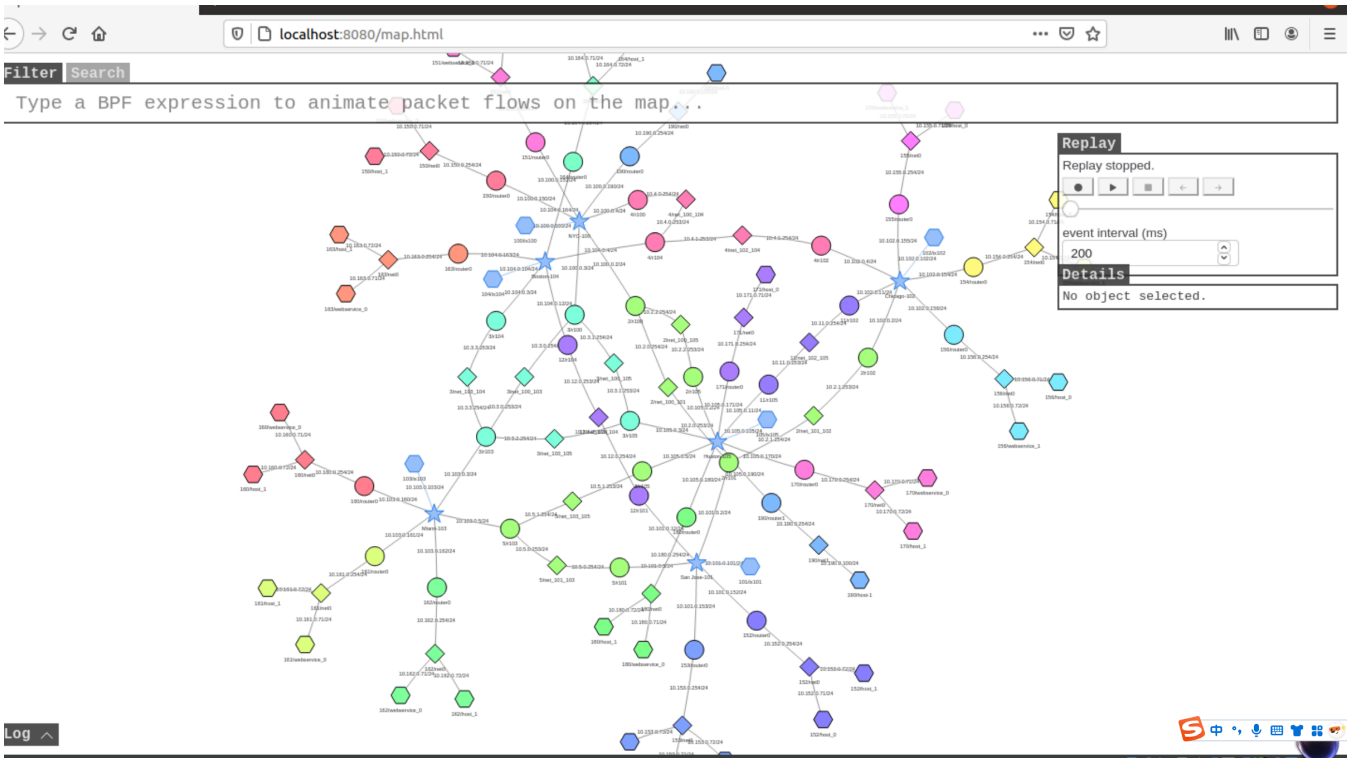
BGP_Exploration Lab

Before:

容器启动废了老大劲，又出现了Firewall实验时的问题，不过还是解决了，但是由于启动的服务很多，还是花了好久，完成后访问 http://localhost:8080/map.html，即可访问互联网仿真器



接着我们摘录一下**关于仿真器中的约定**

为了使用户能够容易地识别仿真实验中的各个节点的角色，我们创建了一套给各种节点分配不同编号时要遵循的约定。这些约定仅适用于仿真实验，在现实世界中并不适用。

• 自治系统号码（ASN）分配：

– ASN 2 - 9： 大型中转自治系统（例如国家骨干网）。

– ASN 10 - 19： 较小的中转自治系统。

– ASN 100 - 149： 互联网交换所 (IX)。

– ASN 150 - 199： Stub 自治系统（Stub AS）。

• 网络前缀和 IP 地址：

– 对于自治系统 N，其第一个内部网络的前缀为 10.N.0.0/24，第二个内部网络为 10.N.1.0/24，依此类推。

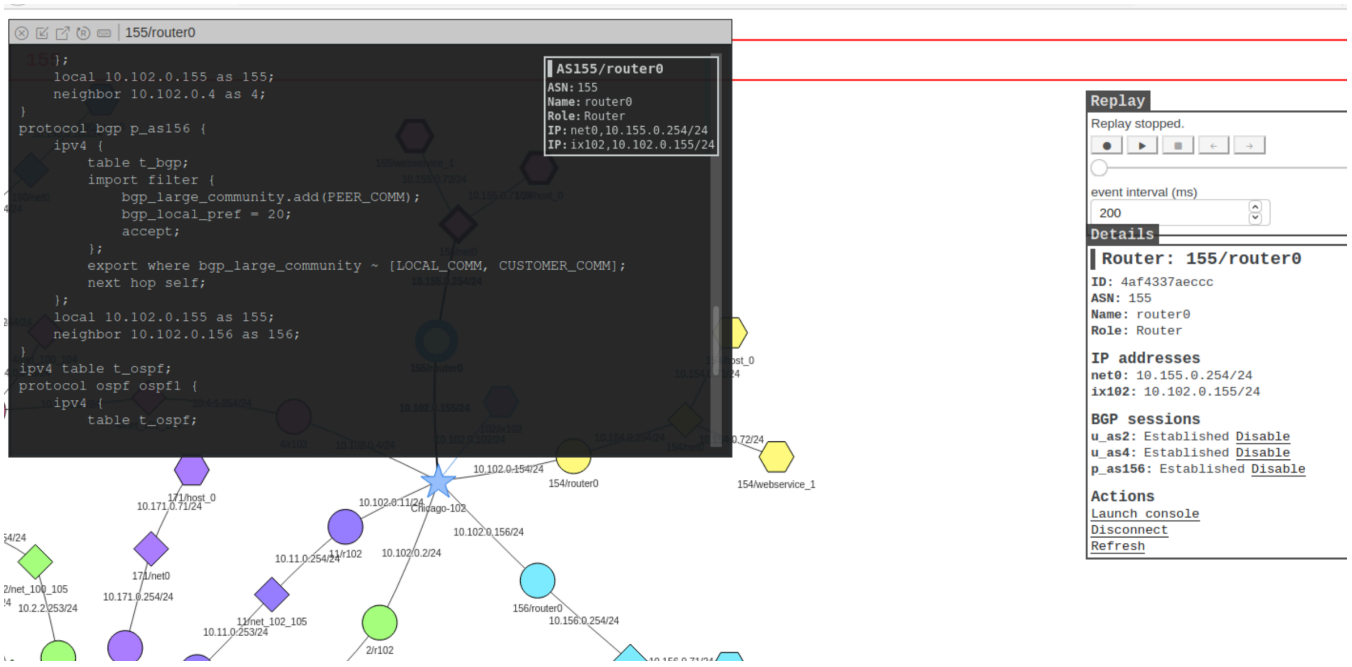
– 在每个网络中，地址从 200 到 255 都是路由器的 IP 地址。对于非路由器（主机），其 IP 地址从 71 开始。例如，在 AS-155 中， 10.155.0.255 是一个 BGP 路由器，而 10.155.0.71则是一个主机。

