Computer Networks
Jacobs University Bremen
MICHAEL MAGAISA

Course: CO20-320301
April 14, 2020

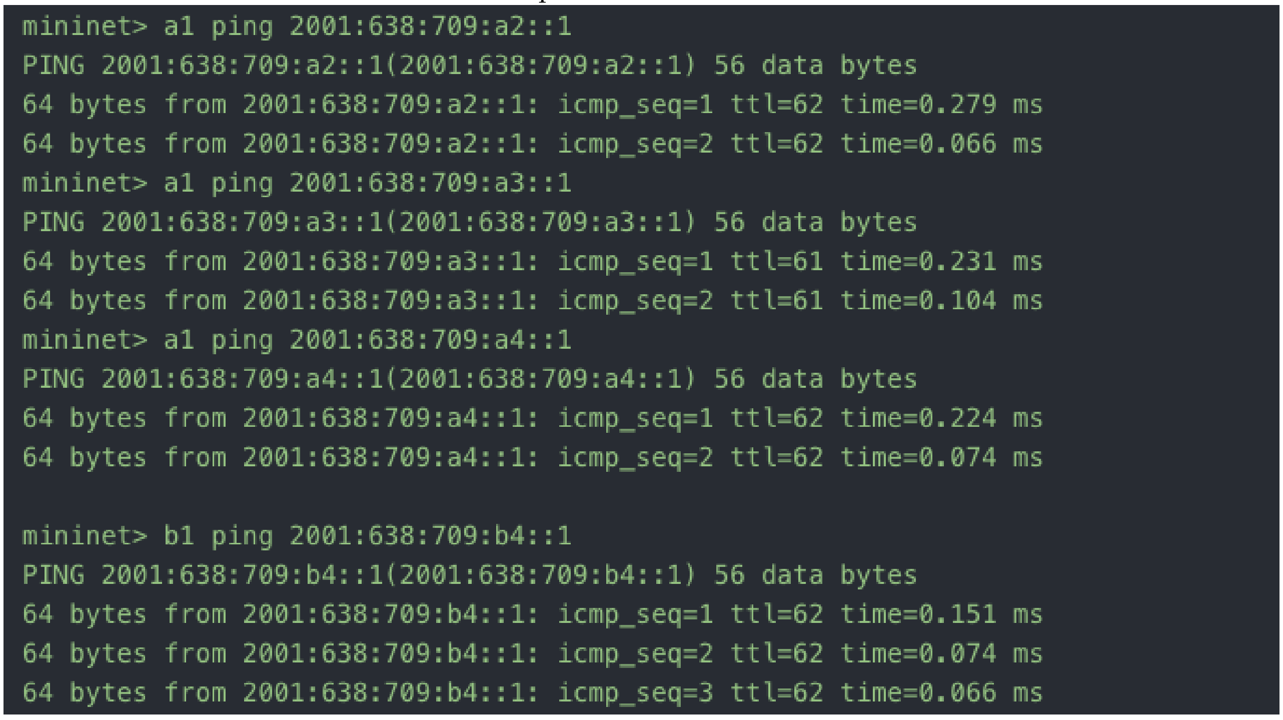
Homework 4

Problem 4.1

Solution:
(a).
In this pdf, results of running the code provided are shown.

Create an OSPF network for ISP F and an OSPF network ISP E. Make sure all hosts Ai can reach each other and all host Bi can reach each other

