.1)  14.013 ms LOND-BOST.group48 (48.0.7.1)  22.857 ms 48.152.0.1 (48.152.0.1)  14.008 ms
root@ATLA_host:~#
```

As we can see, we have two different paths. This is because both of them have the same “cost”, so the route gets chosen randomly each time.

Question 1.5

ATLA and NEWY hosts are directly connected, through ATLA-NEWY route, because this path has the lowest “cost” (10). We were asked, for security reasons, all traffic from ATLA host towards NEWY host to be passed through the MIAM router, and vice versa. We achieved that by creating static routes.

Here is a traceroute from ATLA host to NEWY host.

```
root@ATLA_host:~# traceroute 48.105.0.1
traceroute to 48.105.0.1 (48.105.0.1), 30 hops max, 60 byte packets
 1  ATLA-host.group48 (48.107.0.2)  0.523 ms  0.313 ms  0.353 ms
 2  NEWY-ATLA.group48 (48.0.11.1)  0.726 ms  0.933 ms  0.920 ms
 3  host-NEWY.group48 (48.105.0.1)  0.655 ms  0.660 ms  0.600 ms
root@ATLA_host:~#
```

As we can see, they are directly linked, so there’s no “hop” to MIAM router.

And here is after configuring static routes for this purpose:

```
root@ATLA_host:~# traceroute 48.105.0.1
traceroute to 48.105.0.1 (48.105.0.1), 30 hops max, 60 byte packets
 1  ATLA-host.group48 (48.107.0.2)  1.565 ms  1.275 ms  1.248 ms
 2  MIAM-ATLA.group48 (48.0.13.2)  1.724 ms  1.805 ms  1.792 ms
 3  NEWY-MIAM.group48 (48.0.12.1)  1.918 ms  1.874 ms  1.860 ms
 4  host-NEWY.group48 (48.105.0.1)  2.982 ms  2.876 ms  2.931 ms
root@ATLA_host:~#
```

As we can see, the wanted redirection of traffic is achieved.

```
  *                         via 48.0.13.1, port_ATLA, 1d20h32m
O>* 48.103.0.0/24 [110/35] via 48.0.6.1, port_PARI, 1d20h32m
S>* 48.105.0.0/24 [1/0] via 48.0.12.1, port_NEWY, 1d18h51m
O   48.105.0.0/24 [110/30] via 48.0.12.1, port_NEWY, 1d21h09m
                           via 48.0.13.1, port_ATLA, 1d21h09m
O>* 48.106.0.0/24 [110/40] via 48.0.12.1, port_NEWY, 1d20h32m
  *                        via 48.0.13.1, port_ATLA, 1d20h32m
S>* 48.107.0.0/24 [1/0] via 48.0.13.1, port_ATLA, 00:03:50
O   48.107.0.0/24 [110/20] via 48.0.13.1, port_ATLA, 2d22h33m
O   48.108.0.0/24 [110/10] is directly connected, host, 2d22h36m
C>* 48.108.0.0/24 is directly connected, host, 2d23h28m
O>* 48.152.0.1/32 [110/35] via 48.0.6.1, port_PARI, 1d20h32m
C>* 48.158.0.0/24 is directly connected, lo, 2d23h28m
C>* 158.48.0.0/16 is directly connected, ssh, 03w1d05h
O>* 198.0.0.0/24 [110/45] via 48.0.6.1, port_PARI, 1d20h32m
  *                       via 48.0.12.1, port_NEWY, 1d20h32m
  *                       via 48.0.13.1, port_ATLA, 1d20h32m
MIAM_router#
```

Also, here is the “show ip route” result, showcasing that, no matter what, the paths to subnets 48.105.0.0/24(NEWY host) and 48.107.0.0/24 (ATLA host) are of cost 0, so they will always be preferred.

Project Phase Two

Question 2.1

```
ATLA_router# show ip bgp summary

IPv4 Unicast Summary:
BGP router identifier 48.157.0.1, local AS number 48 vrf-id 0
BGP table version 3
RIB entries 3, using 552 bytes of memory
Peers 7, using 143 KiB of memory