Task1：Stub自治系统

Task1.a.1:

从 BGP 配置文件中识别 AS-155 与其他哪个 AS 进行互连。

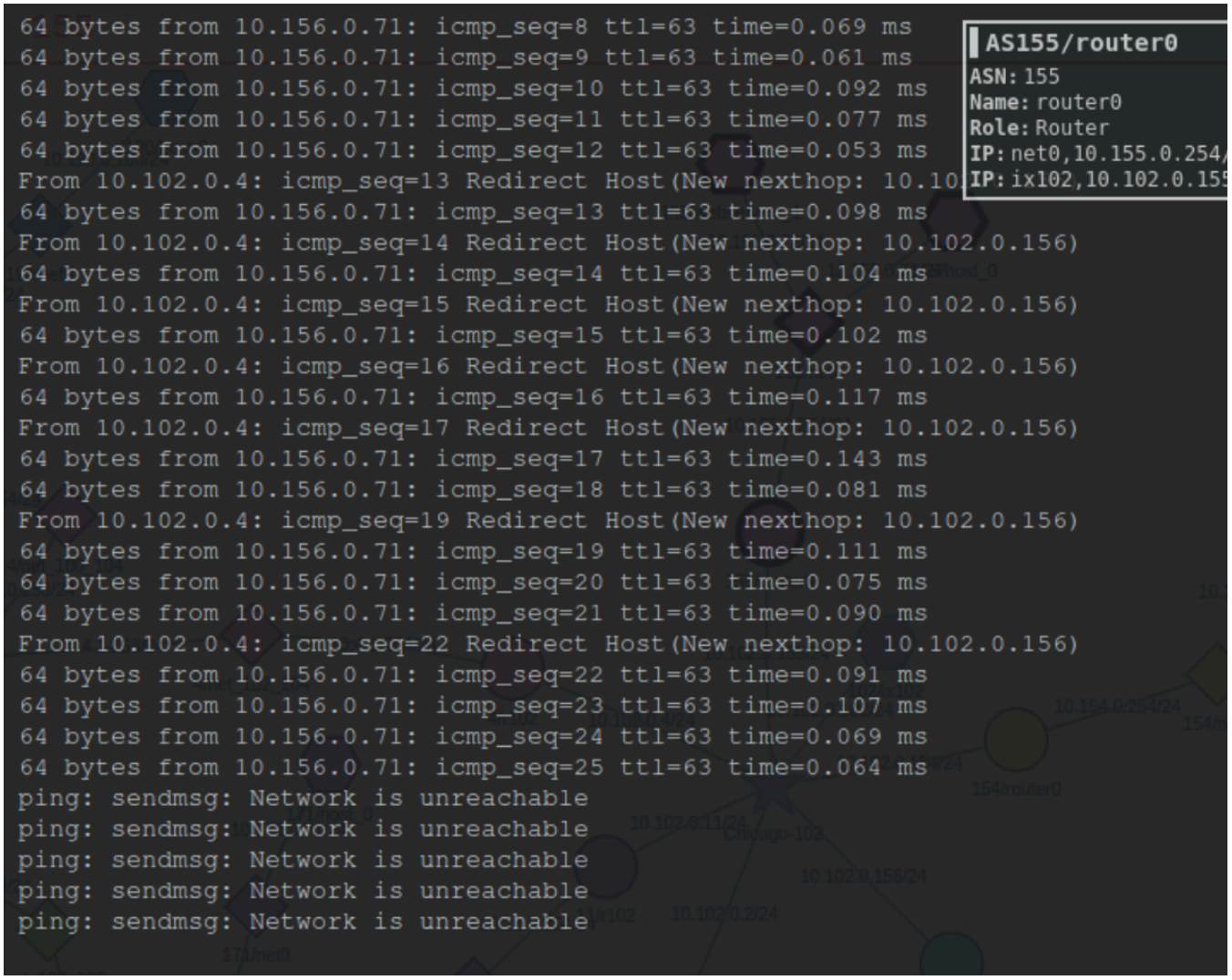
直接搜索155，找到该系统，输入`cat /etc/bird/bird.conf`查看配置文件