Screenshot below shows results of some tests performed

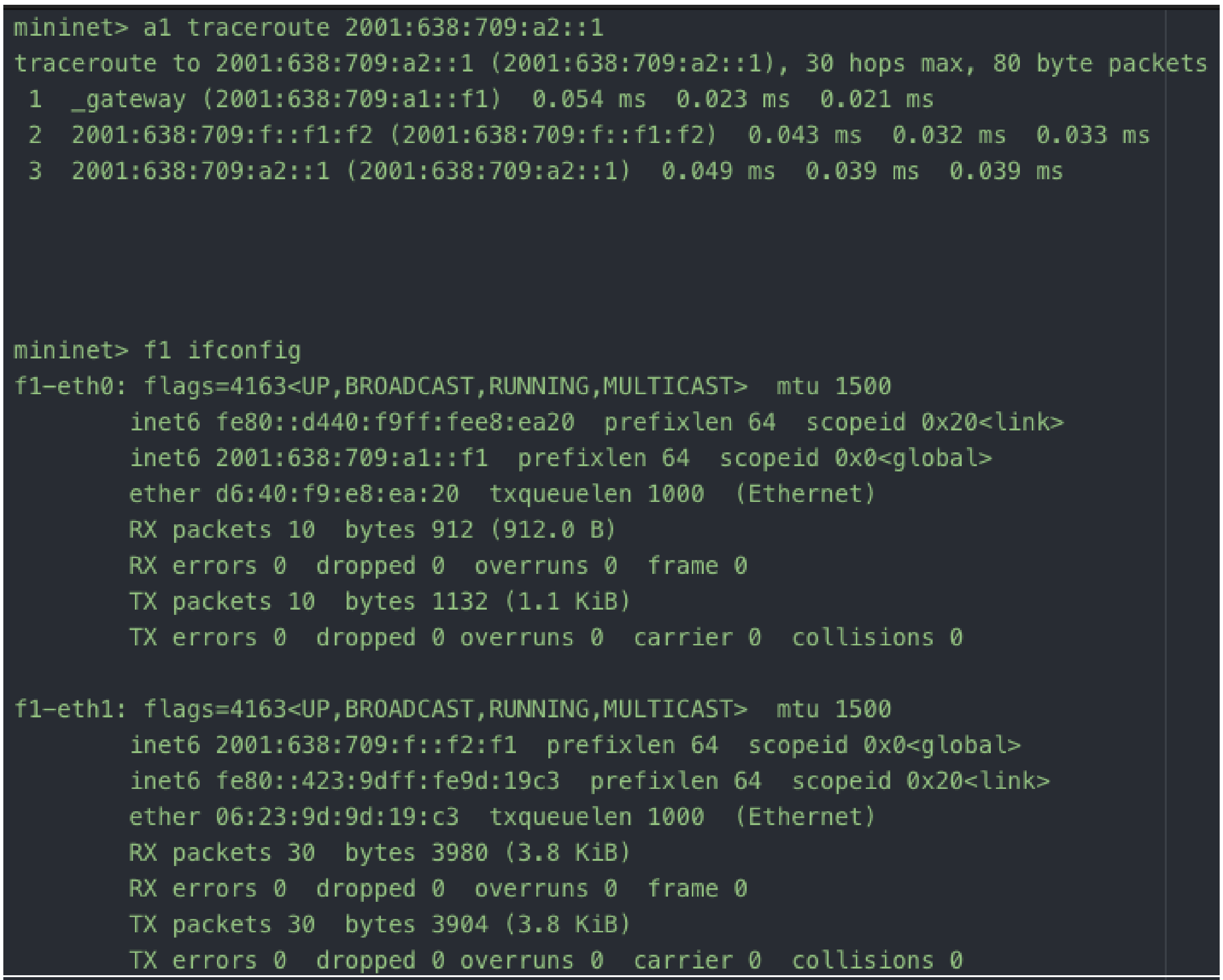


```
mininet> a1 ping 2001:638:709:a2::1
PING 2001:638:709:a2::1(2001:638:709:a2::1) 56 data bytes
64 bytes from 2001:638:709:a2::1: icmp_seq=1 ttl=62 time=0.279 ms
64 bytes from 2001:638:709:a2::1: icmp_seq=2 ttl=62 time=0.066 ms
mininet> a1 ping 2001:638:709:a3::1
PING 2001:638:709:a3::1(2001:638:709:a3::1) 56 data bytes
64 bytes from 2001:638:709:a3::1: icmp_seq=1 ttl=61 time=0.231 ms
64 bytes from 2001:638:709:a3::1: icmp_seq=2 ttl=61 time=0.104 ms
mininet> a1 ping 2001:638:709:a4::1
PING 2001:638:709:a4::1(2001:638:709:a4::1) 56 data bytes
64 bytes from 2001:638:709:a4::1: icmp_seq=1 ttl=62 time=0.224 ms
64 bytes from 2001:638:709:a4::1: icmp_seq=2 ttl=62 time=0.074 ms

mininet> b1 ping 2001:638:709:b4::1
PING 2001:638:709:b4::1(2001:638:709:b4::1) 56 data bytes
64 bytes from 2001:638:709:b4::1: icmp_seq=1 ttl=62 time=0.151 ms
64 bytes from 2001:638:709:b4::1: icmp_seq=2 ttl=62 time=0.074 ms
64 bytes from 2001:638:709:b4::1: icmp_seq=3 ttl=62 time=0.066 ms
```

