ABDULLAH DEDEOĞLU

BGP EXPLORATION AND ATTACK LAB RAPORT

Docker Installation:

Installed docker in the required file with *dcbuild* and *dcup* commands and started bird.

The meanings of these commands were learned.

1. **dcbuild (Alias for: docker-compose build):**
   - This command is a shortened version of **docker-compose build**.
   - When used, Docker creates images of services specified by Compose.
2. **dcup (Alias for: docker-compose up):**
   - This command is a shortened version of **docker-compose up**.
   - When used, it starts the services specified by Docker Compose and stands up containers.
3. **dcdown (Alias for: docker-compose down):**
   - This command is a shortened version of **docker-compose down**.
   - When used, it stops the services specified by Docker Compose and removes the associated containers.

Thanks to Docker, our containers that will work as internet emulators have been activated.

I also verified that my commands were executed with the *dockps* command.

After all this, the emulator started.

Network Map

As a result of Docker and Bird installation, the network map where all routers and hosts work as a container was displayed.

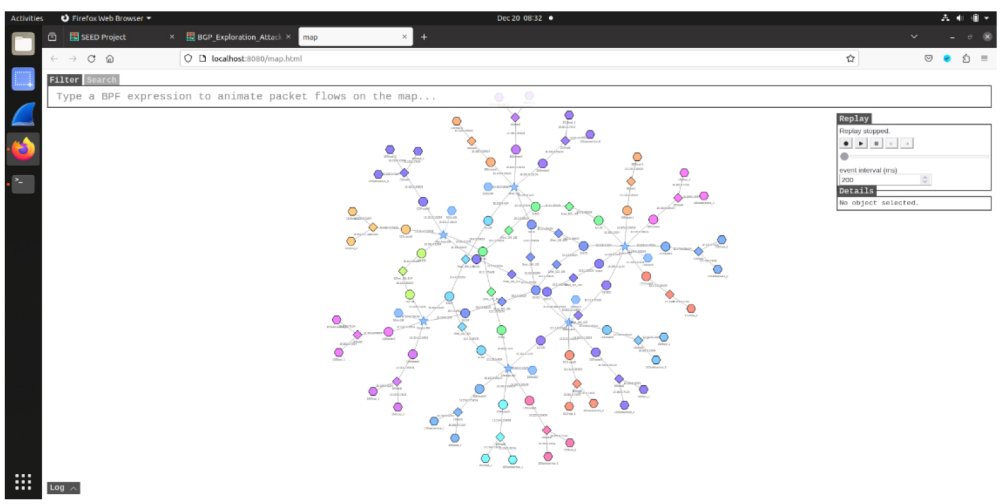


Figure 1 Network Map

Editing BGP Configuration Files

AS-180 File was selected to modify the BGP configuration file in parallel with the lab.

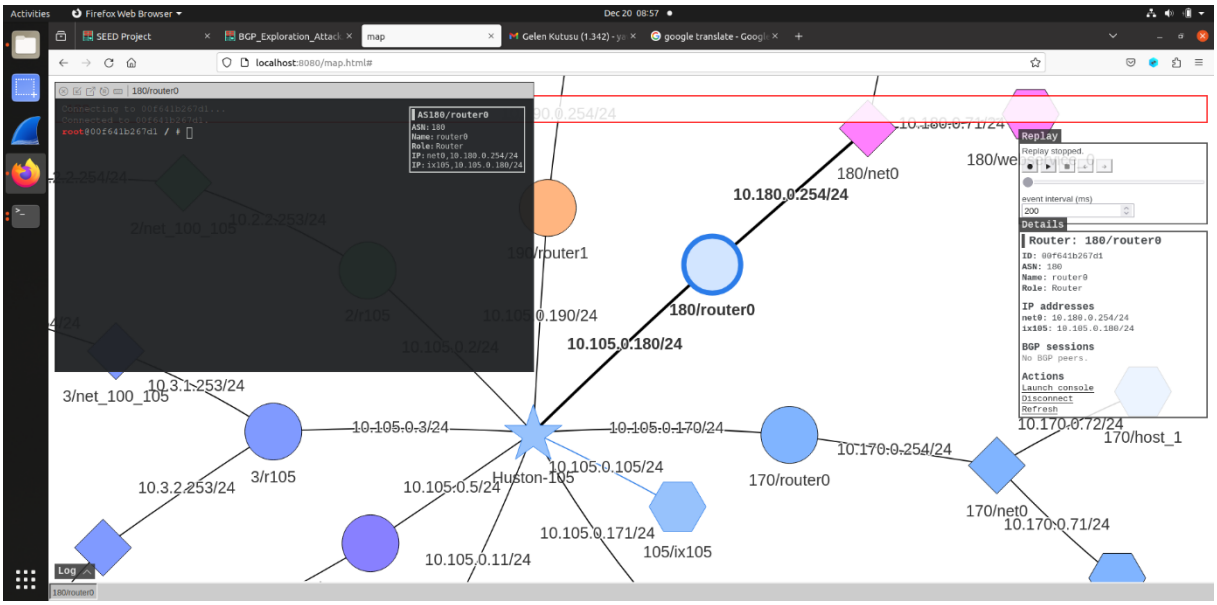


Figure 2 Representation of AS-180 Container on the Network Map

The IP of the AS-180 container was found with the Grep command.

```
seed@VM: ~/Desktop
[12/20/23]seed@VM:~/Desktop$ dockps | grep 180
0d049de33b4c  as180h-host_1-10.180.0.72
fd545a0fa540  as180h-webservice_0-10.180.0.71
00f641b267d1  as180r-router0-10.180.0.254
```

Configuration files inside the Container copied to the main host

[12/20/23]seed@VM:~/Desktop$ docker cp 00f6:/etc/bird/bird.conf ./as180_bird.conf

By editing the BGP configuration file with a text editor or other editor on the host computer, we simulated how interventions could be made in real-world scenarios.

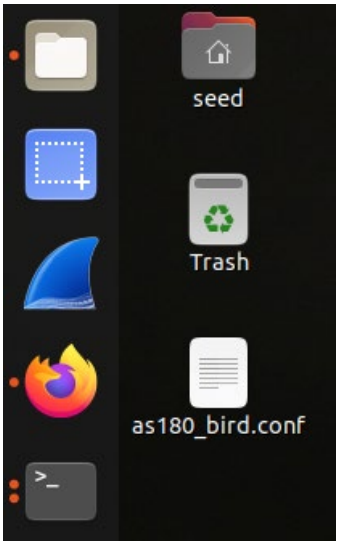


Figure 3 Configuration File Copied to Host Machine

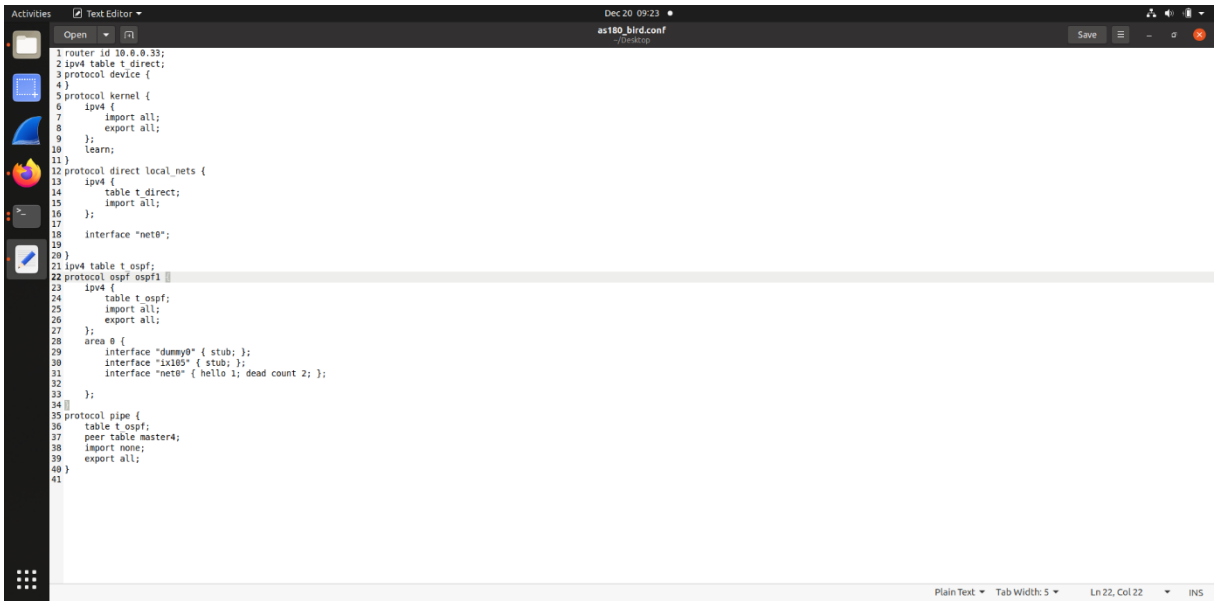


Figure 4 AS-180 Configuration File

Changes were copied back to the container, allowing updates to the configuration file to be applied effectively.

```
[12/20/23]seed@VM:~/Desktop$ docker cp ./as180_bird.conf 00f6:/etc/bird/bird.conf
```

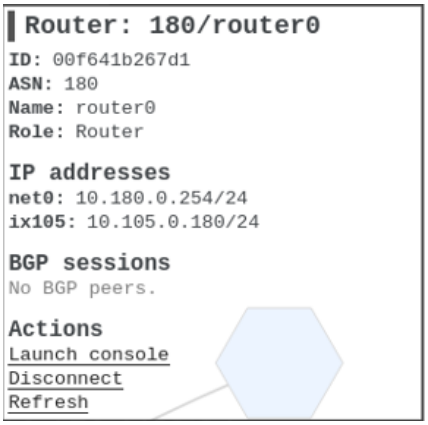
The configuration in the container was reloaded, making the BGP router's changes in the container effective immediately.

```
[12/20/23]seed@VM:~/Desktop$ docker exec 00f6 birdc configure
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured
[12/20/23]seed@VM:~/Desktop$
```

Conventions Used in the Emulator

In order to make it easier to identify the role of each node in the emulator, a set of conventions for assigning various numbers to nodes was created.

!!!! These conventions are for the emulator only and do not apply in the real world.



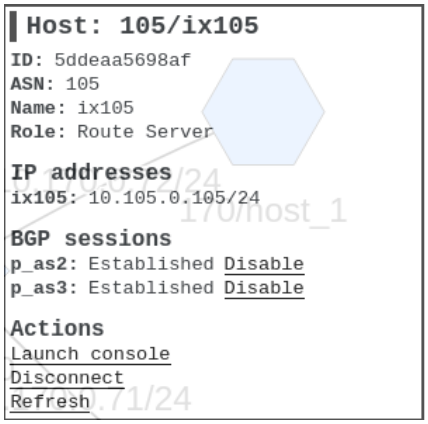


Figure 5 Conventions for Router 180 Figure 6 Conventions for Host 105

Task 1: Stub Autonomous System

> *"An autonomous system (AS) is a collection of connected Internet Protocol (IP) routing prefixes controlled by one or more network operators on behalf of a single administrative unit or domain. It is a basic unit in BGP. A connectionless AS is a type of AS that does not provide transit service to others. Most end users, universities, organizations, and most companies are connectionless ASes. Another type of AS is called a transit AS. These provide transit services to other ASes and are Internet service providers." **

In this task, we will focus on connectionless ASes and see how these ASes match with others. For these AS types, an overview of how BGP works will be obtained.

Task 1.a: Understanding BGP Configuration of AS-155

AS-155 was chosen to be parallel to the laboratory.