可以看到其与三个as互联，其中与p_as156为peer关系

Task1.a.2

让我们在as-155上ping主机10.156.0.71，并尝试切断其他路由链接



由这张图可知，起初一切顺利，切断p_as156与其的联系后，虽然会收到重定向消息，但还是可以ping通的，最后切断所有系统，发现显示不可达。

Task1.b：观察BGP UPDATE消息

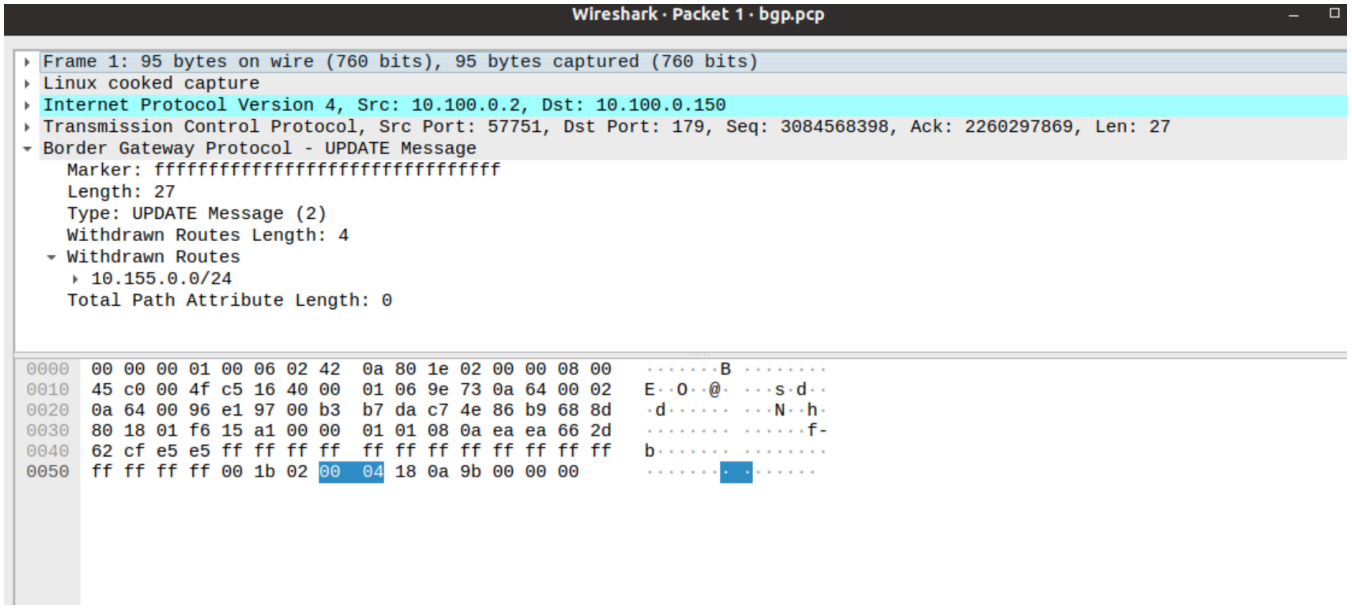
按要求在AS-150的BGP路由器运行tcpdump,命令如下：

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

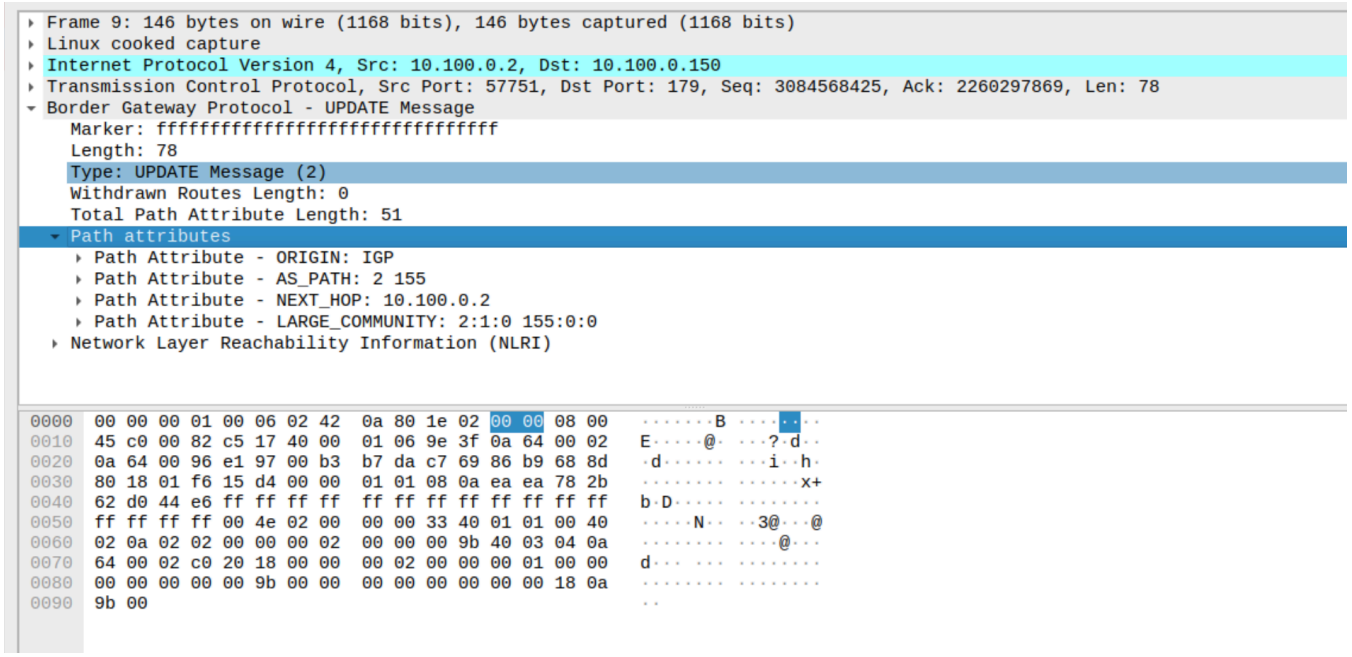
将AS-155与其连接的所有路由关闭再开启，再将报文复制到虚拟机上

```
docker cp 16048:/tmp/bgp.pcap /home/seed/bgp.pcp
```

用wireshark打开查看，可以看到路由退出的报文



也有路由更新的报文



Task1.c试验大型社区

先切断as-4与as-156的连接，然后在10.156.0.72上分别ping 10.156.0.72和10.161.0.71（注意要在主机上做）

```
[08/12/25]seed@VM:~/.../task1$ dockps | grep 10.156.0.72
e3ec8fba1a9e  as156h-webservice_1-10.156.0.72
[08/12/25]seed@VM:~/.../task1$ docksh e3ec
root@e3ec8fba1a9e / # ping 10.156.0.71
PING 10.156.0.71 (10.156.0.71) 56(84) bytes of data.
64 bytes from 10.156.0.71: icmp_seq=1 ttl=64 time=0.124 ms
64 bytes from 10.156.0.71: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.156.0.71: icmp_seq=3 ttl=64 time=0.121 ms
^C
--- 10.156.0.71 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.069/0.104/0.124/0.025 ms
root@e3ec8fba1a9e / # ping 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
^C
--- 10.161.0.71 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2026ms
```