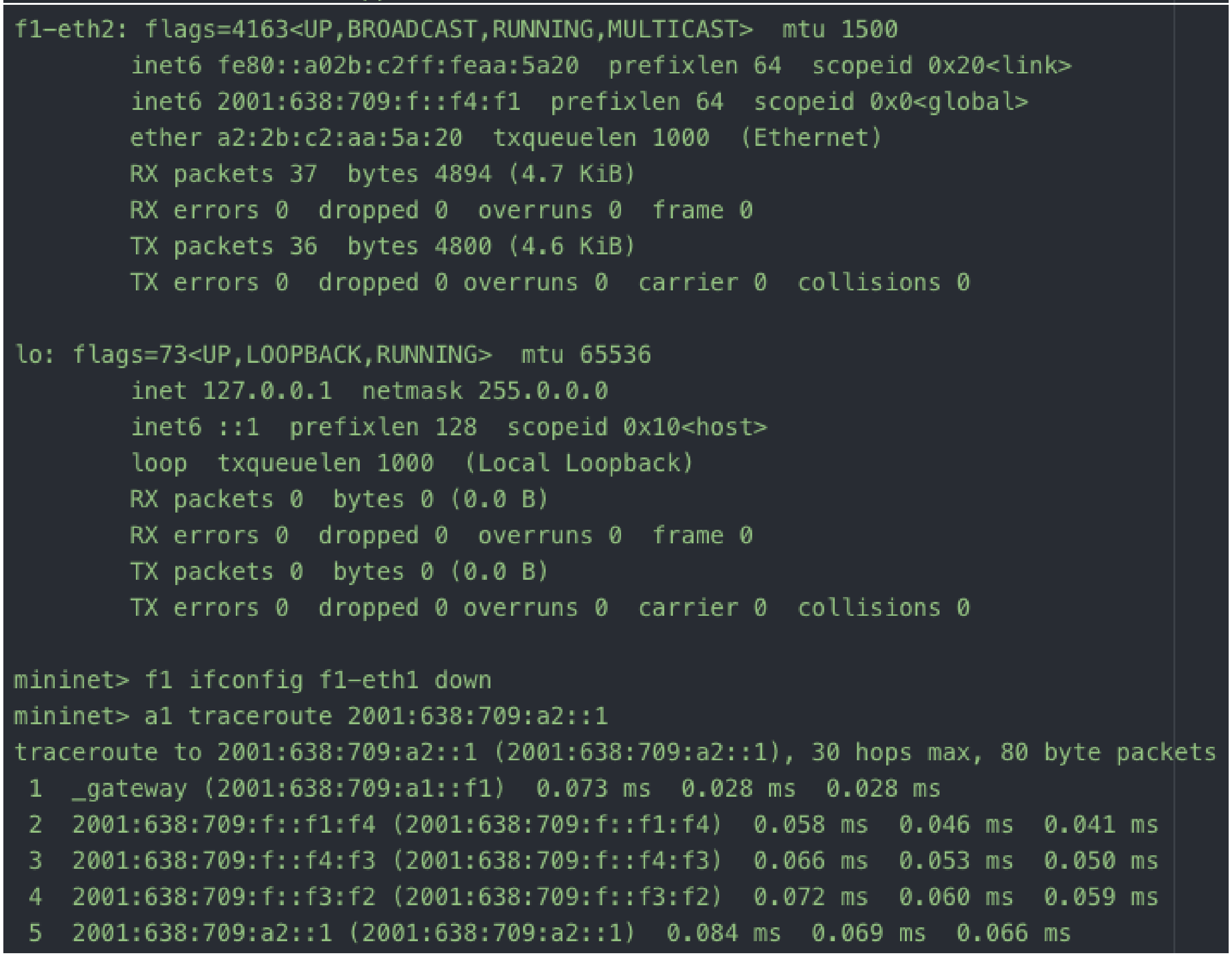
Solution:
(b).

Test that the OSPF networks handle link failures. Take a link down and verify that OSPF calculates alternate paths.

