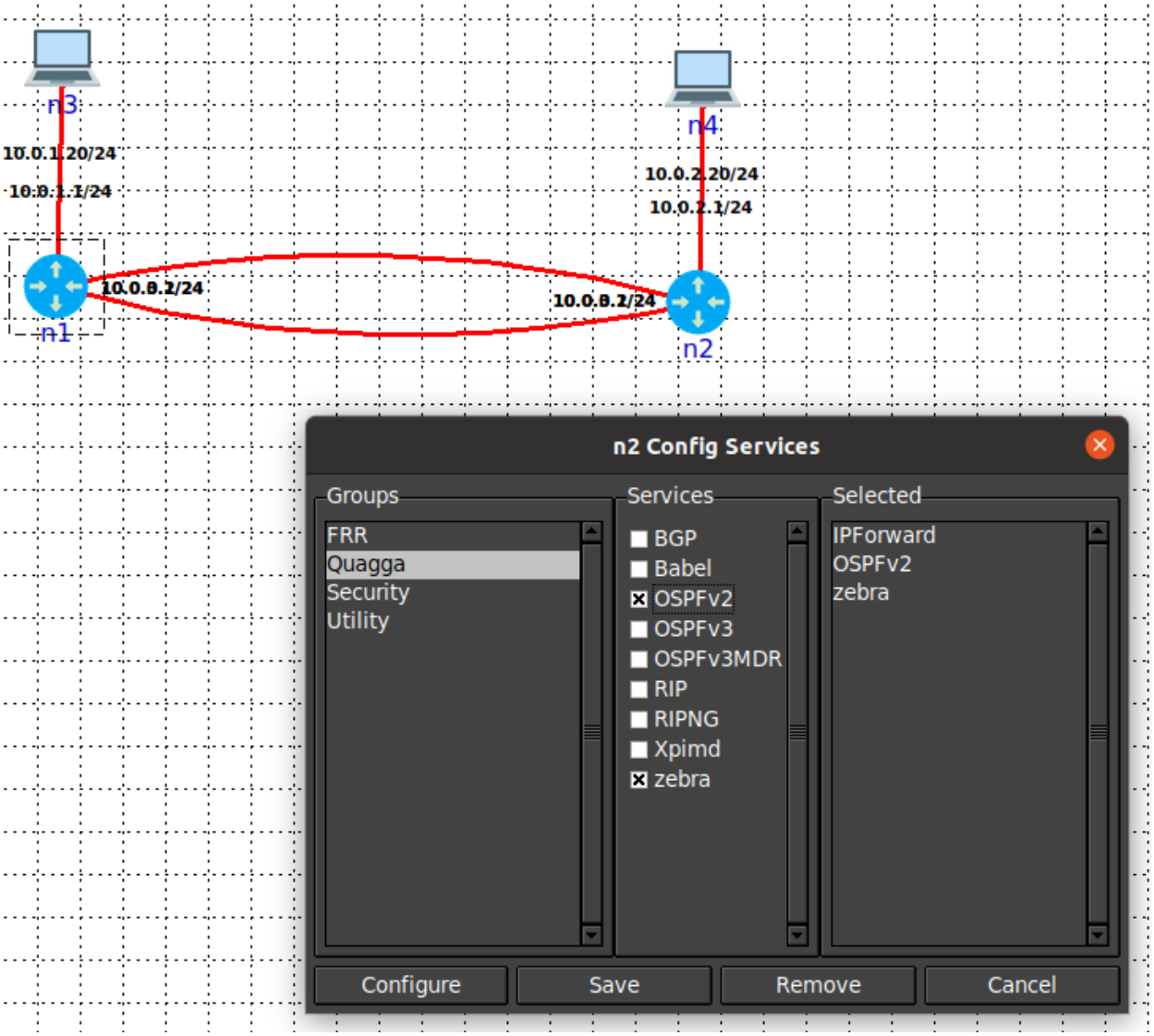
HW6 (18 points)

Tools:

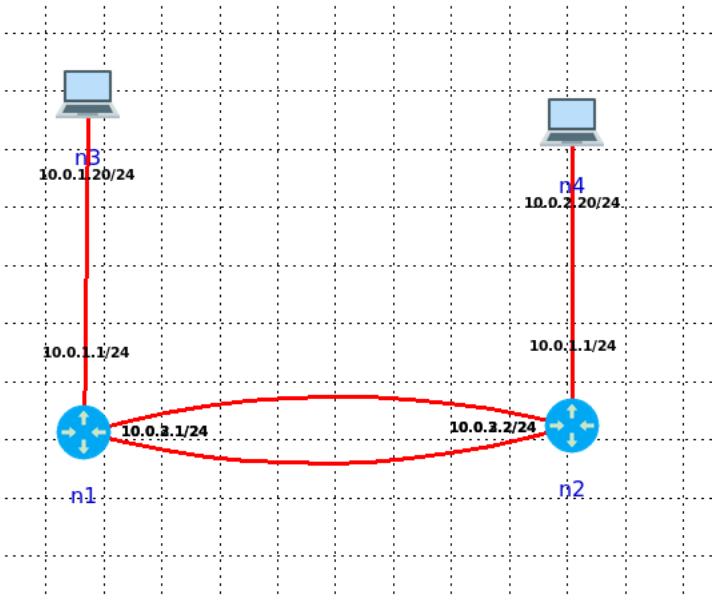
- FRR manual: https://docs.frrouting.org/en/latest/ospfd.html

Setup:

1) Setup a CORE scenario as shown below. Note that nodes n1 and n2 have two links between them. Configure the Quagga OSPFv2 service to be running on both nodes n1 and n2 (turn off OSPFv3).



Part 1: Observing Adjacencies (2points)

1) Start the CORE scenario.
   a. 

2) Run the command “vtysh” on node n1. That puts you on a console that is similar to a Cisco router. Look at the FRR documentation above. FRR is an opensource routing engine. Although the routing engine we ran from the above picture is called “Quagga”, FRR is a fork off of Quagga and is a lot more supported by the community.
3) Type “?” to get list of commands (you can do this anytime)
   a.
```
Hello, this is Quagga (version 0.99.21mr2.2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n1#
  clear         Reset functions
  configure     Configuration from vty interface
  copy          Copy from one file to another
  debug         Enable debug messages for specific or all part.
  disable       Turn off privileged mode command
  end           End current mode and change to enable mode
  exit          Exit current mode and down to previous mode
  list          Print command list
  no            Negate a command or set its defaults
  ping          Send echo messages
  quit          Exit current mode and down to previous mode
  show          Show running system information
  ssh           Open an ssh connection
  start-shell   Start UNIX shell
  telnet        Open a telnet connection
  terminal      Set terminal line parameters
  traceroute    Trace route to destination
  undebug       Disable debugging functions (see also 'debug')
  write         Write running configuration to memory, network, or terminal
n1#
```
4) We will only use “show” and “configure” commands. The command “show” is for displaying current configs and statistics, “configure” is for editing running configuration
5) Run the command “show run” to shows the actively running config of the router
6) (0.5pts) What is the router-id value used for?
   a. Router-id value is it’s address in subnet. Used as the name of the OSPF router.
7) (0.5pts) Show a screenshot of the running config

a.

```
n1# show run
Building configuration...

Current configuration:
!
!
service integrated-vtysh-config
!
interface eth0
 ip address 10.0.0.1/24
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
interface eth1
 ip address 10.0.1.1/24
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
interface eth2
 ip address 10.0.3.2/24
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
router ospf
 ospf router-id 10.0.0.1
 network 10.0.0.1/24 area 0.0.0.0
 network 10.0.1.1/24 area 0.0.0.0
 network 10.0.3.2/24 area 0.0.0.0
!
ip forwarding
ipv6 forwarding
!
line vty
!
```

8) Run the command “show ip ospf neighbor” to show the adjacency list of each router
9) (0.5pts) Show a screenshot of the list of adjacencies.

a.

```
n1# show ip ospf neighbor

    Neighbor ID Pri State           Dead Time Address         Interface            RXmtL RqstL DBsmL
10.0.0.2          1 Full/DROther      5.351s 10.0.0.2         eth0:10.0.0.1            0     0     0
10.0.0.2          1 Full/DROther      5.351s 10.0.3.1         eth2:10.0.3.2            0     0     0
n1#
```

b.