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
48.151.0.1      4         48      14      13        0    0    0 00:09:37            2
48.152.0.1      4         48       4       4        0    0    0 00:02:00            0
48.153.0.1      4         48      12      13        0    0    0 00:09:32            0
48.154.0.1      4         48      12      10        0    0    0 00:04:41            0
48.155.0.1      4         48      12      13        0    0    0 00:09:26            0
48.156.0.1      4         48      12      13        0    0    0 00:09:24            0
48.158.0.1      4         48      13      12        0    0    0 00:09:18            1

Total number of neighbors 7
ATLA_router#
```

Here above are the results of a “show ip bgp summary” on ATLA router. As we can see, the iBGP has been set up correctly, which is verified by the last column, where every “neighbor” has >= 0 connections, meaning that connectivity is present throughout our AS.

As for the “update-source” command, it is used to specify from which interface the BGP updates would be passed through. So, when we use “update-source lo”, we are telling the router that, all BGP updates will be delivered to and from the loopback interface, which is always active, even if the whole network is down.

Question 2.2

The command “next-hop-self” is used to “regulate” the outcoming advertisements of our network, that is done through the specific router that this command is used in. More specifically, when its used, the router broadcasts internally to our other routers, that, whenever we want to reach a network that this router is connected to, it is specifically going to be used over other routers. Here is a “show run” snippet of our ATLA router:

```
address-family ipv4 unicast
 network 48.0.0.0/8
 neighbor 48.151.0.1 next-hop-self
 neighbor 48.152.0.1 next-hop-self
 neighbor 48.153.0.1 next-hop-self
 neighbor 48.154.0.1 next-hop-self
 neighbor 48.155.0.1 next-hop-self
 neighbor 48.156.0.1 next-hop-self
 neighbor 48.158.0.1 next-hop-self
```

Here, it is set that, when we want to reach an outside network(that is “learned” from ATLA router in this example) from our other routers, the “message” first has to go through ATLA router, and then reach the outside network.

Here is the result of a “show ip bgp” from PARI router:

```
PARI_router# show ip bgp
BGP table version is 20, local router ID is 48.153.0.1, vrf id 0
Default local pref 100, local AS 48
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
*>i45.0.0.0/8       48.152.0.1               0    100      0 45 i
*=i46.0.0.0/8       48.155.0.1               0    100      0 46 i
*>i                 48.151.0.1                    100      0 46 i
* i48.0.0.0/8       48.152.0.1               0    100      0 i
* i                 48.154.0.1               0    100      0 i
* i                 48.156.0.1               0    100      0 i
* i                 48.155.0.1               0    100      0 i
*>                  0.0.0.0                  0         32768 i
* i                 48.157.0.1               0    100      0 i
* i                 48.158.0.1               0    100      0 i
* i                 48.151.0.1               0    100      0 i
*>i53.0.0.0/8       48.152.0.1                    100      0 45 68 70 53 i
*>i61.0.0.0/8       48.152.0.1                    100      0 45 62 64 61 i
*>i62.0.0.0/8       48.152.0.1                    100      0 45 62 i
*>i64.0.0.0/8       48.152.0.1                    100      0 45 64 i
*>i65.0.0.0/8       48.152.0.1                    100      0 45 62 64 65 i
*>i66.0.0.0/8       48.152.0.1                    100      0 45 62 64 66 i
*>i68.0.0.0/8       48.152.0.1                    100      0 45 68 i
*>i69.0.0.0/8       48.152.0.1                    100      0 45 68 69 i
*>i70.0.0.0/8       48.152.0.1                    100      0 45 70 i
*>i71.0.0.0/8       48.152.0.1                    100      0 45 68 69 71 i
*>i84.0.0.0/8       48.152.0.1                    100      0 45 68 69 84 i
*>i92.0.0.0/8       48.152.0.1                    100      0 45 68 69 92 i
```

As we can see, here are the prefixes we have “learned” through eBGP.

Here is also a looking-glass snippet which showcases that our network has been correctly advertised(it is an older looking glass snippet from another AS, since it is not currently working, but it demonstrates exactly what we want) :