OSPF can be seen trying to calculate an alternate path after taking a link down.The effects of taking f1 down can be seen in the second part of the screenshot.



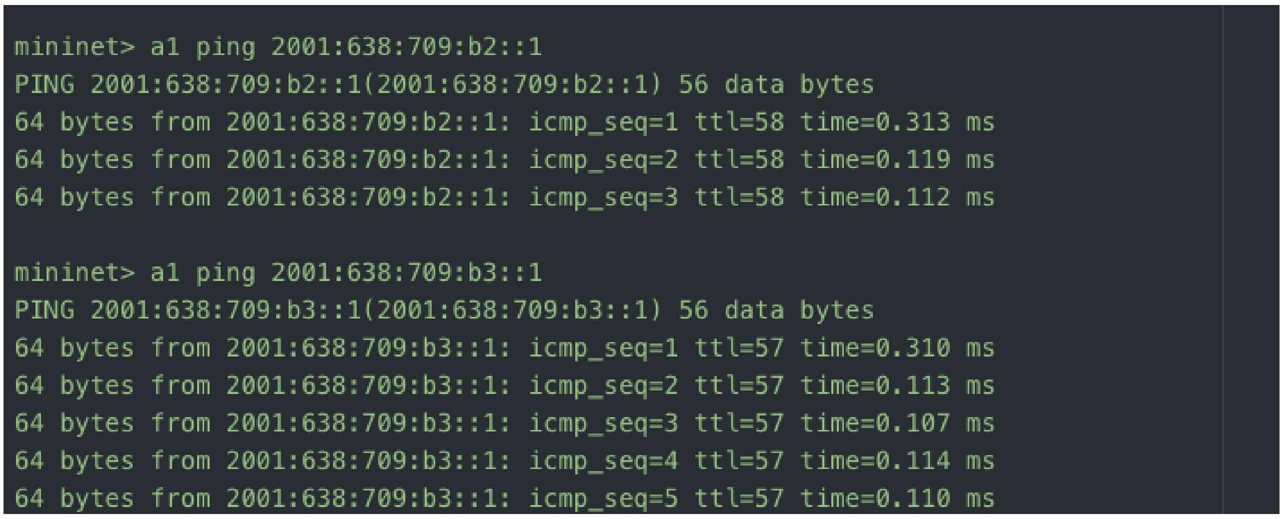
```
mininet> a1 traceroute 2001:638:709:a2::1
traceroute to 2001:638:709:a2::1 (2001:638:709:a2::1), 30 hops max, 80 byte packets
 1  _gateway (2001:638:709:a1::f1)  0.054 ms  0.023 ms  0.021 ms
 2  2001:638:709:f::f1:f2 (2001:638:709:f::f1:f2)  0.043 ms  0.032 ms  0.033 ms
 3  2001:638:709:a2::1 (2001:638:709:a2::1)  0.049 ms  0.039 ms  0.039 ms




mininet> f1 ifconfig
f1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::d440:f9ff:fee8:ea20  prefixlen 64  scopeid 0x20<link>
        inet6 2001:638:709:a1::f1  prefixlen 64  scopeid 0x0<global>
        ether d6:40:f9:e8:ea:20  txqueuelen 1000  (Ethernet)
        RX packets 10  bytes 912 (912.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 1132 (1.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

f1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 2001:638:709:f::f2:f1  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::423:9dff:fe9d:19c3  prefixlen 64  scopeid 0x20<link>
        ether 06:23:9d:9d:19:c3  txqueuelen 1000  (Ethernet)
        RX packets 30  bytes 3980 (3.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 30  bytes 3904 (3.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
f1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::a02b:c2ff:feaa:5a20  prefixlen 64  scopeid 0x20<link>
        inet6 2001:638:709:f::f4:f1  prefixlen 64  scopeid 0x0<global>
        ether a2:2b:c2:aa:5a:20  txqueuelen 1000  (Ethernet)
        RX packets 37  bytes 4894 (4.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 36  bytes 4800 (4.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet> f1 ifconfig f1-eth1 down
mininet> a1 traceroute 2001:638:709:a2::1
traceroute to 2001:638:709:a2::1 (2001:638:709:a2::1), 30 hops max, 80 byte packets
 1  _gateway (2001:638:709:a1::f1)  0.073 ms  0.028 ms  0.028 ms
 2  2001:638:709:f::f1:f4 (2001:638:709:f::f1:f4)  0.058 ms  0.046 ms  0.041 ms
 3  2001:638:709:f::f4:f3 (2001:638:709:f::f4:f3)  0.066 ms  0.053 ms  0.050 ms
 4  2001:638:709:f::f3:f2 (2001:638:709:f::f3:f2)  0.072 ms  0.060 ms  0.059 ms
 5  2001:638:709:a2::1 (2001:638:709:a2::1)  0.084 ms  0.069 ms  0.066 ms
```

Solution:
(c).