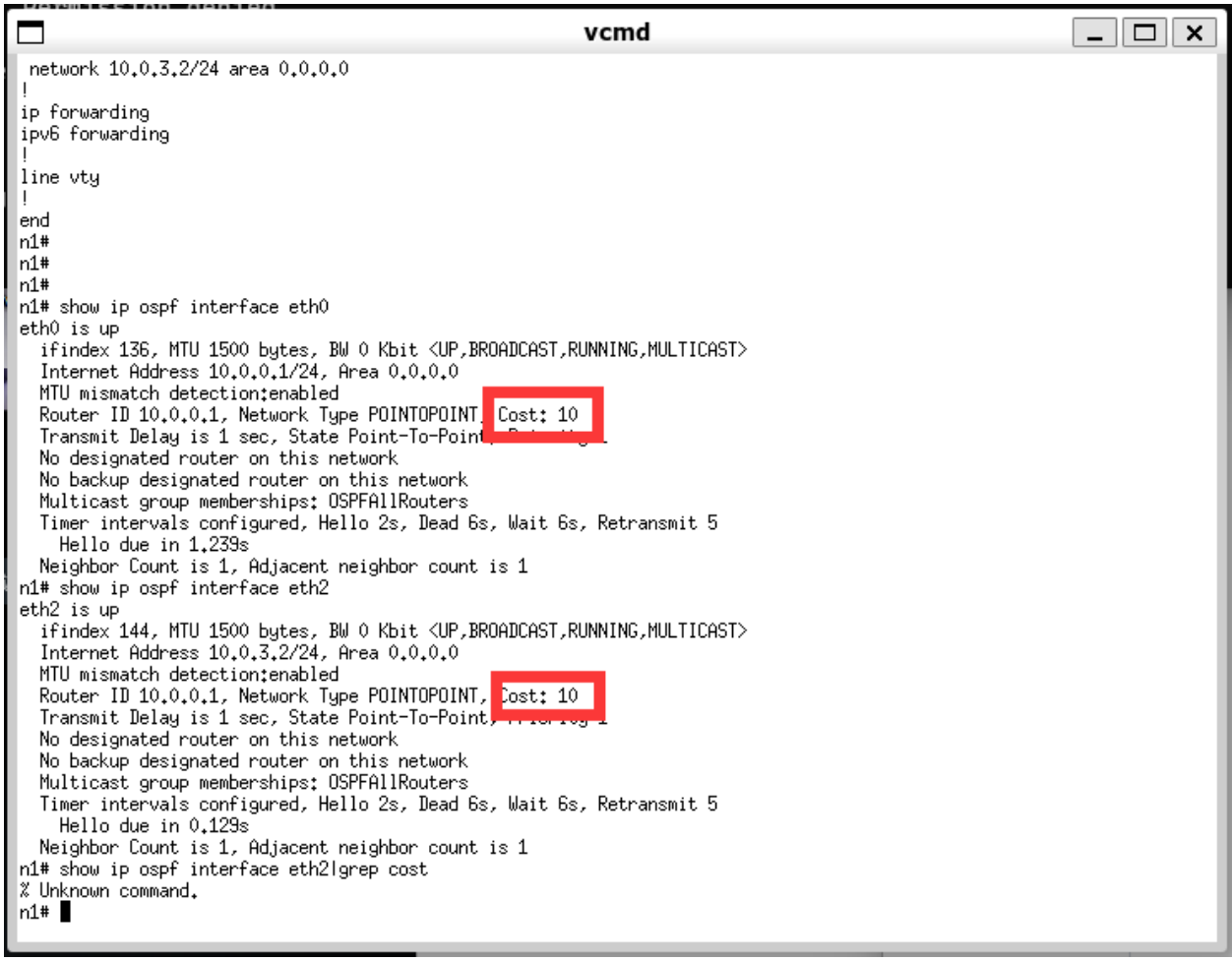
```
n2# show ip ospf neighbor

    Neighbor ID Pri State           Dead Time Address         Interface            RXmtL RqstL DBsmL
10.0.0.1          1 Full/DROther      4.296s 10.0.0.1         eth0:10.0.0.2            0     0     0
10.0.0.1          1 Full/DROther      4.296s 10.0.3.2         eth2:10.0.3.1            0     0     0
n2#
```

10) (0.5pts) What is an adjacency? Where does the neighbor ID in the adjacency come from?
    a. Is a router with direct link to the current router. Neighbor ID comes from the router-id of the neighbor router.
11) Observe the value of the dead timer for 30 seconds. Describe what happens to it? The dead timer on an adjacency resets every time a hello message is received from the neighbor over which this adjacency is established (come back to this when you do step 36)
    a. The dead time ranges from 4-6 second. Seemed that the dead timer is 6 second counting down. And when it receives a hello packet around 2 second, the timer reset to 6 second.