实验手册如此解释：***我们看到 10.155.0.71 仍然可达，因为它属于 AS-155， AS-155 和 AS-156 还是互连的。然而，10.161.0.71（归属于 AS-161）则无法访问，因为没有人会为 AS-156 路由该包。 AS-156 仍然与 AS- 155 进行互连，并且 AS-155 直接连接到互联网，那么为什么 AS-156 不能通过 AS-155 访问互联网呢？这是因为是否转发另一个自治系统中的流量取决于双方之间的业务关系。***

接下来我们要尝试让as-155临时为as-156提供中转服务

修改as-155路由器的配置文件

```
protocol bgp u_as4 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(PROVIDER_COMM);
            bgp_local_pref = 10;
            accept;
        };
        # 添加PEER-COMM，把AS-156路由信息告知AS-4
        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);
            bgp_local_pref = 20;
            accept;
        };
        # 添加PROVIDER_COMM，把AS-4的路由信息告知AS-156
```

```
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM, PROVIDER_COMM];
        next hop self;
    };
    local 10.102.0.155 as 155;
    neighbor 10.102.0.156 as 156;

}
```

再用下列指令重新加载BIRD配置

```
birdc configure
```

再次尝试，顺利ping通

```
1 root@e3ec8fba1a9e / # ping 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
^C
--- 10.161.0.71 ping statistics ---
7 packets transmitted, 0 received, +4 errors, 100% packet loss, time 6141ms

1 root@e3ec8fba1a9e / # ping 10.161.0.71

PING 10.161.0.71 (10.161.0.71) 56(84) bytes of data.
64 bytes from 10.161.0.71: icmp_seq=1 ttl=56 time=0.449 ms
64 bytes from 10.161.0.71: icmp_seq=2 ttl=56 time=0.232 ms
64 bytes from 10.161.0.71: icmp_seq=3 ttl=56 time=0.300 ms
64 bytes from 10.161.0.71: icmp_seq=4 ttl=56 time=0.242 ms
^C
--- 10.161.0.71 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3075ms
rtt min/avg/max/mdev = 0.232/0.305/0.449/0.086 ms
root@e3ec8fba1a9e / #
```

Task1.d：配置AS-180

按要求完成AS-180的BGP路由器及其所有相关的BGP路由器的配置，以实现以下目标：

• 和 AS-171 进行互连，使得他们可以直接互相通信。

• 和 AS-2 和 AS-3 这两个中转自治系统进行互连，以便它们可以通过这些中转到达其他目的地。

将下面的代码加入配置文件

AS-180

```
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;
}
```

as-171

```
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;
}
```

运行另一脚本使配置生效，登录10.180.0.71尝试ping其他主机

```
root@16e5df5a7233 / # ping 10.171.0.71
PING 10.171.0.71 (10.171.0.71) 56(84) bytes of data.
64 bytes from 10.171.0.71: icmp_seq=1 ttl=62 time=0.312 ms
64 bytes from 10.171.0.71: icmp_seq=2 ttl=62 time=0.092 ms
64 bytes from 10.171.0.71: icmp_seq=3 ttl=62 time=0.121 ms
^C
--- 10.171.0.71 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.092/0.175/0.312/0.097 ms
```

再完成第二个要求

在as-180中添加代码

```
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 10.105.0.180 as 180;
    neighbor 10.105.0.2 as 2;
}
```

as-2中添加代码

```
protocol bgp c_as180 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.105.0.2 as 2;
    neighbor 10.105.0.180 as 180;
}
```