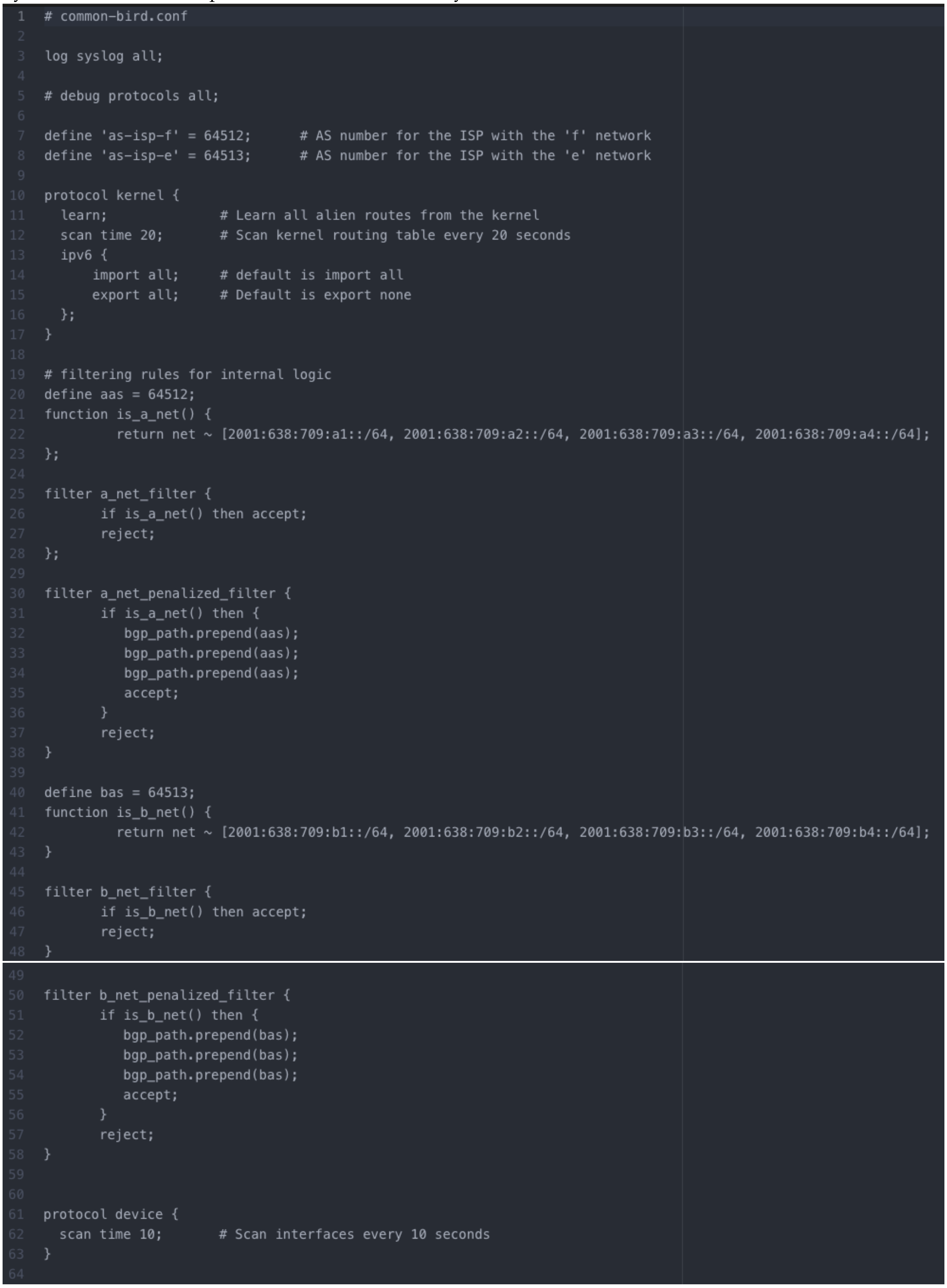
ISP F and ISP E create two BGP peerings. Configure the two BGP peering sessions



```
mininet> a1 ping 2001:638:709:b2::1
PING 2001:638:709:b2::1(2001:638:709:b2::1) 56 data bytes
64 bytes from 2001:638:709:b2::1: icmp_seq=1 ttl=58 time=0.313 ms
64 bytes from 2001:638:709:b2::1: icmp_seq=2 ttl=58 time=0.119 ms
64 bytes from 2001:638:709:b2::1: icmp_seq=3 ttl=58 time=0.112 ms

mininet> a1 ping 2001:638:709:b3::1
PING 2001:638:709:b3::1(2001:638:709:b3::1) 56 data bytes
64 bytes from 2001:638:709:b3::1: icmp_seq=1 ttl=57 time=0.310 ms
64 bytes from 2001:638:709:b3::1: icmp_seq=2 ttl=57 time=0.113 ms
64 bytes from 2001:638:709:b3::1: icmp_seq=3 ttl=57 time=0.107 ms
64 bytes from 2001:638:709:b3::1: icmp_seq=4 ttl=57 time=0.114 ms
64 bytes from 2001:638:709:b3::1: icmp_seq=5 ttl=57 time=0.110 ms
```