Part 2: Observing Costs (2points)

12) Run the command “show ip ospf interface IFNAME” to show the cost of each link where there is an adjacency (replace IFNAME with the actual interface name from 8)
13) (0.5pts) Show a screenshot of the output from 12 highlighting the link costs

a.



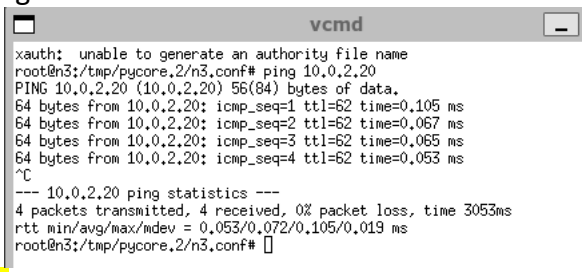
14) Run the command “show ip ospf route” to show the routing table

a.

```
n1# show ip ospf route
============ OSPF network routing table ============
N    10.0.0.0/24          [10] area: 0.0.0.0
                          directly attached to eth0
N    10.0.1.0/24          [10] area: 0.0.0.0
                          directly attached to eth1
N    10.0.2.0/24          [20] area: 0.0.0.0
                          via 10.0.0.2, eth0
                          via 10.0.3.1, eth2
N    10.0.3.0/24          [10] area: 0.0.0.0
                          directly attached to eth2

============ OSPF router routing table =============

============ OSPF external routing table ===========
```

15) Do a ping between the two hosts n3 and n4

a.



16) (0.5pts) Which of the two links connecting n1 and n2 is the ping request traffic going over? Show a screenshot to support your answer

a. It’s using the upper line(10.0.0.1)

b.

```
root@n3:/tmp/pycore.2/n3.conf# traceroute 10.0.2.20
traceroute to 10.0.2.20 (10.0.2.20), 30 hops max, 60 byte packets
 1  10.0.1.1 (10.0.1.1)  0.031 ms  0.022 ms  0.006 ms
 2  10.0.0.2 (10.0.0.2)  0.018 ms  0.009 ms  0.008 ms
 3  10.0.2.20 (10.0.2.20)  0.037 ms  0.011 ms  0.010 ms
root@n3:/tmp/pycore.2/n3.conf#
```

17) (0.5pts) Which of the two links connecting n2 and n1 is the ping reply traffic going over? Show a screenshot to support your answer
    a. It's also the upper line(10.0.2.1)

b.

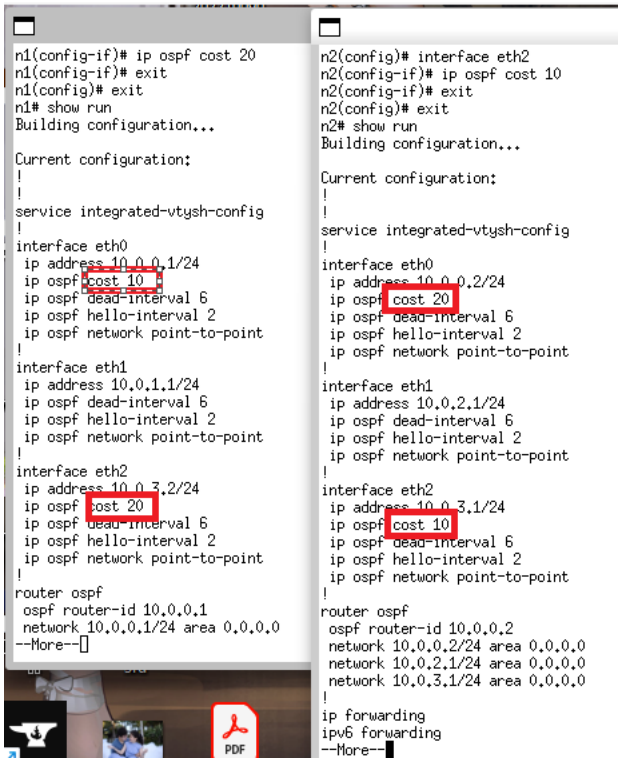
```
root@n4:/tmp/pycore.2/n4.conf# traceroute 10.0.1.20
traceroute to 10.0.1.20 (10.0.1.20), 30 hops max, 60 byte packets
 1  10.0.2.1 (10.0.2.1)  0.028 ms  0.007 ms  0.005 ms
 2  10.0.0.1 (10.0.0.1)  0.016 ms  0.007 ms  0.006 ms
 3  10.0.1.20 (10.0.1.20)  0.018 ms  0.009 ms  0.010 ms
root@n4:/tmp/pycore.2/n4.conf#
```

18) (0.5pts) Can you justify the routes that OSPF picked?
    a. It could because the cost of two link is equal. So it picked the one in higher alphabetical order.