> *"AS-155 is a connectionless AS that has a network (10.155.0.0/24) and a BGP router (10.155.0.254)." **

For the understanding of the lab, the BGP configuration of the container of the AS-155 Router was analyzed

```
[12/20/23]seed@VM:~/Desktop$ dockps | grep 155
e7b66d7e86f0  as155h-host_0-10.155.0.71
c9f4d3e46612  as155h-webservice_1-10.155.0.72
e2204a1fc1c0  as155r-router0-10.155.0.254
[12/20/23]seed@VM:~/Desktop$ docker cp e220:/etc/bird/bird.conf ./as155_bird.con
f
[12/20/23]seed@VM:~/Desktop$ docker cp ./as155_bird.conf e220:/etc/bird/bird.con
f
[12/20/23]seed@VM:~/Desktop$ docker exec e220 birdc configure
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured
[12/20/23]seed@VM:~/Desktop$
```

Figure 7 Copying and running the container files of the AS-155 Router (learned in previous steps)

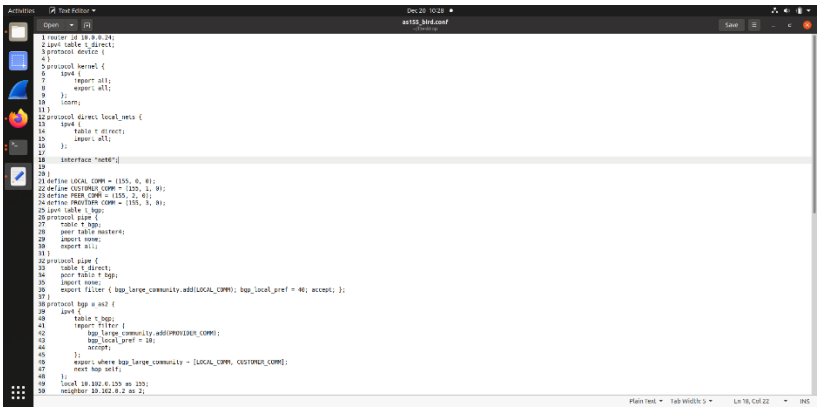


Figure 8 Content of the AS-155 Configuration File copied to the Host Machine

**Taken from Laboratory Documentation*

When the AS-155 Configuration File is examined, the following can be learned:

AS-155 Basics:

- AS-155 is a stub autonomous system.
- It has a 10.155.0.0/24 IP network.
- There is a BGP router with IP address 10.155.0.254.

BGP Peer Configurations:

- AS-155 peers with specific ASes with **protocol bgp u_as2 { }**, **protocol bgp u_as4 { }**, **protocol bgp p_as156 { }**.
- Each BGP protocol establishes a connection with a specific neighbor AS and applies specific BGP filtering.

Community Values:

- **LOCAL_COMM = (155, 0, 0);**, **CUSTOMER_COMM = (155, 1, 0);**, **PEER_COMM = (155, 2, 0);**, **PROVIDER_COMM = (155, 3, 0);**: Specific BGP community values are defined.
- These community values specify the role or relationship of the AS-155.

Pipe Protocols:

- **protocol pipe { }** and **protocol pipe { }**: Two different pipe protocols are defined.
- The first protocol moves data between the **t_bgp** table and the **master4** table.
- The second protocol performs a specific filtering between the **t_direct** table and the **t_bgp** table.

OSPF Protocol:

- **protocol ospf ospf1 { }**: Contains configurations related to the OSPF protocol.
- OSPF includes settings specific to certain network interfaces.

Task 1.a.1:

From the BGP Configuration file, the u_as2 protocol was examined to determine who AS-155 was peering with:

```
protocol bgp u_as2 {
        ipv4 {
                table t_bgp;
                import filter {
                        bgp_large_community.add(PROVIDER_COMM);
                        bgp_local_pref = 10;
                        accept;
                };
                export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
                next hop self;
                };
```

```
        local 10.102.0.155 as 155;
        neighbor 10.102.0.2 as 2;
}
```

Examining this BGP configuration block, AS-155 has the following characteristics:

- **Neighbor:** AS-155 is paired with a neighbor AS (AS-2) with IP address 10.102.0.2.
- **Next Hop Self**:
    - **next hop self;** sets the next hop address as the AS-155 itself.

In the light of this information, AS-155 is matched with the following ASes:

- AS-2 (denoted as Neighbor).

Task 1.a.2:

An experiment will be designed to demonstrate that the AS-155 can pair with multiple ASes and when it loses one connection, it can still access the internet using its other connections.

---

*"When designing this experiment, BGP sessions can be enabled/disabled using the network graph or using the "birdc" command." **

ERROR: The detail part of the routers on the map suddenly became unreachable. Therefore, bird was stopped by dcdown and then restarted with dcup. For this reason, the locations and connections of the AS-155 configuration files have changed. A recall was made and the file was examined again.

```
[12/20/23]seed@VM:~/Desktop$ dockps | grep 155
fdefbd14f369  as155r-router0-10.155.0.254
00bfd67dfc9b  as155h-host_0-10.155.0.71
efcaec7a13d3  as155h-webservice_1-10.155.0.72
[12/20/23]seed@VM:~/Desktop$ docker cp fdef:/etc/bird/bird.conf ./as155_bird.conf
```

Figure 9 The files were copied only to the host machine for proper review.

The previous operations are not shown again as they are already in the report, only the relevant part of the configuration file is quoted.

```
protocol bgp u_as2 {
  ipv4 {
    table t_bgp;
    import filter {
      bgp_large_community.add(PROVIDER_COMM);
      bgp_local_pref = 10;
      accept;
    };
    export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
    next hop self;
  };
  local 10.102.0.155 as 155;
  neighbor 10.102.0.2 as 2;
}
```

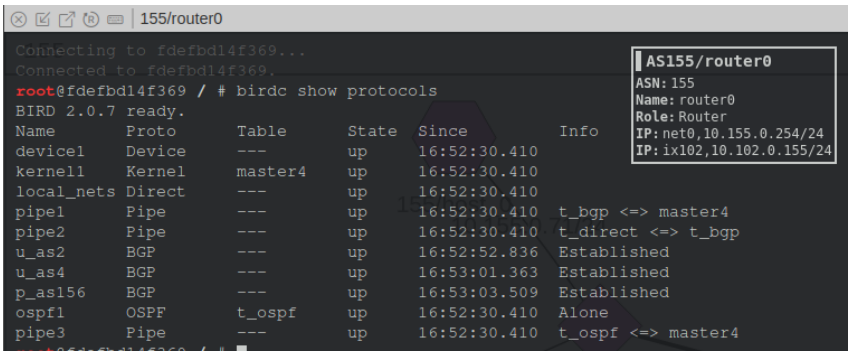
```
protocol bgp u_as4 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.102.0.155 as 155;
   neighbor 10.102.0.4 as 4;
}
protocol bgp p_as156 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PEER_COMM);
         bgp_local_pref = 20;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.102.0.155 as 155;
   neighbor 10.102.0.156 as 156;
}
```

After the error, it was learned that it could be seen that the connection was made with more than one node in more than one protocol, not only in a single protocol.

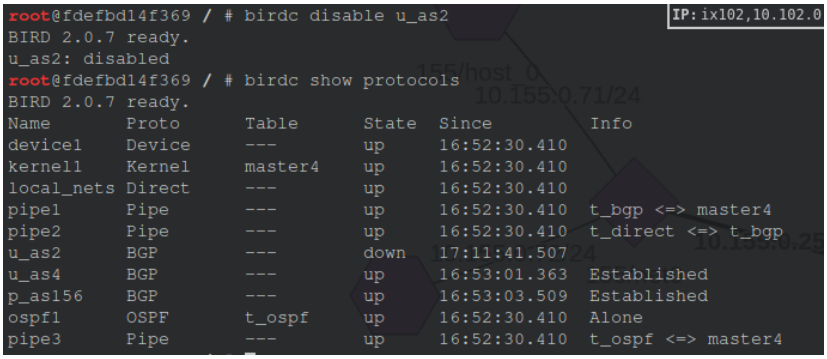
1. **protocol bgp u_as2**: This block establishes a connection with neighbor 2 with IP address 10.102.0.2 and AS number (Autonomous System) using the BGP protocol. The AS number is specified as 2.
2. **protocol bgp u_as4**: This block establishes a connection with neighbor number 4 with IP address 10.102.0.4 and AS number 4 using the BGP protocol. The AS number is specified as 4.
3. **protocol bgp p_as156**: This block establishes a connection with neighbor 156 with IP address 10.102.0.156 and AS number 156 using the BGP protocol. The AS number is specified as 156.

As a result, as-155 appears to be linked to AS number 2 (u_as2), AS number 4 (u_as4) and AS number 156 (p_as156).

The "*birdc show protocols*" command is used to provide information about the status of the BGP protocols of the AS-155 router container in the network map. With this command, it is possible to see which BGP sessions BIRD has established, what state these sessions are in (for example, up or down), when the sessions were started, and so on.



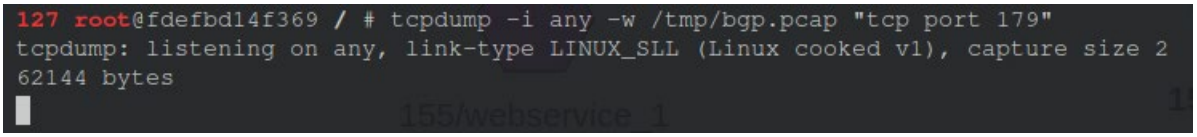
With the "*birdc disable*" command the connection to u_as2 is broken.



As can be seen here, the AS-155 container can continue to access the internet after disconnecting from u_as2 thanks to its other connections

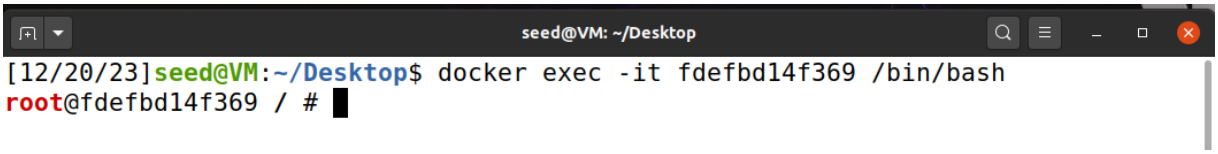
Task 1.b Observing BGP UPDATE Messages

tcpdump -i any -w /tmp/bgp.pcap The BGP packets captured using the *"tcp port 179"* command line are saved in the /tmp/bgp.pcap. directory by using the "tcp port 179" command line.



"Attaching to an existing session; if you don't see the shell prompt, try pressing the return key." message was received on the terminal opened on AS-155 to trigger UPGRADE operations. Pressing the "*return"* key only skipped the line on the terminal.

Using the id of AS-155, a new shell was opened in the terminal with the command "*docker exec -it fdefbd14f369 /bin/bash*".