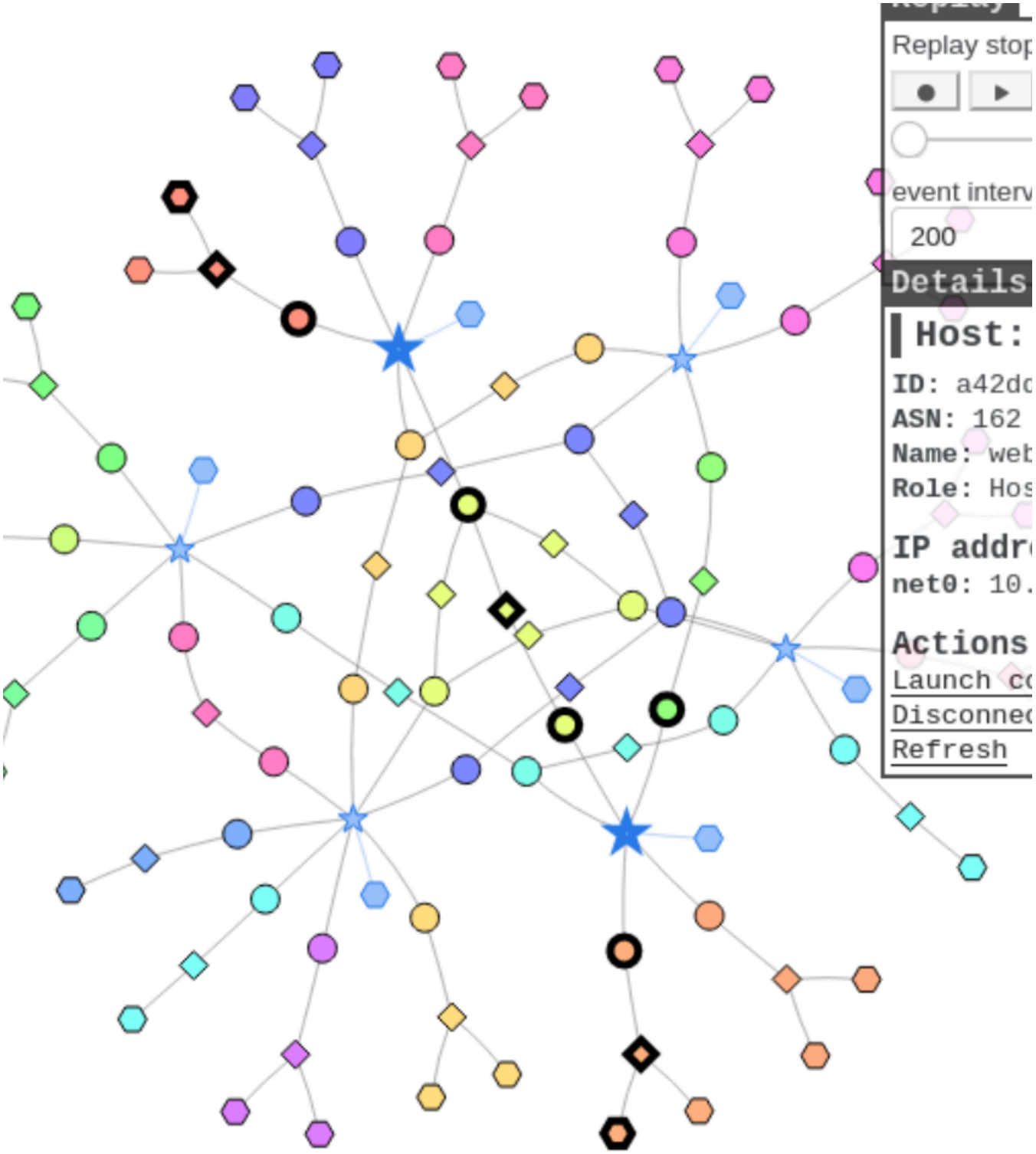
可见已经连接。

至于as-3，与as-2类似，故省略。

Task2：中转自治系统

Task2.a：IBGP的实验

在10.162.0.71上ping10.164.0.71



数据包沿着as-3中转自治系统转发，在地图上关闭IBGP，前后路由表如下

```
root@bfc5bb804a7b / # ip route
10.0.0.5 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.0.0.6 dev dummy0 proto bird scope link metric 32
10.0.0.7 via 10.3.3.253 dev net_103_104 proto bird metric 32
10.0.0.8 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
10.2.2.0/24 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 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
10.3.3.0/24 dev net_103_104 proto kernel scope link src 10.3.3.254
10.3.3.0/24 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 10.3.3.253 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 10.3.3.253 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 10.3.3.253 dev net_103_104 proto bird metric 32
10.153.0.0/24 via 10.3.3.253 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 10.3.3.253 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
root@bfc5bb804a7b / # ip route
10.0.0.5 via 10.3.0.254 dev net_100_103 proto bird metric 32
10.0.0.6 dev dummy0 proto bird scope link metric 32
10.0.0.7 via 10.3.3.253 dev net_103_104 proto bird metric 32
10.0.0.8 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
10.2.2.0/24 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 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
10.3.3.0/24 dev net_103_104 proto kernel scope link src 10.3.3.254
10.3.3.0/24 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 10.3.3.253 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
```

对比后发现，后来路由表少了

```
10.12.0.0/24 via 10.3.3.253 dev net_103_104 proto bird metric 32
10.152.0.0/24 via 10.3.3.253 dev net_103_104 proto bird metric 32
10.153.0.0/24 via 10.3.3.253 dev net_103_104 proto bird metric 32

10.164.0.0/24 via 10.3.3.253 dev net_103_104 proto bird metric 32
```

说明由于ibgp3的关闭，导致这些路径消失了。

Task2.b：IGP试验

还是选择上面的路由做实验，关闭其OSPF，再显示路由表

```
root@bfc5bb804a7b / # ip route
unreachable 10.2.0.0/24 proto bird metric 32
unreachable 10.2.1.0/24 proto bird metric 32
unreachable 10.2.2.0/24 proto bird metric 32
10.3.0.0/24 dev net_100_103 proto kernel scope link src 10.3.0.253
10.3.0.0/24 dev net_100_103 proto bird scope link metric 32
unreachable 10.3.1.0/24 proto bird metric 32
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
10.3.3.0/24 dev net_103_104 proto kernel scope link src 10.3.3.254
10.3.3.0/24 dev net_103_104 proto bird scope link metric 32
unreachable 10.4.0.0/24 proto bird metric 32
unreachable 10.4.1.0/24 proto bird metric 32
unreachable 10.11.0.0/24 proto bird metric 32
unreachable 10.12.0.0/24 proto bird metric 32
10.103.0.0/24 dev ix103 proto kernel scope link src 10.103.0.3
unreachable 10.150.0.0/24 proto bird metric 32
unreachable 10.151.0.0/24 proto bird metric 32
```

```
unreachable 10.152.0.0/24 proto bird metric 32
unreachable 10.153.0.0/24 proto bird metric 32
unreachable 10.154.0.0/24 proto bird metric 32
unreachable 10.155.0.0/24 proto bird metric 32
unreachable 10.156.0.0/24 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
unreachable 10.163.0.0/24 proto bird metric 32
unreachable 10.164.0.0/24 proto bird metric 32
unreachable 10.170.0.0/24 proto bird metric 32
unreachable 10.171.0.0/24 proto bird metric 32
unreachable 10.190.0.0/24 proto bird metric 32
```

会发现大部分路径都失效了

Task2.c：配置AS-5

首先查看AS-5（IX-101）上的IBGP配置

```
protocol bgp ibgp1 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.12 as 5;
    neighbor 10.0.0.13 as 5;
}
protocol bgp ibgp2 {
    ipv4 {
        table t_bgp;
        import all;
        export all;
        igp table t_ospf;
    };
    local 10.0.0.12 as 5;
    neighbor 10.0.0.14 as 5;
}
```

AI有话说：该IBGP配置在**自治系统AS5内部**建立了以路由器`10.0.0.12`为核心的星型拓扑结构，通过两个独立的IBGP会话（`ibgp1`和`ibgp2`）分别连接`10.0.0.13`和`10.0.0.14`，使用`t_bgp`路由表进行全网路由同步，并绑定`t_ospf`IGP路由表解决下一跳可达性问题，实现外部路由在AS内部的无条件传播，实质上构建了一个以中心节点为路由反射器的内部路由分发架构。

修改AS-5的配置

所有as-5配置文件都要加

```
define LOCAL_COMM = (5, 0, 0);
define CUSTOMER_COMM = (5, 1, 0);
define PEER_COMM = (5, 2, 0);
define PROVIDER_COMM = (5, 3, 0);
```

1.101--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 10.0.0.12 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 10.0.0.12 as 5;
}//In the AS-153 file
```

2.103--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 10.0.0.13 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 10.0.0.13 as 5;
}//In the AS-160 file
```