Part 3: Modifying Costs (7points)

19) Do the below on node n1
20) Run the command "configure terminal" to go into config edit mode. This allows us to change the running configuration of the router. Console now should have "(config)"
21) Let's assume the top link connecting n1 and n2 is called eth100 (get actual interface name from 8)
22) Run the command "interface eth100" to edit config of the top link
23) Run "ip ospf cost 10" to set the interface cost to 10
24) Run "exit" to exit configure mode of the link
25) Repeat 22-24 so you have a setting where top link cost is 10 and bottom link cost is 20 on node n1
26) Run vtysh on node n2 and do 20-24 so you have a setting where top link cost is 20 and bottom link cost is 10
27) Run "exit" to exit configure mode on both nodes (you might need to do that more than once until "(config)" is removed from the console)
28) (2pts) Show screenshot of the output of "show run" no n1 and n2 highlighting the costs you assigned to the links

a.



```
n1(config-if)# ip ospf cost 20
n1(config-if)# exit
n1(config)# exit
n1# show run
Building configuration...

Current configuration:
!
!
service integrated-vtysh-config
!
interface eth0
 ip address 10.0.0.1/24
 ip ospf cost 10
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
interface eth1
 ip address 10.0.1.1/24
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
interface eth2
 ip address 10.0.3.2/24
 ip ospf cost 20
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
router ospf
 ospf router-id 10.0.0.1
 network 10.0.0.1/24 area 0.0.0.0
--More--
```

```
n2(config)# interface eth2
n2(config-if)# ip ospf cost 10
n2(config-if)# exit
n2(config)# exit
n2# show run
Building configuration...

Current configuration:
!
!
service integrated-vtysh-config
!
interface eth0
 ip address 10.0.0.2/24
 ip ospf cost 20
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
interface eth1
 ip address 10.0.2.1/24
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
interface eth2
 ip address 10.0.3.1/24
 ip ospf cost 10
 ip ospf dead-interval 6
 ip ospf hello-interval 2
 ip ospf network point-to-point
!
router ospf
 ospf router-id 10.0.0.2
 network 10.0.0.2/24 area 0.0.0.0
 network 10.0.2.1/24 area 0.0.0.0
 network 10.0.3.1/24 area 0.0.0.0
!
ip forwarding
ipv6 forwarding
--More--
```

29) Do a ping between the two hosts n3 and n4

30) (0.5pts) Which of the two links connecting n1 and n2 is the ping request traffic going over? Show a screenshot to support your answer

a.

```
root@n3:/tmp/pycore.2/n3.conf# ping 10.0.2.20 -R
PING 10.0.2.20 (10.0.2.20) 56(124) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=0.053 ms
RR:     10.0.1.20
        10.0.0.1
        10.0.2.1
        10.0.2.20
        10.0.2.20
        10.0.3.1
        10.0.1.1
        10.0.1.20

64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=0.069 ms        (same route)
64 bytes from 10.0.2.20: icmp_seq=3 ttl=62 time=0.072 ms        (same route)
64 bytes from 10.0.2.20: icmp_seq=4 ttl=62 time=0.073 ms        (same route)
```

b. It's using 10.0.2.1(upper line) as request going over

31) (0.5pts) Which of the two links connecting n2 and n1 is the ping reply traffic going over? Show a screenshot to support your answer

a.