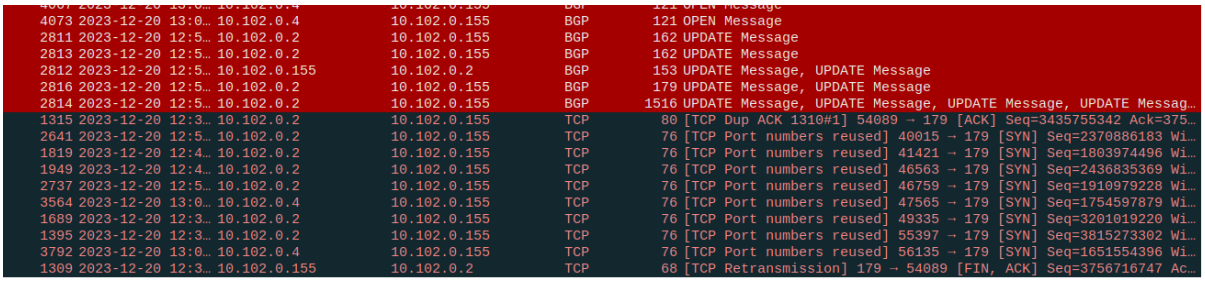
In order to trigger transactions, a route needs to be taken and re-declared, to find this route, the *"show route all*" command was used in the birdc shell and the appropriate route was found.

```
BIRD 2.0.7 ready.
bird> show route all
Table master4:
10.153.0.0/24        unicast [u_as4 17:09:53.424] * (100) [AS153i]
        via 10.102.0.4 on ix102
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 4 12 153
        BGP.next_hop: 10.102.0.4
        BGP.local_pref: 10
        BGP.large_community: (4, 1, 0) (12, 1, 0) (153, 0, 0) (155, 3, 0)
10.161.0.0/24        unicast [u_as4 16:53:09.960] * (100) [AS161i]
        via 10.102.0.4 on ix102
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 4 3 161
        BGP.next_hop: 10.102.0.4
        BGP.local_pref: 10
        BGP.large_community: (3, 1, 0) (4, 2, 0) (161, 0, 0) (155, 3, 0)
10.3.0.0/24          unicast [u_as4 16:53:07.678] * (100) [AS3i]
        via 10.102.0.4 on ix102
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 4 3
```

u_as4 protocol withdrawn and u_as2 protocol re-declared with birdc commands

```
root@fdefbd14f369 / # birdc disable u_as4
BIRD 2.0.7 ready.
u_as4: disabled
root@fdefbd14f369 / # birdc enable u_as2
BIRD 2.0.7 ready.
u_as2: enabled
root@fdefbd14f369 / # birdc show protocols
BIRD 2.0.7 ready.
Name       Proto      Table      State  Since         Info
device1    Device     ---        up     16:52:30.410
kernel1    Kernel     master4    up     16:52:30.410
local_nets Direct     ---        up     16:52:30.410
pipe1      Pipe       ---        up     16:52:30.410  t_bgp <=> master4
pipe2      Pipe       ---        up     16:52:30.410  t_direct <=> t_bgp
u_as2      BGP        ---        up     17:52:09.750  Established
u_as4      BGP        ---        down   17:51:56.924
p_as156    BGP        ---        up     16:53:03.509  Established
ospf1      OSPF       t_ospf     up     16:52:30.410  Alone
pipe3      Pipe       ---        up     16:52:30.410  t_ospf <=> master4
root@fdefbd14f369 / #
```

Bgp traffic was monitored using the tcpdump -i any -r /tmp/bgp.pcap command and UPDATE messages were found and analyzed using wireshark.




```
[12/20/23]seed@VM:~/Desktop$ mkdir ~/Desktop/bgp_capture
[12/20/23]seed@VM:~/Desktop$ docker cp fdefbd14f369:/tmp/bgp.pcap ~/Desktop/bgp_capture
[12/20/23]seed@VM:~/Desktop$
```

Figure 10 Copying BGP Messages to the created directory

Task 1.c: Experimenting with Large Communities

---

*"Suppose that the connection between AS-4 and AS-156 is broken due to a technical problem. Given that AS-4 is the sole service provider for AS-156, this would actually disconnect AS-156 from the Internet." **

The mock-up depicts a scenario in which the connection between AS-156 and AS-4 is disrupted in the event of a technical failure. In order to simulate this scenario, the connection between AS-156 and AS-4 was manually disconnected. To test the scenario, a ping will be sent from Host 10.156.0.72 IP address of AS-156 to Host 10.155.0.71 IP address of AS-155. The expected situation is that AS-155 can be reached because AS-155 and AS-156 are still paired.

```
root@9626b0ec7b30 / # ping -c 5 10.155.0.71

PING 10.155.0.71 (10.155.0.71) 56(84) bytes of data.
64 bytes from 10.155.0.71: icmp_seq=1 ttl=62 time=0.236 ms
64 bytes from 10.155.0.71: icmp_seq=2 ttl=62 time=0.267 ms
64 bytes from 10.155.0.71: icmp_seq=3 ttl=62 time=0.085 ms
64 bytes from 10.155.0.71: icmp_seq=4 ttl=62 time=0.084 ms
64 bytes from 10.155.0.71: icmp_seq=5 ttl=62 time=0.109 ms

--- 10.155.0.71 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4087ms
rtt min/avg/max/mdev = 0.084/0.156/0.267/0.078 ms
```

Since the connection between the two routers is still maintained, the predicted situation is realized and AS-155 can be accessed via AS-156. Now an attempt will be made to access AS-161. The prediction here is that this connection will not be realized.

```
root@9626b0ec7b30 / # ping -c 5  10.161.0.71

PING 10.161.0.71 (10.161.0.71) 56(84) bytes of data.
From 10.156.0.254 icmp_seq=1 Destination Net Unreachable
From 10.156.0.254 icmp_seq=2 Destination Net Unreachable
From 10.156.0.254 icmp_seq=3 Destination Net Unreachable
From 10.156.0.254 icmp_seq=4 Destination Net Unreachable
```

Figure 11 As predicted, the connection did not take place.

As a result of this experiment, it was observed that after the connection between AS-156 and AS-4 was lost due to a technical failure, AS-156 was still accessible via AS-155. However, the connection of AS-156 to AS-161 could not be established. This case provides an important scenario to understand how routing policies and business relationships in the BGP protocol can affect network traffic.

Task 1.d: Configuring the AS-180

In the scenario given in the task, it is requested to configure the BGP router of AS-180, which does not match anyone in the emulator, to the relevant BGP routers.

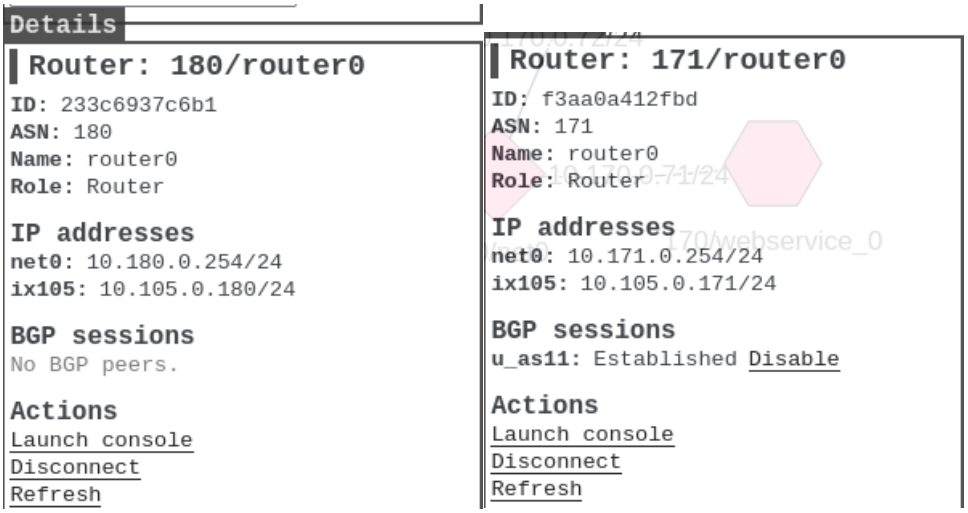


Figure 12 "No BGP Peers" Figure 13 Router 171 Details

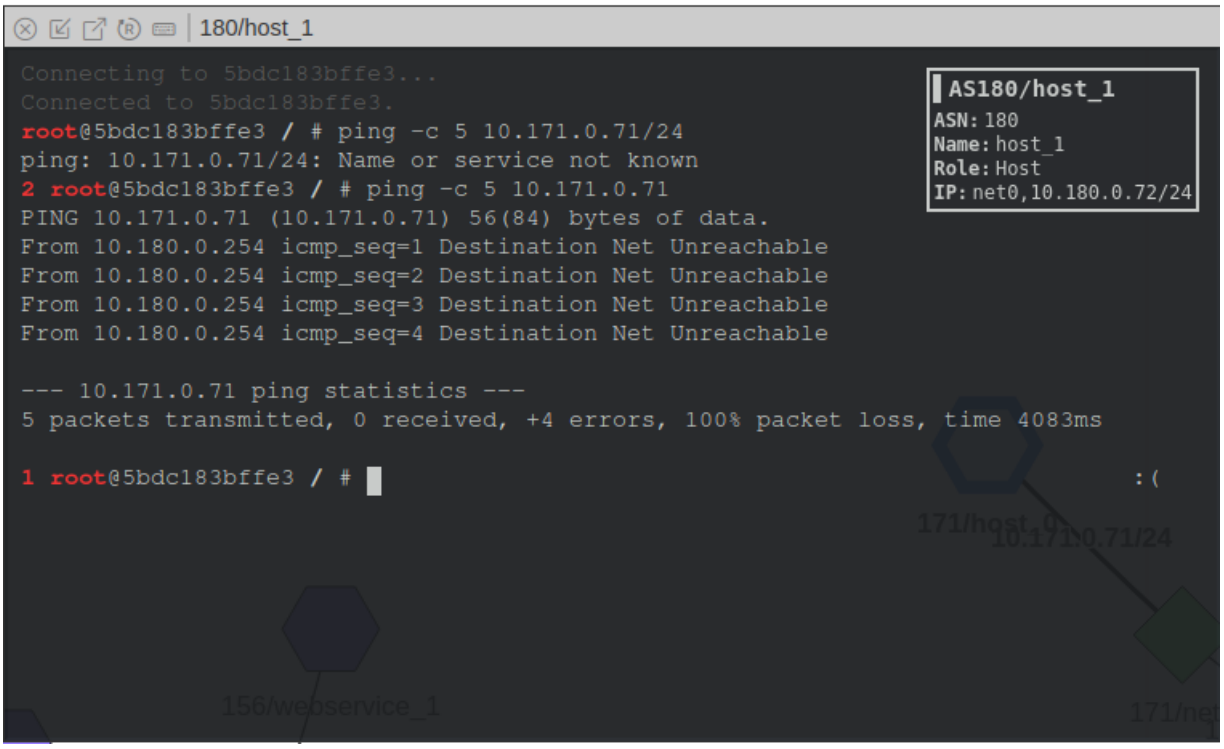


Figure 14 Unreachable AS-171 via AS-180

"*import_bird_conf.sh*" was run to extract the configuration files.

```
[12/21/23]seed@VM:~/Desktop$ cd Home
bash: cd: Home: No such file or directory
[12/21/23]seed@VM:~/Desktop$ cd /home/seed/Downloads/Labsetup/task1
[12/21/23]seed@VM:~/.../task1$ ls
export_bird_conf.sh  import_bird_conf.sh
[12/21/23]seed@VM:~/.../task1$ import_bird_conf.sh
Copy bird.conf from the container: as155r
Copy bird.conf from the container: as180r
Copy bird.conf from the container: as171r
Copy bird.conf from the container: as2r-r105
Copy bird.conf from the container: as3r-r105
[12/21/23]seed@VM:~/.../task1$
```

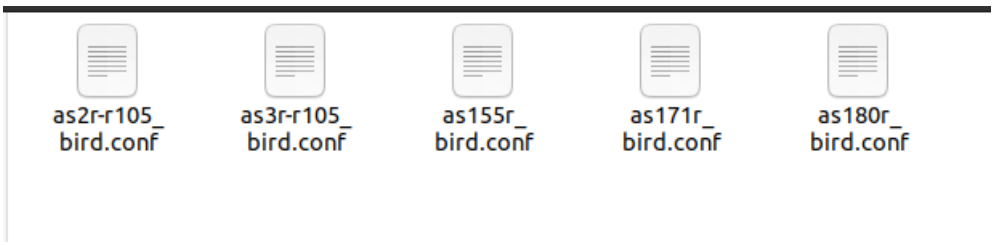


Figure 15 Configuration Files

Manual connections were made in the AS-180 and AS-171 configuration files

AS-180:

```
protocol bgp u_as2 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.180.0.171 as 180;
   neighbor 10.105.0.2 as 105;
}

protocol bgp u_as3 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.180.0.171 as 180;
   neighbor 10.105.0.3 as 105;
```

```
}

protocol bgp u_as171 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.105.0.180 as 180;
   neighbor 10.105.0.171 as 171;
}
```

AS-171:

```
protocol bgp u_as180 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.105.0.171 as 171;
   neighbor 10.105.0.180 as 180;
}
```

Reconfigured files with export_bird_conf.sh.

```
[12/21/23]seed@VM:~/.../task1$ export_bird_conf.sh
== Copy bird.conf to the container: as155r
== Execute 'birdc configure' on the container
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured

== Copy bird.conf to the container: as180r
== Execute 'birdc configure' on the container
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
/etc/bird/bird.conf:84:14 syntax error, unexpected '<'

== Copy bird.conf to the container: as171r
== Execute 'birdc configure' on the container
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured

== Copy bird.conf to the container: as2r-r105
== Execute 'birdc configure' on the container
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured

== Copy bird.conf to the container: as3r-r105
== Execute 'birdc configure' on the container
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured
```

Reconfiguration was achieved with the "*birdc configure*" command.