Solution:
(d). ISP F is interested to announce his customer networks to ISP E but he prefers to not announce his internal network to ISP E. Similarly, ISP F is interested to announce his customer networks to ISP F but he prefers to not announce his internal network to ISP F. Create filters implementing these policies.

The filters implementing the given policies are shown in the screenshot of common-bird.conf file below, whose code is also provided seperately. This file also contains additional filters to meet the conditions that ISP F prefers to use routes announced by E1 with a fallback option to routes announced by E4 in case the link to E1 fails and Similarly, ISP E prefers to use routes announced by F3 with a fall- back option to routes announced by F2 in case the link to F3 fails

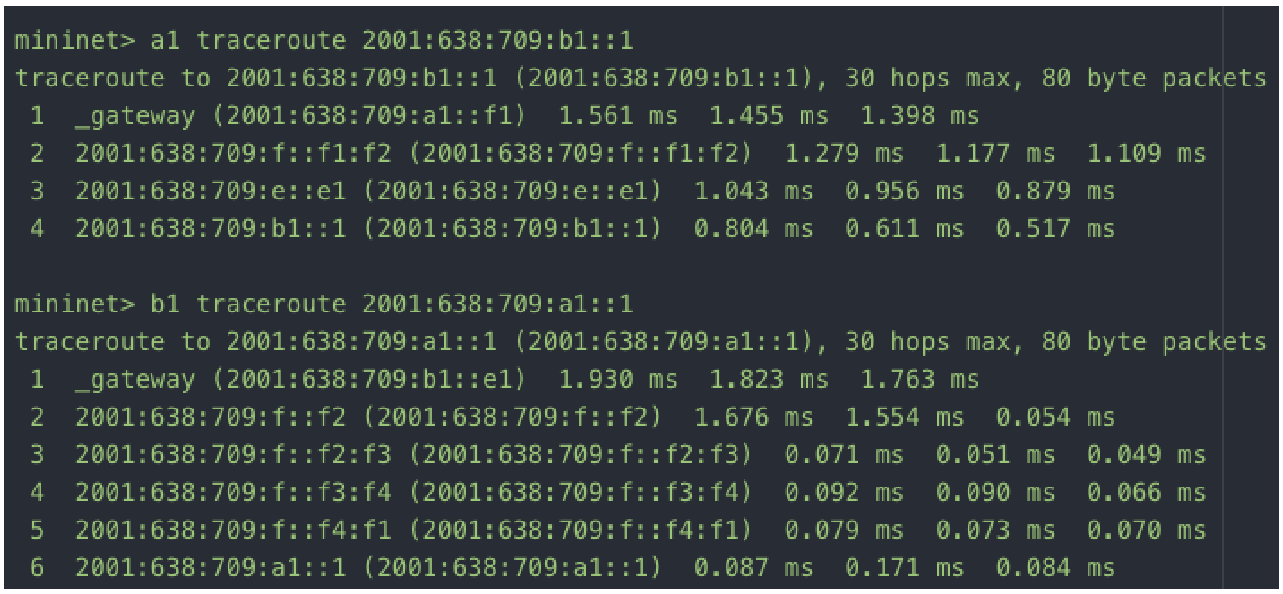


```
# common-bird.conf

log syslog all;

# debug protocols all;

define 'as-isp-f' = 64512;      # AS number for the ISP with the 'f' network
define 'as-isp-e' = 64513;      # AS number for the ISP with the 'e' network

protocol kernel {
  learn;              # Learn all alien routes from the kernel
  scan time 20;       # Scan kernel routing table every 20 seconds
  ipv6 {
      import all;     # default is import all
      export all;     # Default is export none
  };
}

# filtering rules for internal logic
define aas = 64512;
function is_a_net() {
         return net ~ [2001:638:709:a1::/64, 2001:638:709:a2::/64, 2001:638:709:a3::/64, 2001:638:709:a4::/64];
};

filter a_net_filter {
       if is_a_net() then accept;
       reject;
};

filter a_net_penalized_filter {
       if is_a_net() then {
          bgp_path.prepend(aas);
          bgp_path.prepend(aas);
          bgp_path.prepend(aas);
          accept;
       }
       reject;
}

define bas = 64513;
function is_b_net() {
         return net ~ [2001:638:709:b1::/64, 2001:638:709:b2::/64, 2001:638:709:b3::/64, 2001:638:709:b4::/64];
}

filter b_net_filter {
       if is_b_net() then accept;
       reject;
}

filter b_net_penalized_filter {
       if is_b_net() then {
          bgp_path.prepend(bas);
          bgp_path.prepend(bas);
          bgp_path.prepend(bas);
          accept;
       }
       reject;
}


protocol device {
  scan time 10;       # Scan interfaces every 10 seconds
}