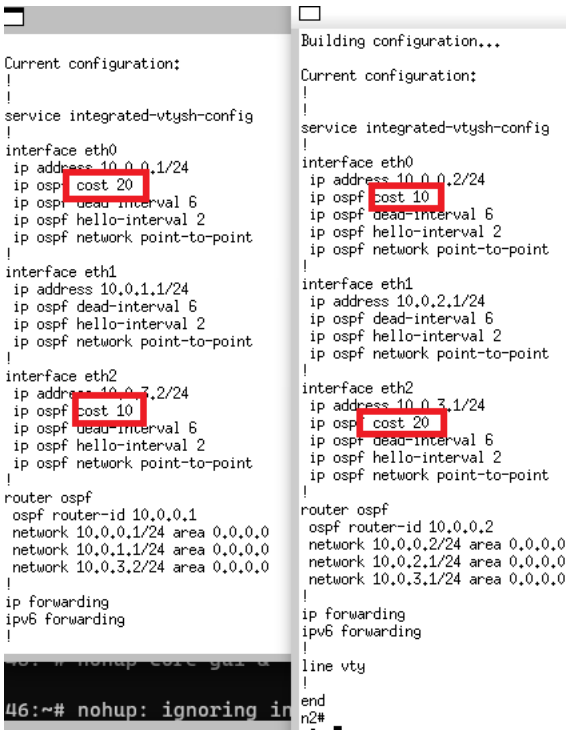
```
root@n3:/tmp/pycore.2/n3.conf# ping 10.0.2.20 -R
PING 10.0.2.20 (10.0.2.20) 56(124) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=0.053 ms
RR:     10.0.1.20
        10.0.0.1
        10.0.2.1
        10.0.2.20
        10.0.2.20
        10.0.3.1
        10.0.1.1
        10.0.1.20

64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=0.069 ms        (same route)
64 bytes from 10.0.2.20: icmp_seq=3 ttl=62 time=0.072 ms        (same route)
64 bytes from 10.0.2.20: icmp_seq=4 ttl=62 time=0.073 ms        (same route)
```

b. It's using 10.0.3.1(bottom line) as reply traffic going over.

32) (0.5pts) Can you justify the routes the OSPF picked?

a. It picked all routes with lowest cost in the OSPF table.

33) Repeat ALL of part 3 but set the cost of top link to 20 and bottom link to 10 on node n1, and cost of top link to 10 and bottom link to 20 on node n2

a. 

b. Request is using 10.0.3.2 using the bottom line

```
root@n3:/tmp/pycore.2/n3.conf# ping 10.0.2.20 -R
PING 10.0.2.20 (10.0.2.20) 56(124) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=0.086 ms
RR:     10.0.1.20
        10.0.3.2
        10.0.2.1
        10.0.2.20
        10.0.2.20
        10.0.0.2
        10.0.1.1
        10.0.1.20

64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=0.075 ms        (same route)
^C
```

i.

c. Reply is replying to 10.0.1.1 using the upper line.

```
root@n3:/tmp/pycore.2/n3.conf# ping 10.0.2.20 -R
PING 10.0.2.20 (10.0.2.20) 56(124) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=0.086 ms
RR:     10.0.1.20
        10.0.3.2
        10.0.2.1
        10.0.2.20
        10.0.2.20
        10.0.0.2
        10.0.1.1
        10.0.1.20

64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=0.075 ms        (same route)
^C
```

i.

d. As the cost swapped, the line choice is also swapped

34) Do you now see how routes can be influenced through costs?
   a. Yes, the ospf chooses the line with lower cost.

Part 4: OSPF Traffic Analysis (3 points)

35) Now do a Wireshark capture to observe OSPF traffic from on the top link on n1
36) (1.5pts) Observe the Hello messages being sent. How often do you see them? Do they correlate to the value in the OSPF Hello Packet “Hello Interval” field in the header? What do they corelate to in the config you showed in (Part1.5, 1.11)

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	224.0.0.5	OSPF	82	Hello Packet
2	0.000157406	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
3	2.000650998	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
4	2.000886240	10.0.0.1	224.0.0.5	OSPF	82	Hello Packet
5	4.001005075	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
6	4.001168117	10.0.0.1	224.0.0.5	OSPF	82	Hello Packet
7	6.001893770	10.0.0.1	224.0.0.5	OSPF	82	Hello Packet
8	6.002105319	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
9	8.002773680	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
10	8.002938980	10.0.0.1	224.0.0.5	OSPF	82	Hello Packet
11	10.003686821	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
12	10.003837360	10.0.0.1	224.0.0.5	OSPF	82	Hello Packet
13	12.004337302	10.0.0.1	224.0.0.5	OSPF	82	Hello Packet
14	12.004530486	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
15	14.004698931	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
16	14.004890724	10.0.0.1	224.0.0.5	OSPF	82	Hello Packet

a.