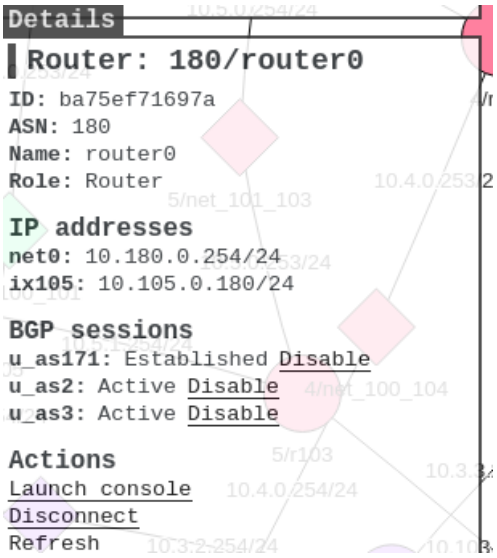


Figure 16 AS-180 Access to the Internet Unlocked

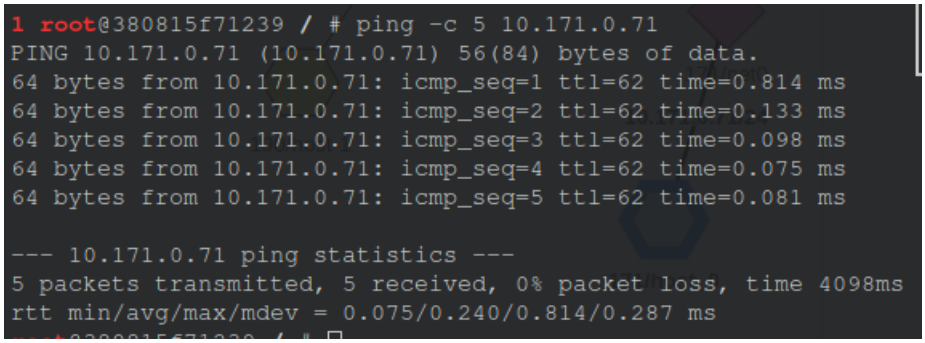


Figure 17 Access to AS-171 via AS-180.

Mission 2: Transit Autonomous System

In this task, we will focus on how two ASes in different locations can reach each other. A special type of AS is needed to solve this problem.

"This type of AS has BGP routers, which peer with other ASes at many Internet Exchange Points. Once packets enter these networks, they are usually pulled from one IX to another through internal routers and eventually handed over to another AS. Such an AS provides transit service for other ASes. This is how hosts in one AS reach hosts in another AS, even if they are not peers with each other. This special type of AS is called a ***Transit AS****."*

Task 2.a Experiments with IBGP

A ping was sent to IP address 10.164.0.71 via AS-162 to generate traffic.

```
root@cf71648a343c / # ping 10.164.0.71
PING 10.164.0.71 (10.164.0.71) 56(84) bytes of data.
64 bytes from 10.164.0.71: icmp_seq=1 ttl=59 time=0.314 ms
From 10.104.0.12 icmp_seq=2 Redirect Host(New nexthop: 164.0.104.10)
64 bytes from 10.164.0.71: icmp_seq=2 ttl=59 time=0.285 ms
From 10.104.0.12 icmp_seq=3 Redirect Host(New nexthop: 164.0.104.10)
64 bytes from 10.164.0.71: icmp_seq=3 ttl=59 time=0.272 ms
From 10.104.0.12 icmp_seq=4 Redirect Host(New nexthop: 164.0.104.10)
64 bytes from 10.164.0.71: icmp_seq=4 ttl=59 time=0.129 ms
From 10.104.0.12 icmp_seq=5 Redirect Host(New nexthop: 164.0.104.10)
64 bytes from 10.164.0.71: icmp_seq=5 ttl=59 time=0.132 ms
From 10.104.0.12 icmp_seq=6 Redirect Host(New nexthop: 164.0.104.10)
64 bytes from 10.164.0.71: icmp_seq=6 ttl=59 time=0.134 ms
```

ASN: 162
Name: host
Role: Host
IP: net0,1

icmp packets were observed on the Network Map.

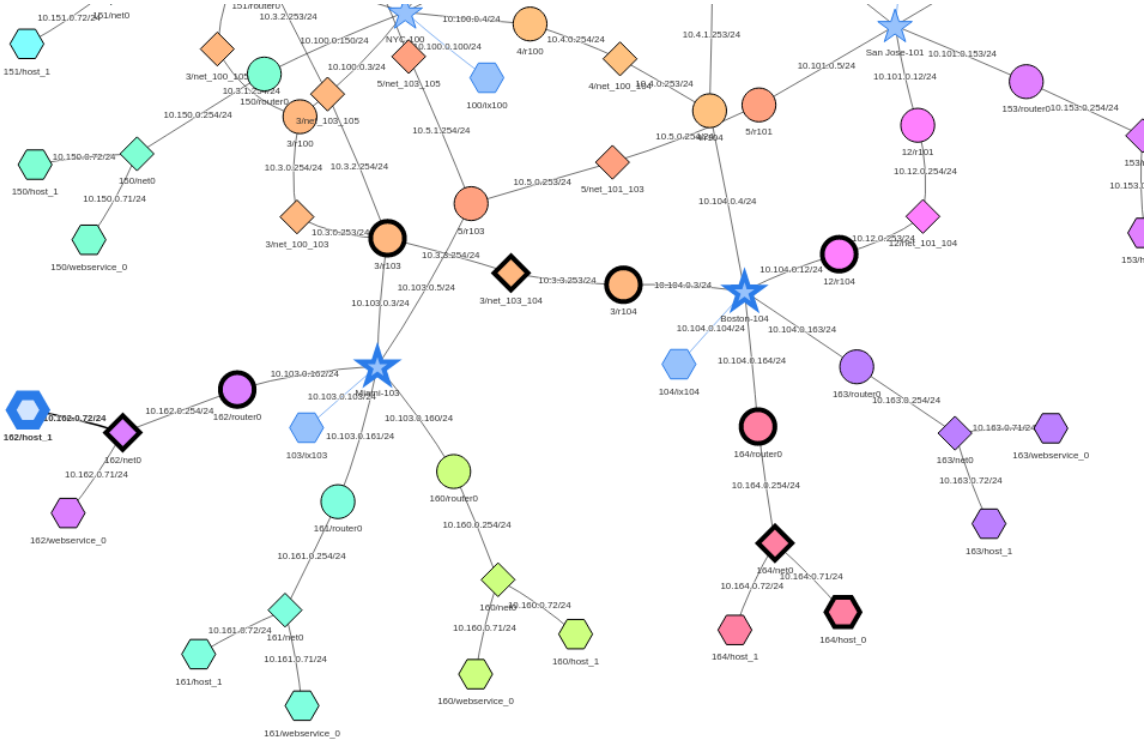


Figure 18 Path taken by ICMP packets on the network

Protocols passed by AS-3 reviewed

```
1 root@55eeb8bacb87 / # birdc
BIRD 2.0.7 ready.
bird> show protocols
Name       Proto      Table      State  Since         Info
device1    Device     ---        up     15:59:15.836
kernel1    Kernel     master4    up     15:59:15.836
local_nets Direct     ---        up     15:59:15.836
pipe1      Pipe       ---        up     15:59:15.836  t_bgp <=> master4
pipe2      Pipe       ---        up     15:59:15.836  t_direct <=> t_bgp
c_as160    BGP        ---        up     15:59:16.903  Established
c_as161    BGP        ---        up     15:59:31.715  Established
c_as162    BGP        ---        up     15:59:43.046  Established
ospf1      OSPF       t_ospf     up     15:59:15.836  Running
pipe3      Pipe       ---        up     15:59:15.836  t_ospf <=> master4
ibgp1      BGP        ---        up     15:59:24.552  Established
ibgp2      BGP        ---        up     15:59:21.177  Established
ibgp3      BGP        ---        up     15:59:23.694  Established
bird>
```

IP: ix103,10.103.0.3
IP: net_100_103,10.3
IP: net_103_105,10.3
IP: net_103_104,10.3

IBGP3 Protocol is then disabled

```
bird> disable ibgp3
ibgp3: disabled
bird> show protocols ibgp3
Name       Proto      Table      State  Since         Info
ibgp3      BGP        ---        down   16:11:16.210
bird>
```

The result obtained with the "*ip route*" command when the IBGP3 Protocol is active:

```
10.0.0.0.5 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.0.0.0.6 dev dummy0 proto bird scope link metric 32
10.0.0.0.7 via 10.3.3.3.253 dev net_103_104 proto bird metric 32
10.0.0.0.8 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.2.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.2.1.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.2.2.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.3.0.0/24 dev net_100_103 proto kernel scope scope link src 10.3.0.253
10.3.0.0/24 dev net_100_103 proto bird scope link metric 32
10.3.1.0/24 proto bird metric 32
      nexthop via 10.3.0.254 dev net_100_103 weight 1
      nexthop via 10.3.2.253 dev net_103_105 weight 1
10.3.2.0/24 dev net_103_105 proto kernel scope link src 10.3.2.254
10.3.2.0/24 dev net_103_105 proto bird scope link metric 32
10.3.3.3.0/24 dev net_103_104 proto kernel scope scope link src 10.3.3.3.254
10.3.3.3.0/24 dev net_103_104 proto bird scope link metric 32
10.4.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.4.1.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.11.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.12.0.0/24 via 10.3.3.3.253 dev net_103_104 proto bird metric 32
10.100.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.103.0.0/24 dev ix103 proto kernel scope link src 10.103.0.3
10.103.0.0/24 dev ix103 proto bird scope link metric 32
10.104.0.0/24 via 10.3.3.3.253 dev net_103_104 proto bird metric 32
10.105.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.150.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.151.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.152.0.0/24 via 10.3.3.3.253 dev net_103_104 proto bird metric 32
10.153.0.0/24 via 10.3.3.3.253 dev net_103_104 proto bird metric 32
```

```
10.154.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.155.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.156.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.160.0.0/24 via 10.103.0.160 dev ix103 proto bird metric 32
10.161.0.0/24 via 10.103.0.161 dev ix103 proto bird metric 32
10.162.0.0/24 via 10.103.0.162 dev ix103 proto bird metric 32
10.163.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.164.0.0/24 via 10.3.3.3.253 dev net_103_104 proto bird metric 32
10.170.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.171.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.190.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
```