3.105--171

```
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 10.0.0.14 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 10.0.0.14 as 5;
}//In the AS-171 file
```

代码改完后可以在图中找到相应的连接已经connect

此外还要让as-5和as-3在IX-103上互连

```
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;
}//In the AS-5 file
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 10.103.0.3 as 3;
  neighbor 10.0.0.13 as 5;
}//In the AS-3 file
```

此时二者已连接。

Task3：路径选择

Task3.a

查看AS-150的BGP路由器上的所有BGP路径，找到具有多路径的网络前缀，这里以10.163.0.0/24为例

```
10.163.0.0/24        unicast [u_as2 08:54:34.180] * (100) [AS163i]
        via 10.100.0.2 on ix100
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 2 4 163
        BGP.next_hop: 10.100.0.2
        BGP.local_pref: 10
        BGP.large_community: (2, 2, 0) (4, 1, 0) (163, 0, 0) (150, 3, 0)
                     unicast [u_as3 08:54:33.765] (100) [AS163i]
        via 10.100.0.3 on ix100
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path: 3 4 163
        BGP.next_hop: 10.100.0.3
        BGP.local_pref: 10
        BGP.large_community: (3, 2, 0) (4, 1, 0) (163, 0, 0) (150, 3, 0)
```

而查看IP route，会发现转发时会选择第一条路径

```
10.163.0.0/24 via 10.100.0.2 dev ix100 proto bird metric 32
```

由于二者的优先级相同，且AS路径长度似乎也相同，可能是由其他分配机制影响

Task3.b

为了使AS-150平时优先用AS-3，只要在配置中调高优先级即可

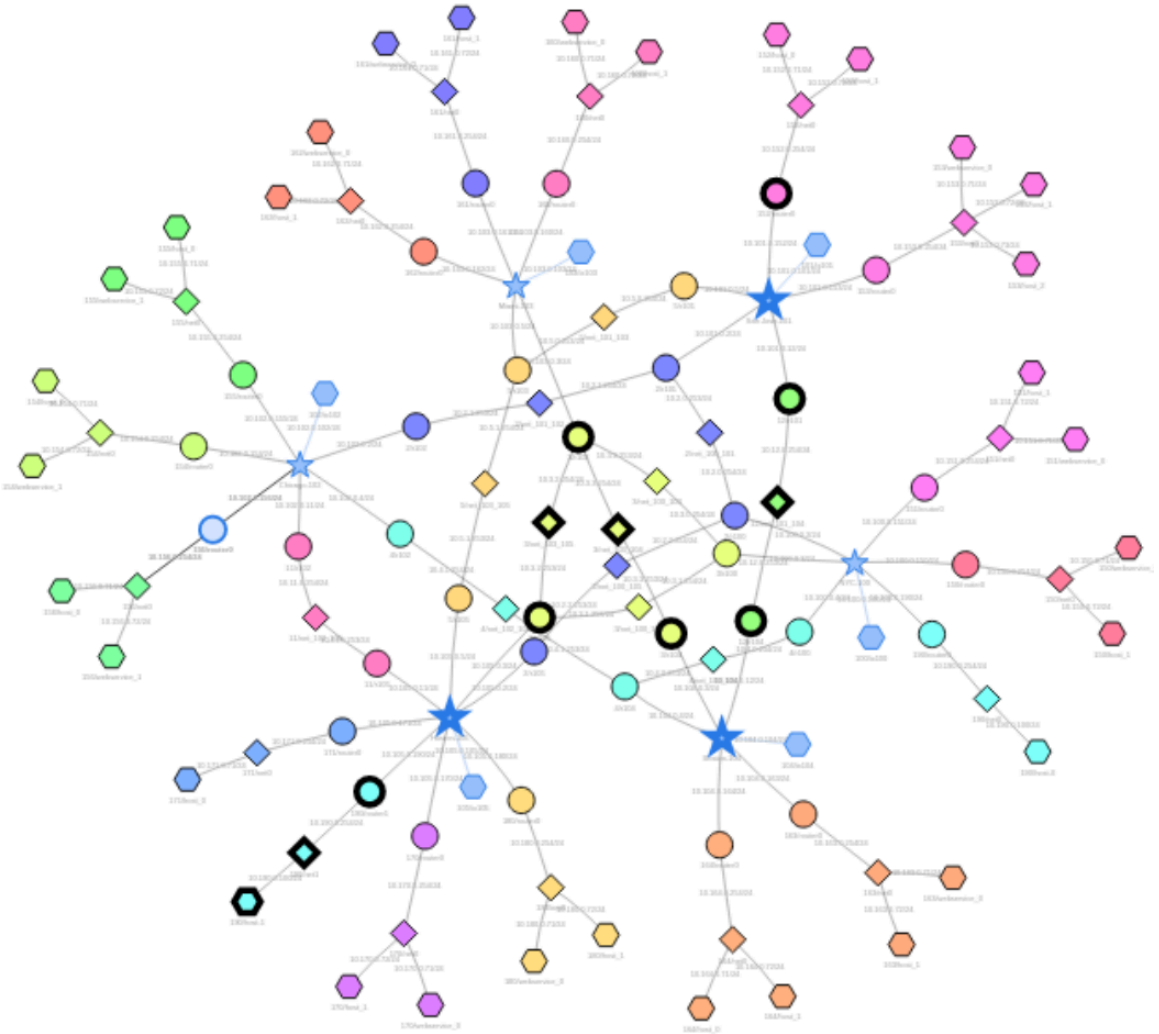
```
protocol bgp u_as3 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(PROVIDER_COMM);
            bgp_local_pref = 15;
            accept;
        };
```