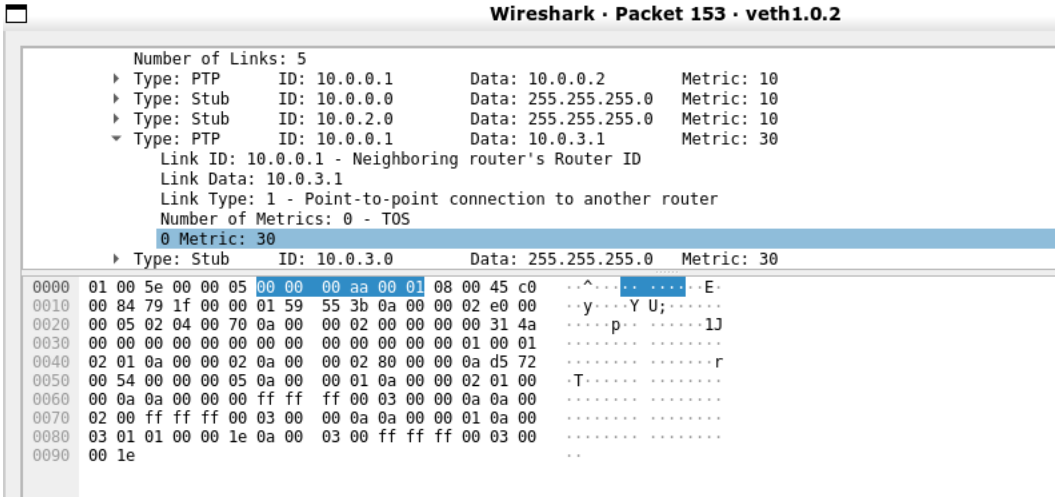
b. Every 2 second.

c. Yes the configuration is also 2 second.

37) Now change the cost of the bottom link on node n2 to anything different

38) (0.5pts) Do you see OSPF traffic flooded on the top link of node n1 as a result? Show the field within the OSPF LSA capture that includes the new cost value.

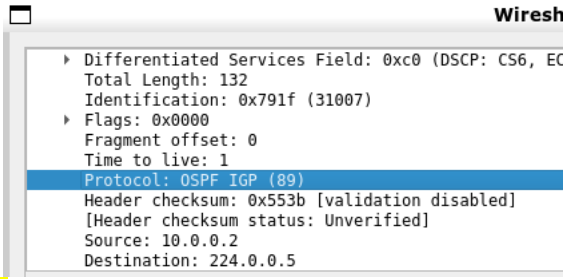
a. Yes



b.

c. The new value shows in the LS update LSA as Type PTP.

39) (0.5pts) What protocol number is OSPF (you should see it in the IP header)? Highlight the part of the IP header with the protocol number



a.

40) (0.5pts) What is the destination address of the LSA? That address is a multicast address which is somewhere between unicast and broadcast: a router will forward it ONLY if an host registered to receive it and the router is on the shortest path to that host.

a. 224.0.0.5

Part 5: SDN Configuration (4 points)

~~41) Disable OSPF and Zebra on both nodes n1 and n2 as shown above~~

42) Install the following software in Ubuntu: openvswitch and bridge-utils
    a. sudo apt-get install bridge-utils openvswitch-switch

43) The links between nodes n1 and n2 are connected by bridges. No show the bridges connecting the interfaces run the command: brctl show. You should see something like the screenshot below

```
root@core-VM:/home/core# brctl show
bridge name     bridge id               STP enabled     interfaces
b.5.1           8000.62f77ea54f53       no              veth1.0.1
                                                        veth2.0.1
b.6.1           8000.667953e80aa3       no              veth1.1.1
                                                        veth3.0.1
b.7.1           8000.82bd816089a5       no              veth2.1.1
                                                        veth4.0.1
b.8.1           8000.0292feb23a84       no              veth1.2.1
                                                        veth2.2.1
```

44) Obviously, the bridges between n1 and n2 are the ones that will have both veth1.X and veth2.X in them. In my scenario that would be b.5.1 and b.8.1. You should note the names of bridges and interfaces associated with them for your case and use them in place of mine.