Result obtained with the "ip route" command with the IGBP3 Protocol disabled:

```
10.0.0.0.5 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.0.0.0.6 dev dummy0 proto bird scope link metric 32
10.0.0.0.7 via 10.3.3.3.253 dev net_103_104 proto bird metric 32
10.0.0.0.8 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.2.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.2.1.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.2.2.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.3.0.0/24 dev net_100_103 proto kernel scope scope link src 10.3.0.253
10.3.0.0/24 dev net_100_103 proto bird scope link metric 32
10.3.1.0/24 proto bird metric 32
      nexthop via 10.3.0.254 dev net_100_103 weight 1
      nexthop via 10.3.2.253 dev net_103_105 weight 1
10.3.2.0/24 dev net_103_105 proto kernel scope link src 10.3.2.254
10.3.2.0/24 dev net_103_105 proto bird scope link metric 32
10.3.3.3.0/24 dev net_103_104 proto kernel scope link src 10.3.3.3.254
10.3.3.3.0/24 dev net_103_104 proto bird scope link metric 32
10.4.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.4.1.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.11.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.100.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.103.0.0/24 dev ix103 proto kernel scope link src 10.103.0.3
10.103.0.0/24 dev ix103 proto bird scope link metric 32
10.104.0.0/24 via 10.3.3.3.253 dev net_103_104 proto bird metric 32
10.105.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.150.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.151.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.154.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.155.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.156.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.160.0.0/24 via 10.103.0.160 dev ix103 proto bird metric 32
10.161.0.0/24 via 10.103.0.161 dev ix103 proto bird metric 32
10.162.0.0/24 via 10.103.0.162 dev ix103 proto bird metric 32
10.163.0.0/24 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.170.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.171.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
10.190.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32
```

Comparing the output of the "*ip route*" command before and after disabling the BGP3 protocol, the following changes are observed:

When the IBGP3 Protocol is in Effect:

10.164.0.0/24 via 10.3.3.3.253 dev net_103_104 proto bird metric 32

After IBGP3 Protocol is Disabled:

10.164.0.0/24 via 10.3.2.253 dev net_103_105 proto bird metric 32

This change indicates that the routing of the 10.164.0.0/24 network changed after the IBGP3 protocol was disabled. When the IBGP3 protocol was disabled, routing to this network was via 10.3.2.253.

This indicates that the IBGP3 protocol affects routing in the network, causing traffic route changes. The IBGP3 protocol being enabled or disabled can significantly affect the routing and traffic route behavior in the network. By observing these changes, it is possible to understand the interaction of the protocols and the overall performance of the network.

Task 2.b: Experimenting with IGP

In this task, the operations performed in the previous task are requested to be redone for the OSPF protocol. Since the steps were shown in detail in the previous task, only the result output will be given.
result
with ospf3 protocol enabled:

```
10.0.0.0.5 dev dummy0 proto bird scope link metric 32
10.0.0.0.6 via 10.3.0.0.253 dev net_100_103 proto bird metric 32
10.0.0.0.7 via 10.3.0.0.253 dev net_100_103 proto bird metric 32
10.0.0.0.8 via 10.3.1.253 dev net_100_105 proto bird metric 32
10.2.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.2.1.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.2.2.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.3.0.0/24 dev net_100_103 proto kernel scope scope link src 10.3.0.254
10.3.0.0/24 dev net_100_103 proto bird scope link metric 32
10.3.1.0/24 dev net_100_105 proto kernel scope link src 10.3.1.254
10.3.1.0/24 dev net_100_105 proto bird scope link metric 32
10.3.2.0/24 proto bird metric 32
      nexthop via 10.3.0.253 dev net_100_103 weight 1
      nexthop via 10.3.1.253 dev net_100_105 weight 1
10.3.3.3.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.4.0.0.0/24 via 10.100.0.4 dev ix100 proto bird metric 32
10.4.1.0/24 via 10.100.0.4 dev ix100 proto bird metric 32
10.11.0.0/24 via 10.3.1.253 dev net_100_105 proto bird metric 32
10.12.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.100.0.0/24 dev ix100 proto kernel scope link src 10.100.0.3
10.100.0.0/24 dev ix100 proto bird scope link metric 32
10.103.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.104.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.105.0.0/24 via 10.3.1.253 dev net_100_105 proto bird metric 32
10.150.0.0/24 via 10.100.0.150 dev ix100 proto bird metric 32
10.151.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.152.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.153.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.154.0.0/24 via 10.3.1.253 dev net_100_105 proto bird metric 32
10.155.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.156.0.0/24 via 10.100.0.4 dev ix100 proto bird metric 32
10.160.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
```

```
10.161.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.162.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.163.0.0/24 via 10.100.0.4 dev ix100 proto bird metric 32
10.164.0.0/24 via 10.3.0.253 dev net_100_103 proto bird metric 32
10.170.0.0/24 via 10.3.1.253 dev net_100_105 proto bird metric 32
10.171.0.0/24 via 10.3.1.253 dev net_100_105 proto bird metric 32
10.190.0.0/24 via 10.3.1.253 dev net_100_105 proto bird metric 32
```

result after disabling the ospf3 protocol:

```
10.2.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.2.1.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.2.2.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.3.0.0/24 dev net_100_103 proto kernel scope scope link src 10.3.0.254
10.3.0.0/24 dev net_100_103 proto bird scope link metric 32
10.3.1.0/24 dev net_100_105 proto kernel scope link src 10.3.1.254
10.3.1.0/24 dev net_100_105 proto bird scope link metric 32
unreachable 10.3.2.0/24 proto bird metric 32
unreachable 10.3.3.3.0/24 proto bird metric 32
10.4.0.0.0/24 via 10.100.0.4 dev ix100 proto bird metric 32
10.4.1.0/24 via 10.100.0.4 dev ix100 proto bird metric 32
10.11.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.12.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.100.0.0/24 dev ix100 proto kernel scope link src 10.100.0.3
10.150.0.0/24 via 10.100.0.150 dev ix100 proto bird metric 32
10.151.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.152.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.153.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.154.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.155.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
10.156.0.0/24 via 10.100.0.4 dev ix100 proto bird metric 32
unreachable 10.160.0.0/24 proto bird metric 32
unreachable 10.161.0.0/24 proto bird metric 32
unreachable 10.162.0.0/24 proto bird metric 32
10.163.0.0/24 via 10.100.0.4 dev ix100 proto bird metric 32
10.164.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
unreachable 10.170.0.0/24 proto bird metric 32
10.171.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
unreachable 10.190.0.0/24 proto bird metric 32
```

Comparing the results between the two cases, we observe some significant changes after disabling the ospf3 protocol:

When OSPF3 Protocol is Active:

Routes for 10.3.2.0/24 and 10.3.3.3.0/24 networks are available.

There are routes to 10.150.0.0/24, 10.160.0.0/24, 10.161.0.0/24 and 10.162.0.0/24 networks.

After the OSPF3 Protocol is Disabled:

Routes for 10.3.2.0/24 and 10.3.3.3.0/24 networks are "unreachable".

Routes to 10.160.0.0/24, 10.161.0.0/24, 10.162.0.0/24 and 10.190.0.0/24 are in "unreachable" state.

These changes show that disabling the ospf3 protocol resulted in losing connections to these networks. The OSPF3 protocol was involved in the creation of routes to these networks and provided transit for traffic passing through them.

Protocols such as the Intrinsic Gateway Protocol (IGP) are used to route traffic between Automated Systems (AS) and provide connectivity between different ASes. OSPF (Open Shortest Path First) is an IGP protocol and determines routes between networks using a shortest path algorithm. Therefore, traffic routing between transit autonomous systems occurs seamlessly, especially when IGP protocols are in place.

Based on our observations, we can say that IGP protocols are essential for traffic routing between transit autonomous systems. Disabling IGP protocols may cause disruptions in traffic routing and loss of connectivity to some networks. Therefore, it is important to use IGP protocols to ensure a reliable connection between transit ASes.

Task 2.c: AS-5 Configuration

In this task, operations will be performed on AS-5, which connects to three internet exchanges (IX-101, IC-103, IX-105) but does not match any AS.

First, the IBGP packet configuration between AS-5 and IX-101 was analyzed. Then the elements were investigated.

```
protocol bgp ibgp1 {
   ipv4 {
      table t_bgp;
      import all;
      export all;
      igp table t_ospf;
   };
   local 10.0.0.0.12 as 5;
   neighbor 10.0.0.0.13 as 5;
}
protocol bgp ibgp2 {
   ipv4 {
      table t_bgp;
      import all;
      export all;
      igp table t_ospf;
   };
   local 10.0.0.0.12 as 5;
   neighbor 10.0.0.0.14 as 5;
}
```

ibgp1 and ibgp2 Protocols: In both protocols, the local and neighbor parameters specify the IP addresses of the BGP routers in the AS-5 to themselves and to each other.

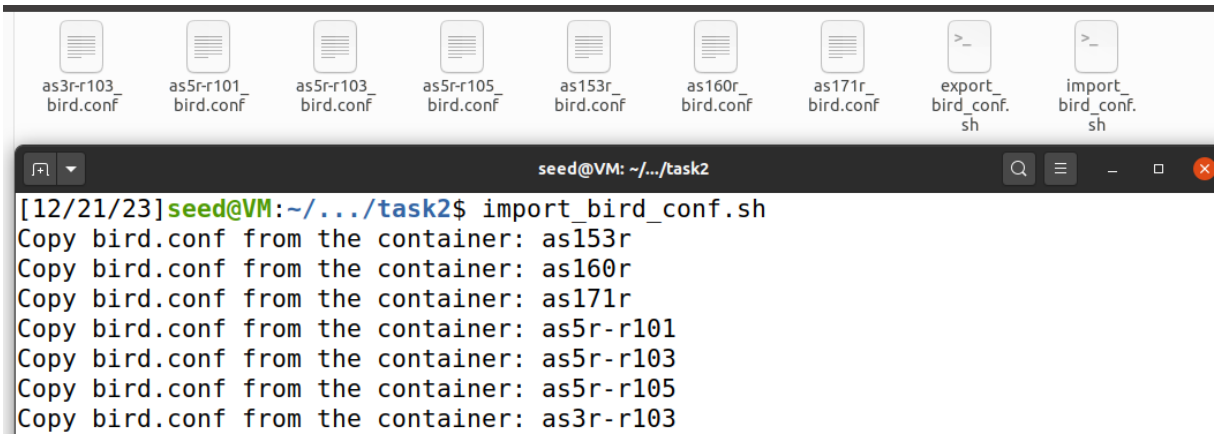
AS Number: In both cases the as parameter is set to 5 because these routers belong to AS-5.

table Parameter under ipv4: The table t_bgp parameter, which specifies which table BGP will use, is set to t_bgp in both cases.

igp table Parameter: igp table t_ospf: Specifies the IGP table and is set to t_ospf in both cases.

import and export Parameters: The import all and export all statements allow BGP to import and export all routes.

For the configuration of AS-5, we first downloaded the necessary configuration files with the import_bird_conf.sh command.