```

Solution:
(e).

ISP F prefers to use routes announced by E1 with a fallback option to routes announced by E4 in case the link to E1 fails. Similarly, ISP E prefers to use routes announced by F3 with a fall- back option to routes announced by F2 in case the link to F3 fails. Create filters implementing these policies and verify that they work correctly.

check the screenshot above for the filters, while the verification is provided below:



```
mininet> a1 traceroute 2001:638:709:b1::1
traceroute to 2001:638:709:b1::1 (2001:638:709:b1::1), 30 hops max, 80 byte packets
 1  _gateway (2001:638:709:a1::f1)  1.561 ms  1.455 ms  1.398 ms
 2  2001:638:709:f::f1:f2 (2001:638:709:f::f1:f2)  1.279 ms  1.177 ms  1.109 ms
 3  2001:638:709:e::e1 (2001:638:709:e::e1)  1.043 ms  0.956 ms  0.879 ms
 4  2001:638:709:b1::1 (2001:638:709:b1::1)  0.804 ms  0.611 ms  0.517 ms

mininet> b1 traceroute 2001:638:709:a1::1
traceroute to 2001:638:709:a1::1 (2001:638:709:a1::1), 30 hops max, 80 byte packets
 1  _gateway (2001:638:709:b1::e1)  1.930 ms  1.823 ms  1.763 ms
 2  2001:638:709:f::f2 (2001:638:709:f::f2)  1.676 ms  1.554 ms  0.054 ms
 3  2001:638:709:f::f2:f3 (2001:638:709:f::f2:f3)  0.071 ms  0.051 ms  0.049 ms
 4  2001:638:709:f::f3:f4 (2001:638:709:f::f3:f4)  0.092 ms  0.090 ms  0.066 ms
 5  2001:638:709:f::f4:f1 (2001:638:709:f::f4:f1)  0.079 ms  0.073 ms  0.070 ms
 6  2001:638:709:a1::1 (2001:638:709:a1::1)  0.087 ms  0.171 ms  0.084 ms
```

References :

https://www.slashroot.in/how-does-traceroute-work-and-examples-using-traceroute-comm
http://www.rfc-editor.org/rfc/rfc1191.txt
https://cnds.jacobs-university.de/courses/cn-2019/p4.pdf