此时再查看ip route

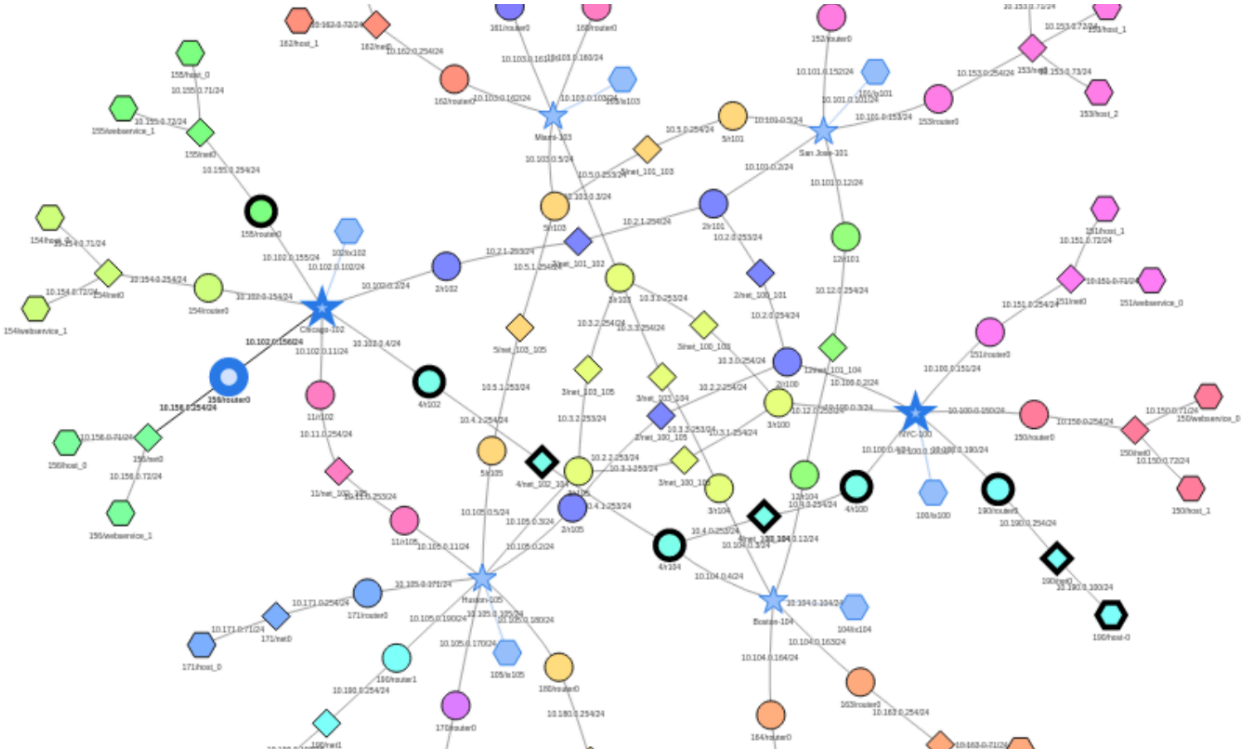
均选择10.100.0.3路径。

Task4：IP任播

从as-152ping10.190.0.100



从as-156ping 10.190.0.100



Anycast的实现机理在于，路由器并不关心目的主机的具体位置（哪怕有多个），而只关心到达主机的路径。两台10.190.0.100分别告知AS-3和AS-4自己的位置，并由AS-3和AS-4向外扩散，其他路由器接收到路由信息后，会根据路由选择算法挑选最优的路径转发，转发路径只有一条，所以消息只能抵达10.190.0.100中某一台主机。

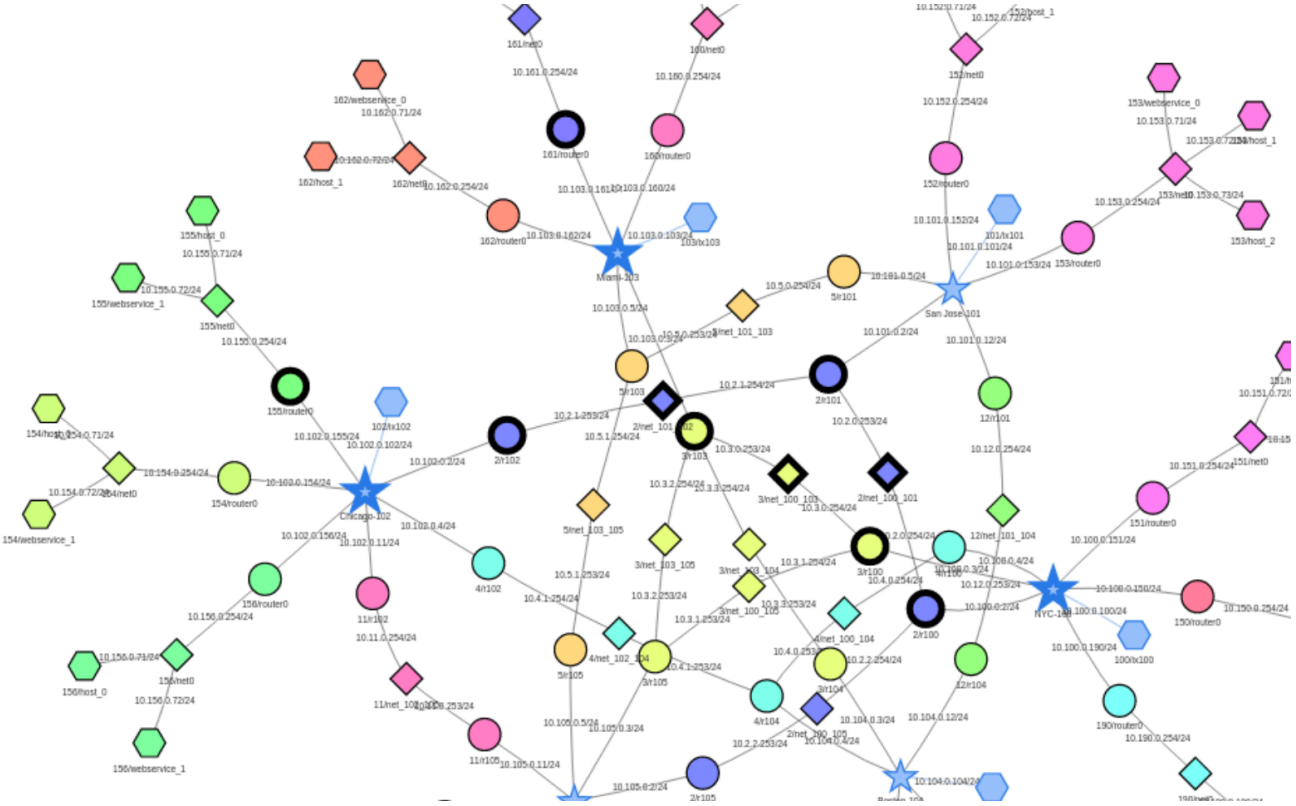
Task5：BGP前缀攻击

Task5.a:使用AS—161的BGP路由器执行前缀劫持攻击

修改as-161的配置信息

```
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);
        };
}
```