Manually modified configuration file to connect AS-101 to AS-5 on IX-153

```
protocol bgp p_rs153 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PEER_COMM);
         bgp_local_pref = 20;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.0.0.0.12 as 5;
   neighbor 10.101.0.153 as 153;
} //In the AS-5 file
protocol bgp u_as5 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.101.0.153 as 153;
   neighbor 10.0.0.0.12 as 5;
} //In the AS-153 File
```

Manually modified configuration file to connect AS-103-5 to AS-160 on IX-160

```
protocol bgp p_rs160 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PEER_COMM);
         bgp_local_pref = 20;
```

```
        accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.0.0.0.13 as 5;
   neighbor 10.103.0.160 as 160;
} //In the AS-5 File
protocol bgp u_as5 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.103.0.160 as 160;
   neighbor 10.0.0.0.13 as 5;
}//AS-160 file
```

Manually modified configuration file to connect AS-105-5 to AS-171 on IX-105

```
protocol bgp p_rs171 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PEER_COMM);
         bgp_local_pref = 20;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.0.0.0.14 as 5;
   neighbor 10.105.0.171 as 171;
} //In the AS-5 file
protocol bgp u_as5 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.105.0.171 as 171;
   neighbor 10.0.0.0.14 as 5;
}//AS-171 file
```

Executing *"export_bird_conf.sh"* sends the configuration files back to their containers

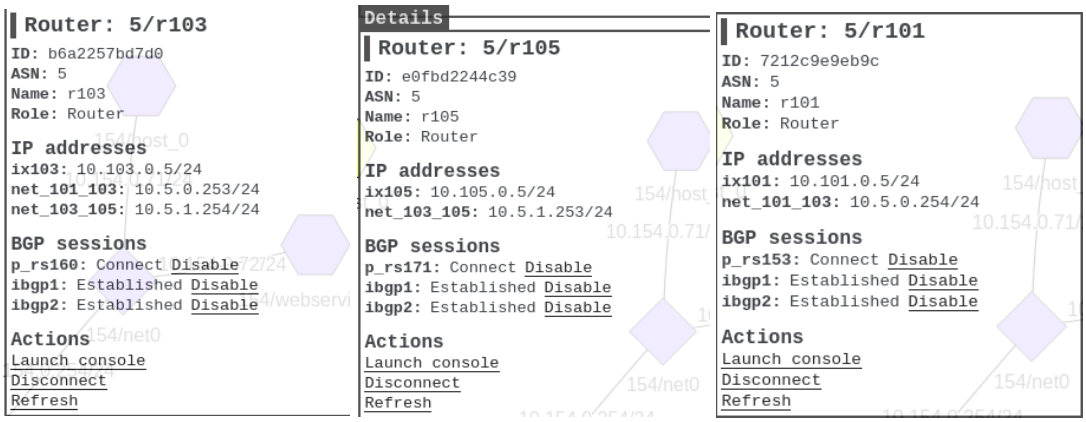


Figure 19 BGP connections successfully realized

> *"AS-5 and AS-3 are both transit ASes and decide to peer at IX-103 (Miami). Since they are about the same size, they benefit equally from peering, so they decide to make their relationship peer-to-peer instead of provider-to-customer. They do not pay each other fees."*

For the realization of this scenario given in the documentation, a connection was established between AS-5 and AS-3.

```
protocol bgp u_as3 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(PEER_COMM);
            bgp_local_pref = 20;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
        next hop self;
    };
    local 10.0.0.13 as 5;
    neighbor 10.103.0.3 as 3;
}
```

```
protocol bgp u_as5 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(PEER_COMM);
            bgp_local_pref = 20;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
        next hop self;
    };
    local 0.103.0.3 as 3;
    neighbor  10.0.0.13 as 5;
}
```

Executing *"export_bird_conf.sh"* sends the configuration files back to their containers

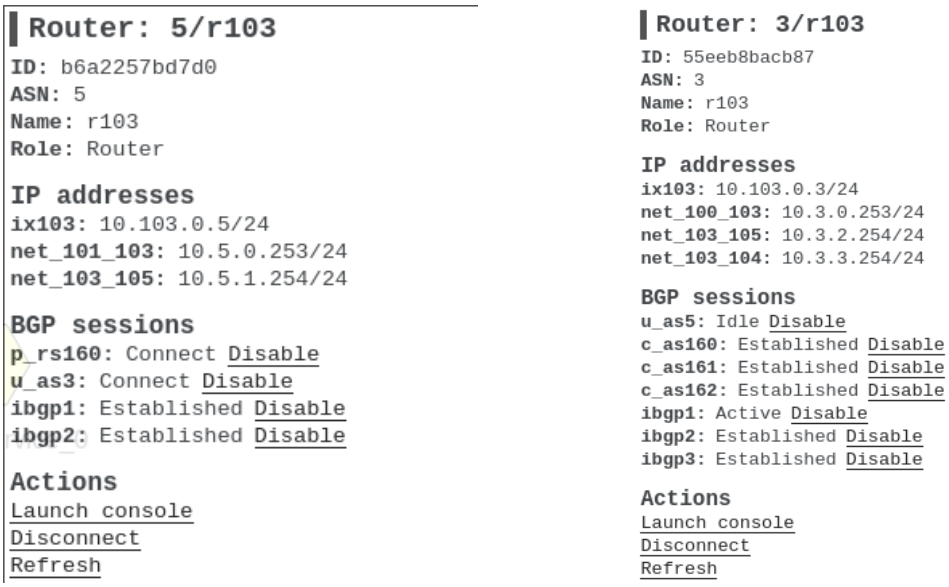


Figure 20 BGP connections successfully realized

Task 3: Path Selection

"BGP routers typically receive multiple routes to the same network. All these routes are kept, but BGP selects one route as the best valid route by running a route selection algorithm. This route will be the route advertised to peers and will also be the route given to the core routing table, so routing is done over this selected route. When the best route is revoked, the BGP router re-runs the route selection algorithm to find the current best route.

The top 2 criteria used by BIRD are: (1) preference for the route with the highest Local Preference attribute, (2) preference for the route with the shortest AS path. In this task, you will experiment with these two criteria and see how they affect route selection."

Task 3.a

With the command line "*birdc show route all*" we navigated to the BGP router of the AS-150 and showed all BGP routes.

See AS150_Location_Signs File in the same file

```
10.153.0.0/24 unicast [u_as2 15:59:24.826] * (100) [AS153i]
        via 10.100.0.2 on ix100
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 2 12 153
        BGP.next_hop: 10.100.0.2
        BGP.local_pref: 10
        BGP.large_community: (2, 1, 0) (12, 1, 0) (153, 0, 0) (150, 3, 0)
          unicast [u_as3 16:33:52.376] (100) [AS153i]
        via 10.100.0.3 on ix100
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 3 2 12 153
        BGP.next_hop: 10.100.0.3
```

```
BGP.local_pref: 10
BGP.large_community: (2, 1, 0) (3, 2, 0) (12, 1, 0) (153, 0, 0) (150, 3, 0)
```

The "*" indicates the best available route. We can also guess that 10.153.0.0./24 is the fastest route because its as_path is short and its local_pref value is 10.

Task 3.b

AS-150's configuration file copied from container to host machine

```
[12/21/23]seed@VM:~/Desktop$ docker cp 55bb:/etc/bird/bird.conf ./as150_bird.conf
[12/21/23]seed@VM:~/Desktop$
```

In the configuration file of AS-150, the local_pref values of AS-2 and AS-3 protocols were changed. AS-3's local_pref value was increased from 10 to 20, so it gained priority in the preference order.

```
protocol bgp u_as2 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 10;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.100.0.150 as 150;
   neighbor 10.100.0.2 as 2;
}
protocol bgp u_as3 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 20;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.100.0.150 as 150;
   neighbor 10.100.0.3 as 3;
}
```

The AS-150 configuration file was sent back to the container and reconfigured.

```
[12/21/23]seed@VM:~/Desktop$ docker cp ./as150_bird.conf 55bb:/etc/bird/bird.conf
[12/21/23]seed@VM:~/Desktop$ docker exec 55bb birdc configure
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured
[12/21/23]seed@VM:~/Desktop$
```

With the command line "*birdc show route all*" we navigated to the BGP router of the AS-150 and showed all BGP routes.

See AS_3vsAS_2 in the same file

The changes made here confirmed that AS-3 was preferred earlier.

Task 4: IP Anycast

> *"IP Anycast is a methodology of using and routing a single IP address that is usually shared by multiple machines located in different locations. When we send a packet to this IP address, the machine receiving the packet is uncertain. Because which machine will receive it depends on the routing. IP anycast is natively supported by BGP. A well-known application of IP anycast is DNS, because all 13 root servers from A to M are located in more than one location."*

Two networks of AS-190 connected to IX-100 and IX-105 were found on the network map. It was observed that neither of them had a connection.

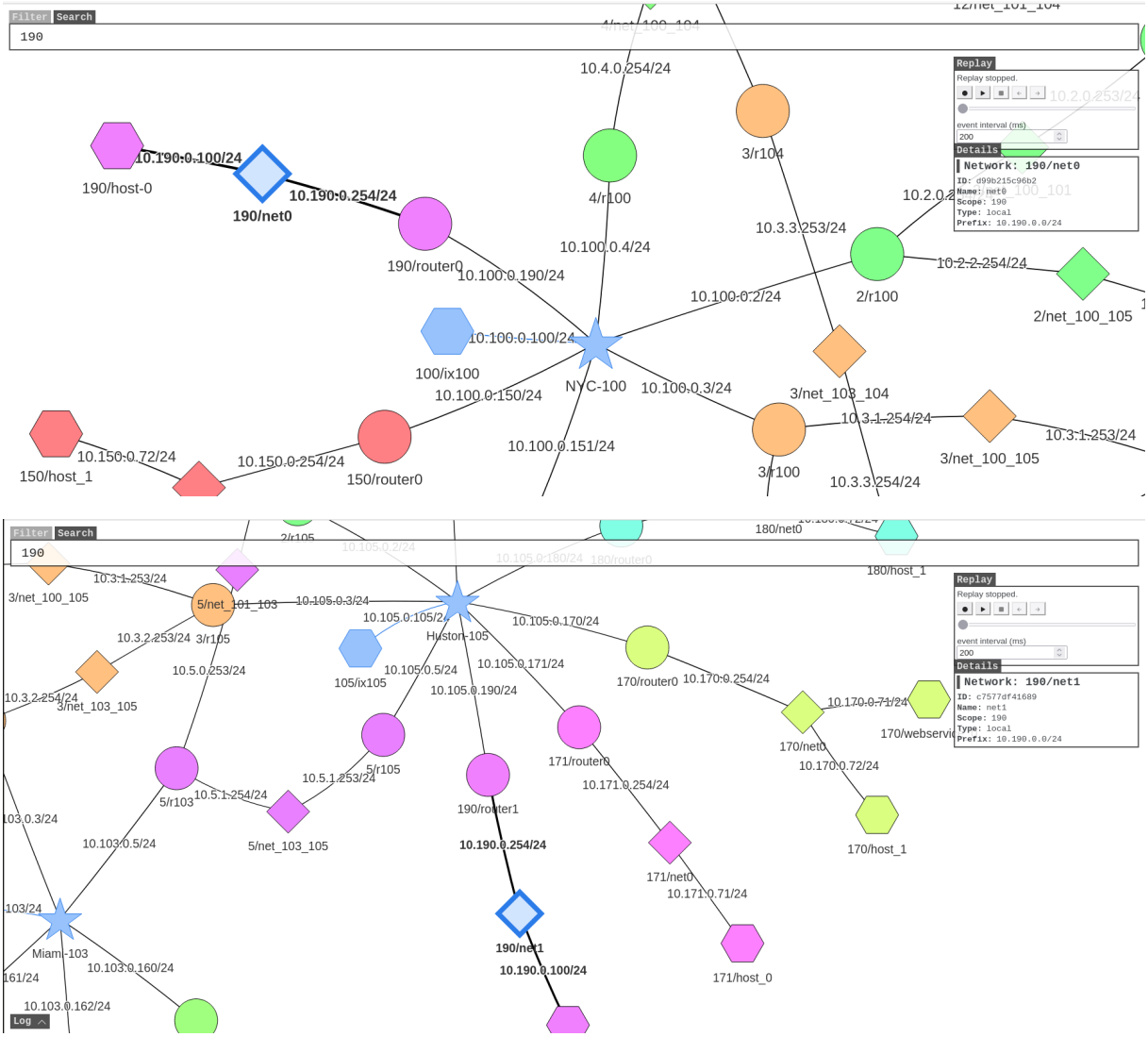


Figure 21 AS-190 Network 0 and Network 1

It was observed that both networks have the same prefix (10.190.0.0/24.). The hosts connected to these two networks have exactly the same IP Address (10.190.0.100).