```
Database query script, trigger timestamp --> ****2024-05-20 18:06:41****

WARNING:  could not create relation-cache initialization file "global/pg_internal.init.989816": No s
DETAIL:  Continuing anyway, but there's something wrong.
WARNING:  could not create relation-cache initialization file "base/16384/pg_internal.init.989816":
ice
DETAIL:  Continuing anyway, but there's something wrong.
92-ZURI
BGP table version is 71, local router ID is 92.152.0.1, vrf id 0
Default local pref 100, local AS 92
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
*> 45.0.0.0/8       180.124.0.61                50      0 61 64 62 45 i
*> 46.0.0.0/8       180.124.0.61                50      0 61 64 62 45 48 46 i
*> 48.0.0.0/8       180.124.0.61                50      0 61 64 62 45 48 i
*> 53.0.0.0/8       180.124.0.61                50      0 61 64 62 53 i
*> 61.0.0.0/8       180.124.0.61             0  50      0 61 i
*> 62.0.0.0/8       180.124.0.61                50      0 61 64 62 i
*> 64.0.0.0/8       180.124.0.61                50      0 61 64 i
*> 65.0.0.0/8       180.124.0.65             0  50      0 65 i
*> 66.0.0.0/8       180.124.0.61                50      0 61 64 66 i
*> 68.0.0.0/8       180.124.0.61                50      0 61 64 62 68 i
*> 69.0.0.0/8       180.124.0.61                50      0 61 64 62 68 69 i
*> 70.0.0.0/8       180.124.0.61                50      0 61 64 62 70 i
*> 71.0.0.0/8       180.124.0.71             0  50      0 71 i
*> 84.0.0.0/8       180.124.0.84             0  50      0 84 i
*> 92.0.0.0/8       0.0.0.0                  0         32768 i
```

And lastly, here is a traceroute to another AS:

```
PARI_router# traceroute 65.103.0.1
traceroute to 65.103.0.1 (65.103.0.1), 64 hops max
  1   48.0.1.1  5.267ms  0.902ms  0.638ms
  2   179.1.28.1  5.366ms  3.021ms  2.569ms
  3   45.0.11.1  5.654ms  2.977ms  2.988ms
  4   180.123.0.62  16.508ms  7.621ms  17.025ms
  5   179.0.31.2  16.473ms  8.957ms  8.911ms
  6   180.123.0.64  19.795ms  8.538ms  8.123ms
  7   64.0.13.1  22.347ms  9.471ms  9.257ms
  8   65.0.1.1  19.848ms  11.688ms  11.426ms
  9   179.1.64.1  11.957ms  11.301ms  11.730ms
 10   65.103.0.1  18.621ms  13.843ms  11.309ms
PARI router#
```

Question 2.3

```
NEWY_router# conf t
NEWY_router(config)# ip prefix-list OWN_PREFIX seq 5 permit 48.0.0.0/8
```

- “ip prefix-list OWN_PREFIX” : Using this command, we create a prefix list named OWN_PREFIX(we could have used any other name we wanted).
- “seq 5”: We assign the sequence number 5 on this list. Sequence number determines the order in which entries are processed.
- “permit 48.0.0.0/8”: We permit this IP range to be advertised or accepted.