让我们在as-155的机器上ping一下10.154.0.71

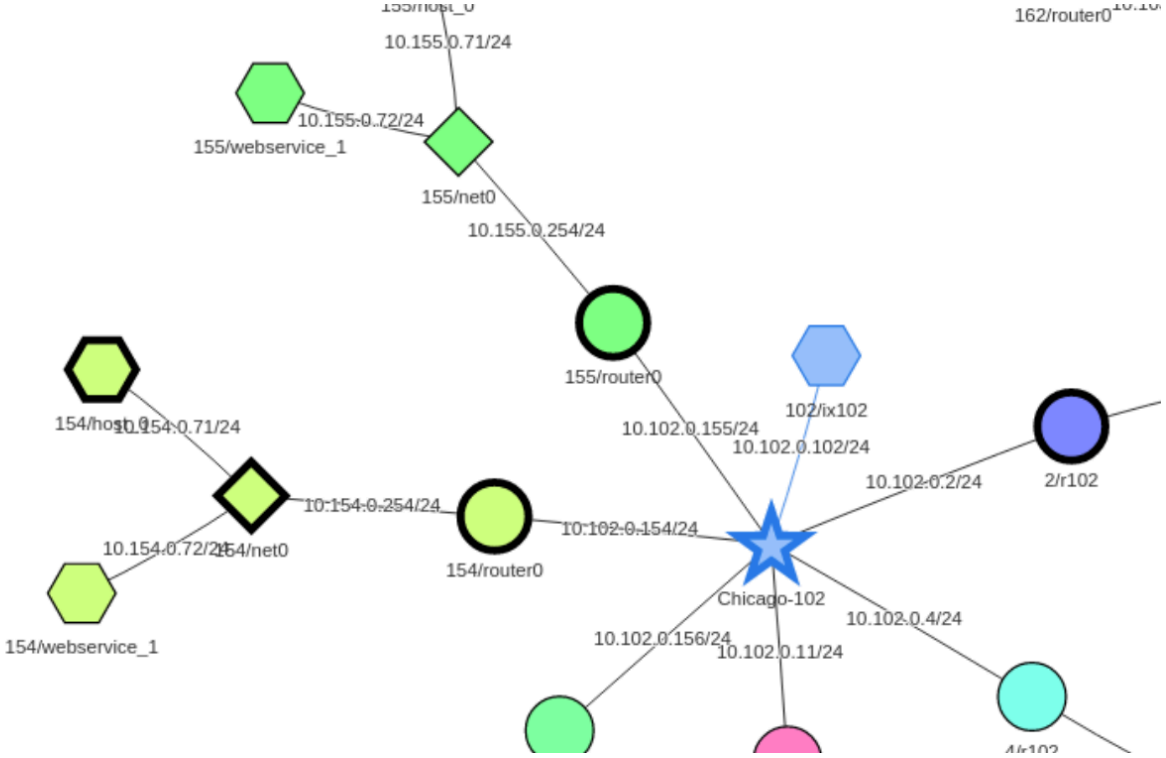


发现数据包全部发往as-161

Task5.b：AS-154的反击

修改As-154的配置，

```
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);
    };
}
```



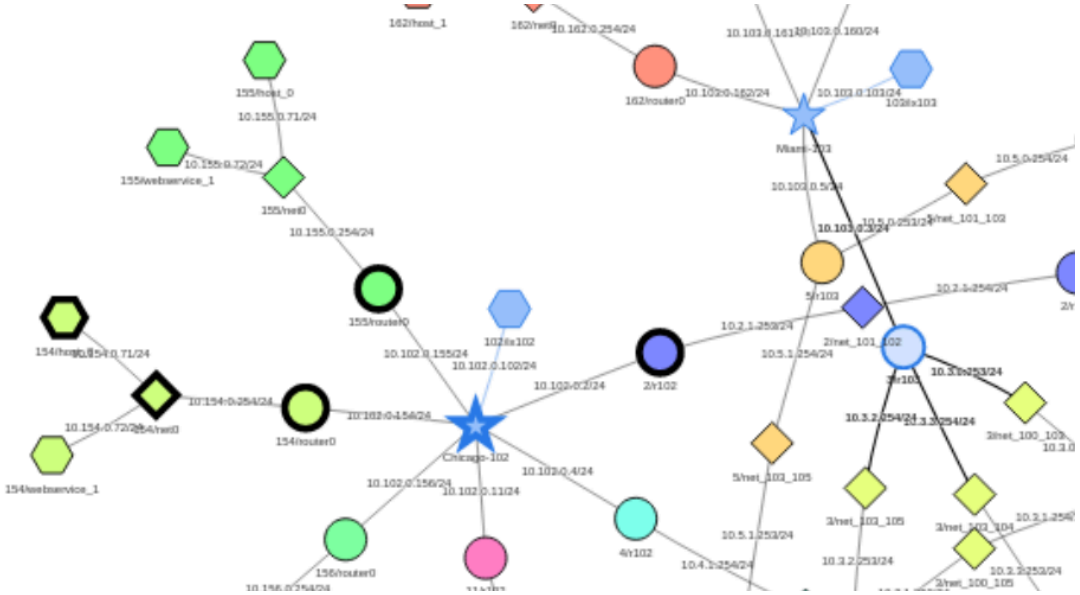
此时夺回了自己的流量

Task5.c：在AS-3处解决问题

修改其过滤器以解决问题

```
protocol bgp c_as161 {
    ipv4 {
        table t_bgp;
```

```
        import filter {
            bgp_large_community.add(CUSTOMER_COMM);
            bgp_local_pref = 30;
            # 添加以下内容
            if (net != 10.161.0.0/24) then reject;
            accept;
        };
        export all;
        next hop self;
    };
    local 10.103.0.3 as 3;
    neighbor 10.103.0.161 as 161;
}
```



此时as-154的反击已撤回，但数据包仍然正常发送到位置，可见问题成功解决

Over!

```
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; 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_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, 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);
            bgp_local_pref = 20;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM, PROVIDER_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;
}
```