Two different hosts were selected for the experiment (155 and 171). When pinging the IP Address 10.190.0.100 from these two hosts, the predicted result should show that there are different destinations in the icmp packet streams.

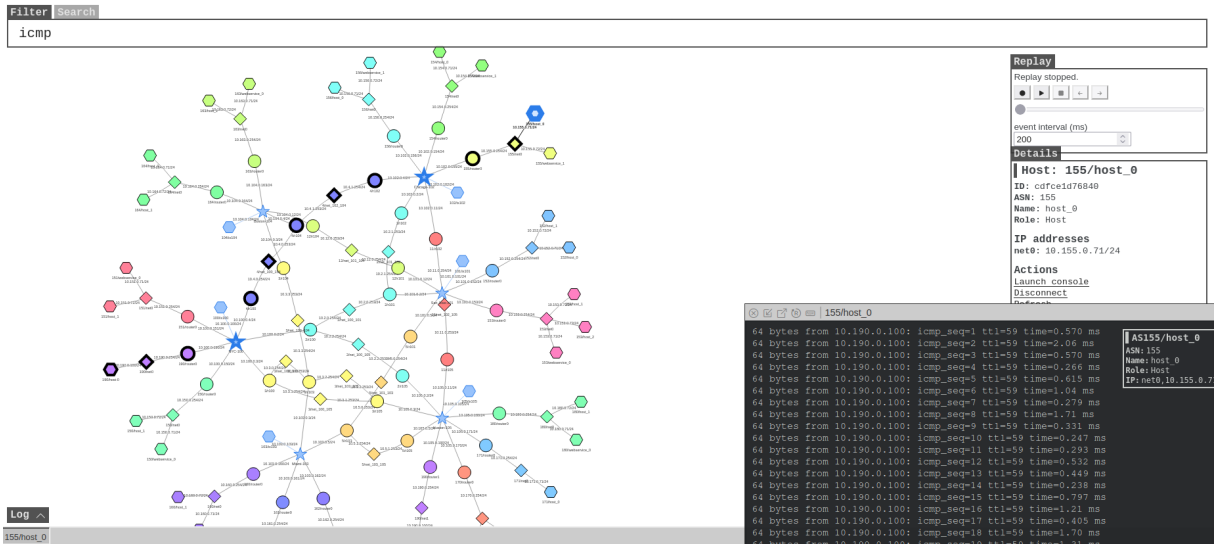


Figure 22 Ping Sent from 23

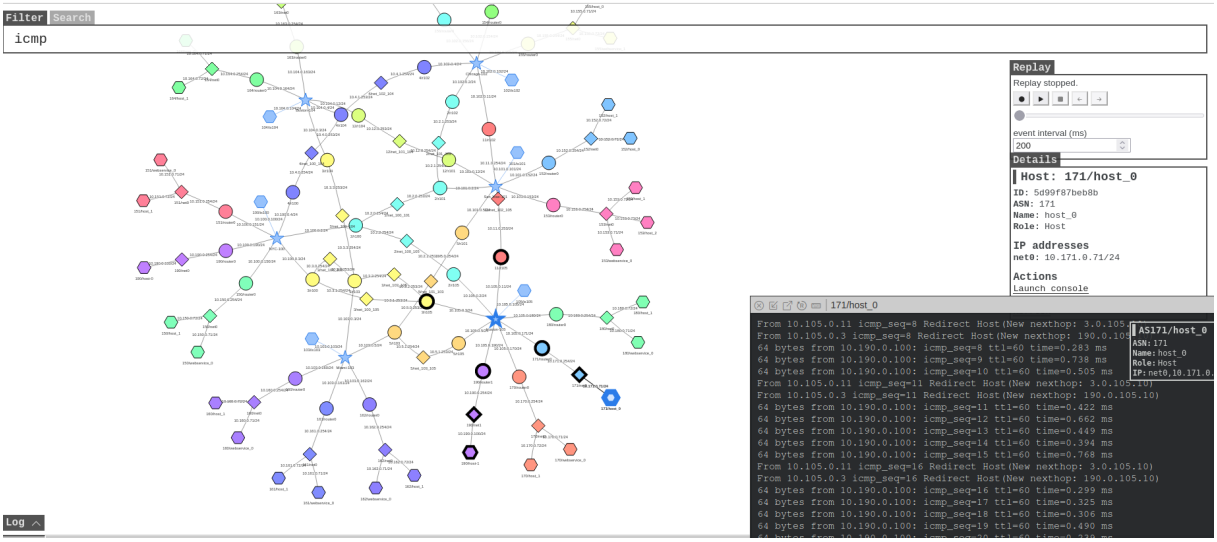


Figure 24 Ping Sent from AS-171

The result was in line with what was expected before the experiment. When pinging the IP Address 10.190.0.100 from the hosts of AS-150 and AS-171, packets flowed through different paths even if they were the same IP address. Based on these observations, a report was written explaining why the packets reached different destinations.

Observation Report: IP Anycast and BGP Route Selection Analysis

This report examines the IP Anycast and BGP route selection process serving over two Points of Presence (PoPs) of AS-190. The two PoPs of AS-190 at locations IX-100 (peer with AS-4) and IX-105 (peer with AS-3) are not connected to each other. The networks in both PoPs are identified by the network prefix 10.190.0.0/24 and both have a host with IP address 10.190.0.100.

The BGP routers of AS-190 announce the 10.190.0.0/24 network prefix from both locations. These announcements are received and forwarded by other ASes, and some ASes, especially transit ASes, may see more than one AS path to this network. The AS path selection algorithm selects the best path among the multiple paths and announces it to the other ASes. AS path length is an important criterion for AS path selection. In particular, if there is a tie in other important criteria (e.g. local preference), a shorter AS path is more likely to be selected.

In the "ping 10.190.0.100" experience from two different sender ASes, AS-156 and AS-160, the ICMP traffic from both locations reaches different destinations. This means that traffic from AS-156 is routed to New York City (IX-100), while traffic from AS-160 is routed to Houston (IX-105).

In the BGP routing table of AS-156, the AS path is set as "4 190", while in the BGP routing table of AS-160, the AS path is set as "3 190". This indicates that AS-156 is connecting with AS-4 in IX-100 (New York City) and AS-160 is connecting with AS-3 in IX-105 (Houston).

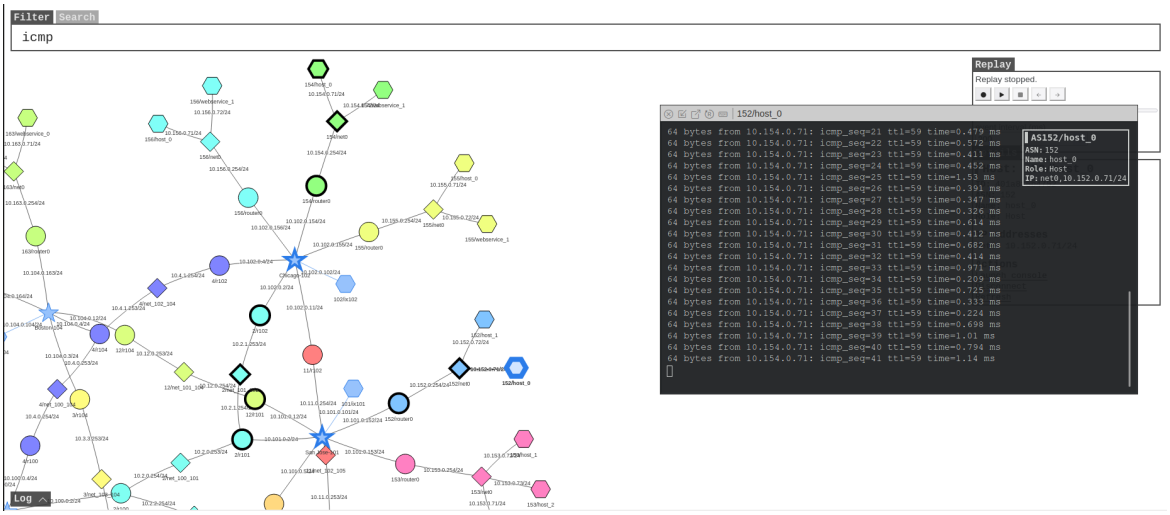
As a result, BGP's route selection process routes traffic from both locations to different destinations based on criteria such as AS path length. The BGP routing tables of AS-156 and AS-160 play an important role in determining which path to send packets to.

Task 5: BGP Prefix Attacks

BGP Route Hijacking is a type of BGP attack, also called "prefix hijacking". In this attack, the attackers' BGP routers announce IP prefixes that are not assigned to them, so traffic destined to that IP prefix is routed to the attackers' side, and the attackers can approve or modify that traffic. Many incidents on the Internet are triggered by this type of "attack", although usually due to misconfigured BGP routers.

Task 5.a: Launching a Prefix Hijacking Attack from AS-161

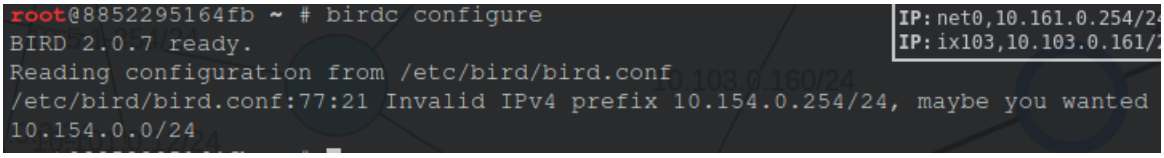
The configuration files of AS-154 and AS-161 were copied to the host machine. In the experiment, it was detected that AS-154 was black holed and packets sent to AS-154 were requested to be redirected to AS-161. First, the packets were traced by pinging AS-154 from the Host of AS-152 while the attack was not made.



The IP Addresses of AS-154's Router were found (10.154.0.254/24 || 10.102.0.154/24). Then the code for prefix hijacking was written in the configuration file of AS-161.

```
protocol static {
        ipv4 {
                table t_bgp;
        };
        route 10.154.0.254/24 blackhole {bgp_large_community.add(LOCAL_COMM); };
        route 10.102.0.154/24 blackhole {bgp_large_community.add(LOCAL_COMM); };
}
```

The edited configuration file was copied back to the container and reconfigured. To test that the attack worked successfully, AS-152 to AS-154 was pinged again. No change in the path was seen and an error was received while reconfiguring AS-161.



Then, after a reading of the Lab file, it was learned that what was needed for routing was not the IP Address, but the prefix. It was understood that the prefix could be learned from the Direct protocol. As a result, the configuration file of the AS-154 was examined.

```
protocol direct local_nets {
   ipv4 {
      table t_direct;
      import all;
   };
   interface "net0";
}
```

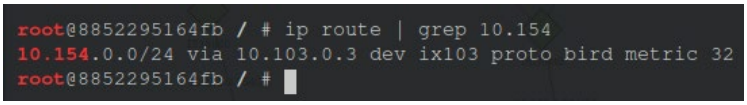
If the interface is "net0", the prefix is 10.154.0.0/24.

```
protocol static {
        ipv4 {
                table t_bgp;
        };
        route 10.154.0.0/24 blackhole {bgp_large_community.add(LOCAL_COMM); };
}
```

The edited configuration file was copied back to the container and reconfigured. To test that the attack worked successfully, AS-152 to AS-154 was pinged again. There was no change in the path and AS-161 was reconfigured but again without success. Investigations have begun to find out what went wrong.