```
NEWY_router(config)# route-map IXP_OUT permit 10
NEWY_router(config-route-map)# match ip address prefix-list OWN_PREFIX
NEWY_router(config-route-map)# set community 122:21 122:23 122:25 122:27 122:29 122:31 122:33 122:42 122:44 122:46 122:48 122:50 122:52 122:54
NEWY_router(config-route-map)#
```

- “route-map IXP_OUT permit 10” : Here, we create a new route map named IXP_OUT, and give the “priority” of 10 (more like position in the queue? Something of this sort).

- “match ip address prefix-list OWN_PREFIX”: This line specifies that the route map should apply to routes that match the OWN_PREFIX list (IPs in the range of 48.0.0.0/8)

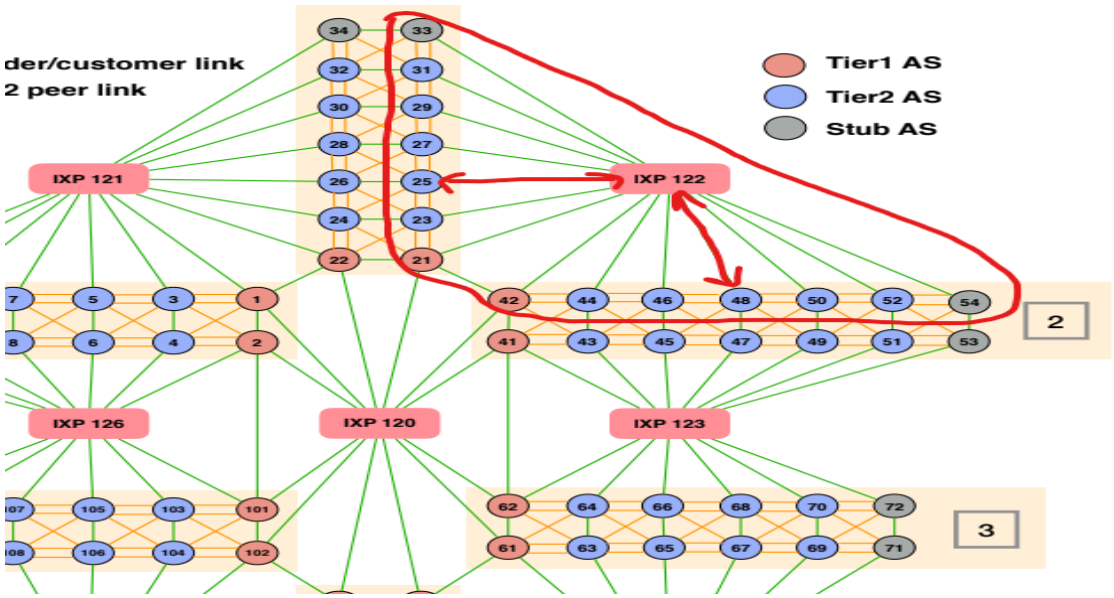
- “set community 122:21 122:23 122:25 … “: This command creates a “community” between these ASes and the IXP, in which all are connected to it. The community is a way of tagging routes with additional information, and more specifically, routing decisions(as demonstrated later, we “force” these ASes to communicate exclusively through the IXP)

```
NEWY_router# conf t
NEWY_router(config)# router bgp 48
NEWY_router(config-router)# neighbor 180.122.0.122 route-map IXP_OUT out
NEWY_router(config-router)#
```

And here, lastly, we apply the route map to our neighboring IXP, specifically for outbound (...IXP_OUT **out**) traffic and advertisements.

After this is done, here is the result of a traceroute from AS 25 to our AS (48), that shows the traffic is being passed through our IXP:

```
root@5072cf122265:~# ./launch_traceroute.sh 25 48.103.0.1
Hop 1:  25.0.199.1 TTL=0 during transit
Hop 2:  25.0.2.2 TTL=0 during transit
Hop 3:  25.0.8.2 TTL=0 during transit
Hop 4:  180.122.0.48 TTL=0 during transit
Hop 5:  48.0.5.1 TTL=0 during transit
Hop 6:  48.103.0.1 Echo reply (type=0/code=0)
Hop 7:  48.103.0.1 Echo reply (type=0/code=0)
Hop 8:  ^CHop 9:  ^CHop 10:  ^C
root@5072cf122265:~# ^C
root@5072cf122265:~# ^C
root@5072cf122265:~#
```



Question 2.4

Both in and out route-maps are from the setting up of the NEYW(IXP) router, all satisfying the Gao-Rexford Rules:

"GUIDELINES":

	Community	LOCAL_PREF
Provider:	30	20
Customer:	10	100
Peer:	20	50

IN:

Here, we create a new route-map for the IXP called "IXP_IN". We set the community to 20 based on our guidelines and then we set the preference value to 50

```
NEWY_router# conf t
NEWY_router(config)# route-map IXP_IN permit 10
NEWY_router(config-route-map)# set community 48:20
NEWY_router(config-route-map)# set local-preference 50
NEWY_router(config-route-map)# exit
```

Some commands that follow the ones above but we forgot to take screenshots of:

"ip prefix-list OWN-PREFIX seq 5 permit 48.0.0.0/8": create a prefix-list of our subnet

"bgp community-list 1 permit 48:10": create a community-list that matches outbound traffic with tag 10 (customer).

OUT:

Here, we create a new route-map for the IXP called “IXP_out”.