After reading the relevant chapters of the book Internet Security: A Hands-on Approach (Computer & Internet Security), it was decided to start a trial process again. First of all, the contents of the AS-161's configuration file were cleaned and restored to its original state.

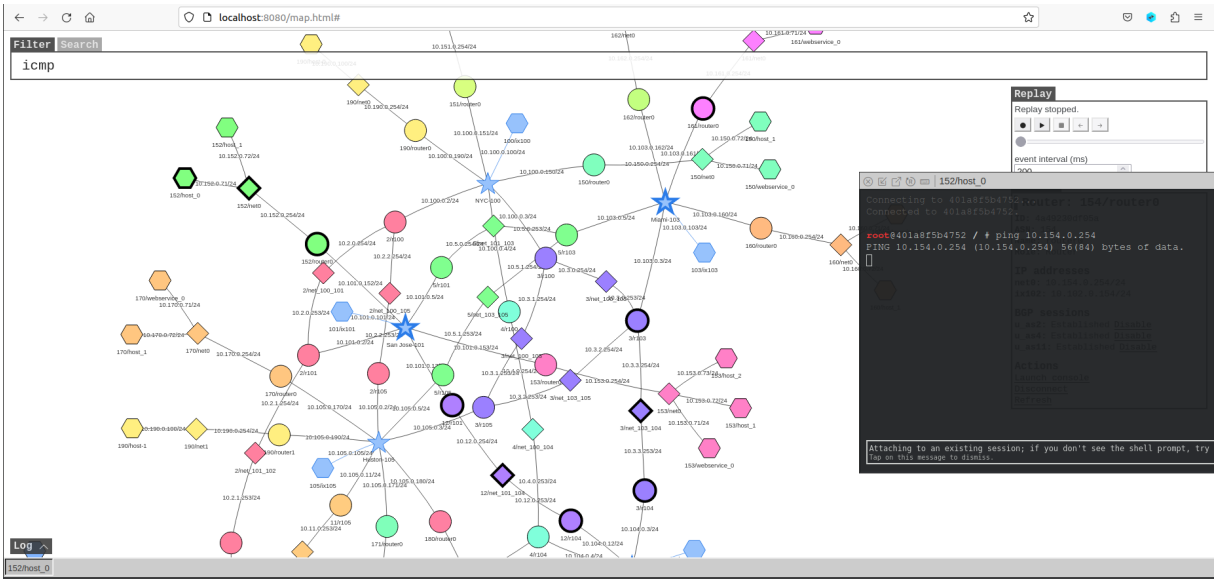
The first step was to find the prefix of AS-154 with the IP router via AS-161. The "*ip route | grep*" command was used for this.



Using the information obtained, the static protocol was written to the AS-161. An extra 1 bit was added to the IP addresses.

```
protocol static hijacks {
        ipv4 { table t_bgp; };
        route 10.154.0.0.0/25 blackhole {
                bgp_large_community.add(LOCAL_COMM);
        };
        route 10.154.0.128/25 blackhole {
                bgp_large_community.add(LOCAL_COMM);
        };
}
```

The edited AS-161 protocol was copied from the host to the container and reconfigured.



The attack was successful and packets sent from AS-152 to AS-154 were redirected to AS-161.

Mission 5.b: AS-154's Counterattack

"The AS-154 has a detection system. After the attack, it detected the attack immediately. It contacted the operators of AS-161, the upstream service provider of AS-3, and asked them to block the attack. Unfortunately for the AS-3 operators, it was midnight and no one could be reached. The loss of service is crucial, so the operators of AS-154 decide to fight back: They want to "steal back" their network prefix without the help of AS-3. Please reconfigure AS-154's BGP routers so that you can get the traffic back."

In order to process this scenario given in the laboratory documentation, readings were done first. Related chapters of the book Internet Security: A Hands-on Approach (Computer & Internet Security) were read. And focused on a real life example given in the book.

Real Life Example:

"On Sunday, February 24, 2008, after receiving a censorship order from the government to block YouTube (due to some material posted on YouTube), Pakistan Telecom (AS17557) decided to block

access to YouTube's IP address, which includes various addresses in 208.65. .153.0/24 network domain. The technique used in the blocking is the same as IP prefix hijacking, i.e. AS17557 started advertising the prefix 208.65.153.0/24 to its peers.

These BGP announcements (BGP UPDATE messages) were supposed to be announced only to peers in Pakistan, but a mistake was made and therefore the UPDATE messages were announced to one of the upstream peers, Hong Kong-based PCCW Global (AS3491). AS3491 was supposed to catch this fake announcement but failed to do so and forwarded the fake announcement to the rest of the Internet. This resulted in the hijacking of YouTube traffic on a global scale [RIPE NCC, 2008].

One of the prefixes YouTube announced (AS36561) is 208.65.152.0/22. Since 208.65.153.0/24 is a more specific prefix within the 208.65.152.0/22 address space, packets going to 208.65.153.0/24 will be routed to Pakistan instead of YouTube. Pakistan Telecom's announced prefix has therefore hijacked part of YouTube's prefix.

YouTube soon realized this, while trying to solve the problem on the normal channel (contacted PCCW to fix the error), YouTube tried to reclaim IP prefixes by announcing the same 208.65.153.0/24, but this did not solve the problem completely, because this prefix has the same length as the one introduced by Pakistan, so which path to choose depends on the path selection algorithm of each BGP router (the length of the AS path is one of the criteria). With this announcement, YouTube can get some traffic back, but not all of it.

YouTube has fixed this by announcing two more prefixes: 208.65.153.128/25 and 208.65.153.0/25. These prefixes cover the entire 208.65.153.0/24 domain and are longer, so all routers on the Internet started redirecting traffic back to YouTube. Eventually YouTube got in touch with PCCW, PCCW withdrew the fake announcements and the problem was solved."

Based on this example, it was decided to generate prefixes that are 1 bit longer than the attacker's prefix, as YouTube does.

```
protocol static {
        ipv4 { table t_bgp; };
        route 10.154.0.0/26 via "net0" {
                bgp_large_community.add(LOCAL_COMM);
        };
        route 10.154.0.64/26 via "net0" {
                bgp_large_community.add(LOCAL_COMM);
        };
        route 10.154.0.128/26 via "net0" {
                bgp_large_community.add(LOCAL_COMM);
        };
        route 10.154.0.192/26 via "net0" {
                bgp_large_community.add(LOCAL_COMM);
        };
}// Inside the AS-154 Configuration File
```

The modified AS-154 protocol was copied from the host to the container and reconfigured. Examination of packet destinations in the AS-152 protocol showed that AS-154 resisted the attacker.

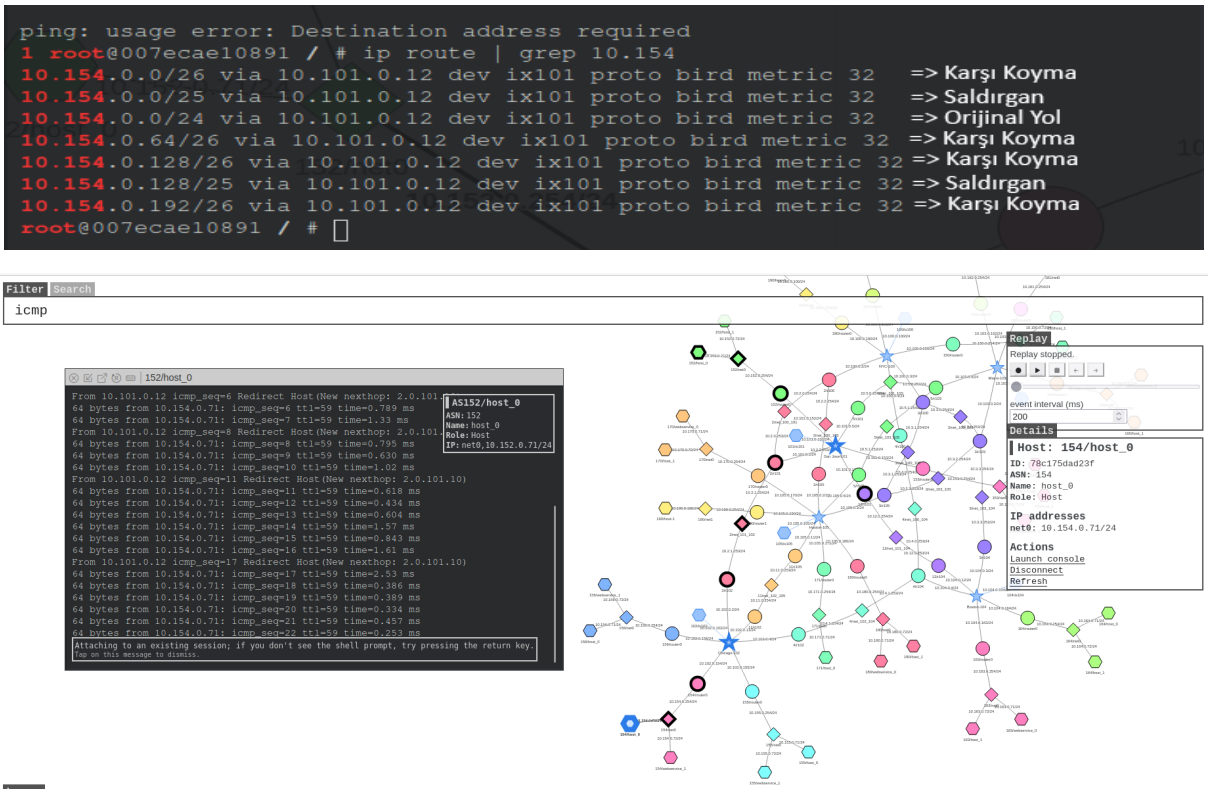


Figure 25 Packages Go Back to Where They Belong

Task 5.c: Solving the Problem in AS-3

"Eventually the operators of AS-3 are reached. Not knowing whether this is a misconfiguration on the AS-161 side or a deliberate attack, AS-3 decides not to cut the peering with AS-161, so that AS161 users can still access the Internet (AS-3 is the only AS-161 provider). But AS-3 needs to stop the spread of the fake announcement."

First of all, the configuration file of the AS-3 was examined in order to find a solution to the problem. Since it was already downloaded in Task 2, there was no need to copy it back to the host machine.

A filter rule was added to AS-3's configuration at IX-103 where they peer with AS-161, so that when a route is passed in from AS-161, only prefix 10.150.0.0/24 is passed in. In this way, fake routes announced by AS-161 will not be accepted by AS-3 and will not be able to access the internet.

```
if (net != 10.161.0.0/24) then reject;
```

After adding the filter condition, the destination connections of AS-152 to AS-161 were examined with ip route.

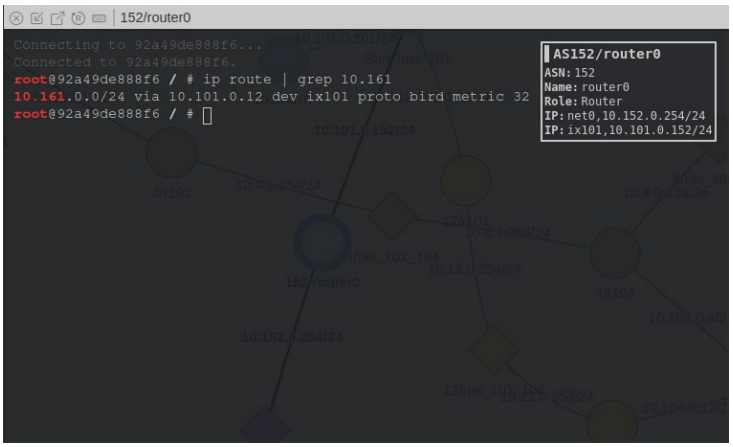


Figure 26 Filter Working Successfully