```
NEWY_router(config)# route-map IXP_OUT permit 10
NEWY_router(config-route-map)# match ip address prefix-list OWN-PREFIX
NEWY_router(config-route-map)# set community 122:21 122:23 122:25 122:27 122:29 122:31 122:33 122:42 122:44 122:46 122:48 122:50 122:52 122:54
NEWY_router(config-route-map)# EXIT
% Unknown command: EXIT
NEWY_router(config-route-map)# exit
```

Here, we check that if our ip address is in the prefix-list, announce that prefix in the entire community(set community 122:12 ...) that is connected through the IXP.

Question 2.5

In this last question, we were tasked with creating our own monitoring tool using python. Here below follow the screenshots of the results of running this tool, with LOND-host being the client and ATLA-host, BOST-host and MIAM-host being the servers:

ATLA-host:

```
root@ATLA_host:~/quest2_5# python3 server.py 48.107.0.1 5000 2048
```

BOST-host:

```
root@BOST_host:~# cd quest2_5/
root@BOST_host:~/quest2_5# python3 server.py 48.106.0.1 5000 2048
```

MIAM-host:

```
root@MIAM_host:~/quest2_5# ls
client.py  config.txt  server.py
root@MIAM_host:~/quest2_5# python3 server.py 48.108.0.1 5000 2048
```

And here are the results:

```
root@LOND_host:~/quest2_5# python3 client.py ATLA-host,BOST-host,MIAM-host 5000
TCP target IP: 48.107.0.1
TCP target PORT: 5000
message: 48.106.0.1 48.108.0.1
<class 'str'>
creating socket...
connecting to ip: 48.107.0.1 ...
sending data...
waiting to receive data...
received data = 48.107.0.1,48.106.0.1 0.0 4 48.107.0.2,48.0.11.1,48.0.10.2,48.106.0.1
48.107.0.1,48.108.0.1 0.0 3 48.107.0.2,48.0.13.2,48.108.0.1

['48.107.0.2', '48.0.11.1', '48.0.10.2', '48.106.0.1']
['48.107.0.2', '48.0.13.2', '48.108.0.1']
|----- Hosts -----|---- Latency ----|---- No of hops -----|---- Path ----|
 BOST-host ATLA-host-    0.0             4      ATLA-router-NEWY-router-BOST-router-BOST-host-
 MIAM-host ATLA-host-    0.0             3      ATLA-router-MIAM-router-MIAM-host-

closing socket ...
TCP target IP: 48.106.0.1
TCP target PORT: 5000
message: 48.107.0.1 48.108.0.1
<class 'str'>
creating socket...
connecting to ip: 48.106.0.1 ...
sending data...
waiting to receive data...
received data = 48.106.0.1,48.107.0.1 0.0 4 48.106.0.2,48.0.10.1,48.0.11.2,48.107.0.1
48.106.0.1,48.108.0.1 0.0 4 48.106.0.2,48.0.10.1,48.0.12.2,48.108.0.1

['48.106.0.2', '48.0.10.1', '48.0.11.2', '48.107.0.1']
['48.106.0.2', '48.0.10.1', '48.0.12.2', '48.108.0.1']
|----- Hosts -----|---- Latency ----|---- No of hops -----|---- Path ----|
 BOST-host-ATLA-host     0.0             4      BOST-router-NEWY-router-ATLA-router-ATLA-host-
 MIAM-host BOST-host-    0.0             4      BOST-router-NEWY-router-MIAM-router-MIAM-host-

closing socket ...
TCP target IP: 48.108.0.1
TCP target PORT: 5000
message: 48.107.0.1 48.106.0.1
<class 'str'>
creating socket...
connecting to ip: 48.108.0.1 ...
sending data...
waiting to receive data...
received data = 48.108.0.1,48.107.0.1 0.0 3 48.108.0.2,48.0.13.1,48.107.0.1
48.108.0.1,48.106.0.1 0.0 4 48.108.0.2,48.0.12.1,48.0.10.2,48.106.0.1

['48.108.0.2', '48.0.13.1', '48.107.0.1']
['48.108.0.2', '48.0.12.1', '48.0.10.2', '48.106.0.1']
|----- Hosts -----|---- Latency ----|---- No of hops -----|---- Path ----|
 MIAM-host-ATLA-host     0.0             3      MIAM-router-ATLA-router-ATLA-host-
 MIAM-host-BOST-host     0.0             4      MIAM-router-NEWY-router-BOST-router-BOST-host-

closing socket ...
root@LOND_host:~/quest2_5#
```

- Note, the names shown at Hosts column are sometimes listed in the opposite order of their occurrence. The names that have a “-“ at the end are meant to be shown first. Format: IP_from-IP_to