45) (0.5pts) Show the output of "brctl show"
    a.

```
root@ip-172-31-41-246:~# brctl show
bridge name     bridge id               STP enabled     interfaces
b.5.1           8000.9e8011c4e6bb       no              veth1.0.1
                                                        veth2.0.1
b.6.1           8000.3648cb0e6c19       no              veth1.1.1
                                                        veth3.0.1
b.7.1           8000.2ecf1531ed0d       no              veth2.1.1
                                                        veth4.0.1
b.8.1           8000.4253f6806f23       no              veth1.2.1
                                                        veth2.2.1
```

46) We will remove these bridges and replacing with a software programmable switch, which is what is used in an SDN architecture
    a. Remove interfaces from bridge b.5.1
        i. brctl delif b.5.1 veth1.0.1
        ii. brctl delif b.5.1 veth2.0.1
    b. Remove interfaces from bridge b.8.1
        i. brctl delif b.8.1 veth1.2.1
        ii. brctl delif b.8.1 veth2.2.1
    c. Bring both bridge interfaces down
        i. ifconfig b.5.1 down
        ii. ifconfig b.8.1 down
    d. Delete both bridges
        i. brctl delbr b.5.1
        ii. brctl delbr b.8.1

47) (0.5pts) Show the output of “brctl show”

a.

```
root@ip-172-31-41-246:~# brctl show
bridge name     bridge id               STP enabled     interfaces
b.6.1           8000.3648cb0e6c19       no              veth1.1.1
                                                        veth3.0.1
b.7.1           8000.2ecf1531ed0d       no              veth2.1.1
                                                        veth4.0.1
```

48) If you try to ping n4 from n3 you should not be able to. The links are disconnected

49) Now we add the virtual switch and attach the interfaces to it

50) To add a virtual switch:

a. ovs-vsctl add-br br0

51) To add interfaces to the switch

a. ovs-vsctl add-port br0 veth1.0.1
b. ovs-vsctl add-port br0 veth2.0.1
c. ovs-vsctl add-port br0 veth1.2.1
d. ovs-vsctl add-port br0 veth2.1.1

52) (0.5pts) Show the output of: ovs-vsctl dump-ports br0

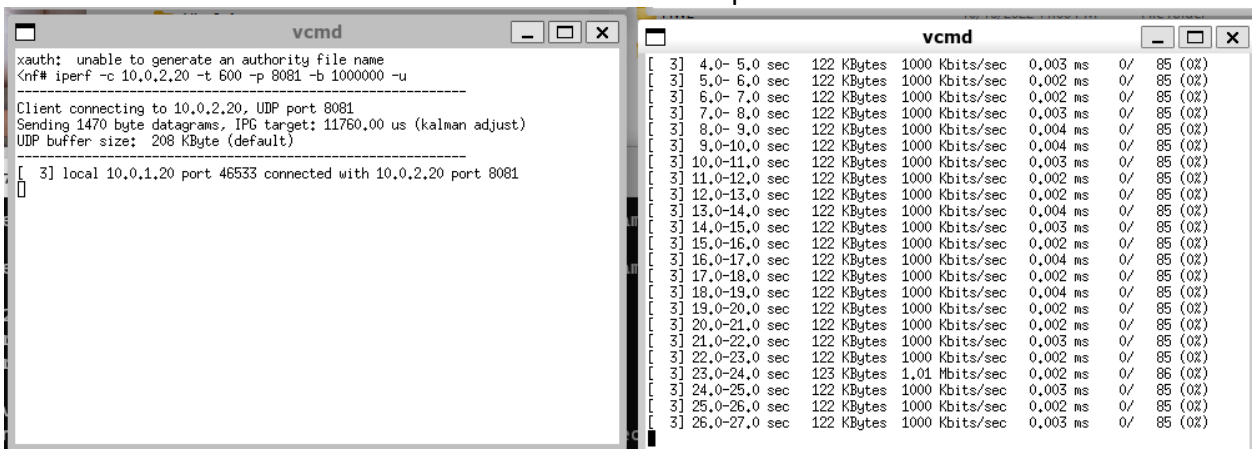
a.

```
root@ip-172-31-41-246:~# ovs-ofctl dump-ports br0
OFPST_PORT reply (xid=0x2): 5 ports
  port  "veth2.2.1": rx pkts=156, bytes=14312, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=161, bytes=15838, drop=0, errs=0, coll=0
  port LOCAL: rx pkts=0, bytes=0, drop=116, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "veth1.0.1": rx pkts=159, bytes=14910, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=173, bytes=16950, drop=0, errs=0, coll=0
  port  "veth2.0.1": rx pkts=165, bytes=15862, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=175, bytes=17574, drop=0, errs=0, coll=0
  port  "veth1.2.1": rx pkts=166, bytes=16092, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=183, bytes=18138, drop=0, errs=0, coll=0
```

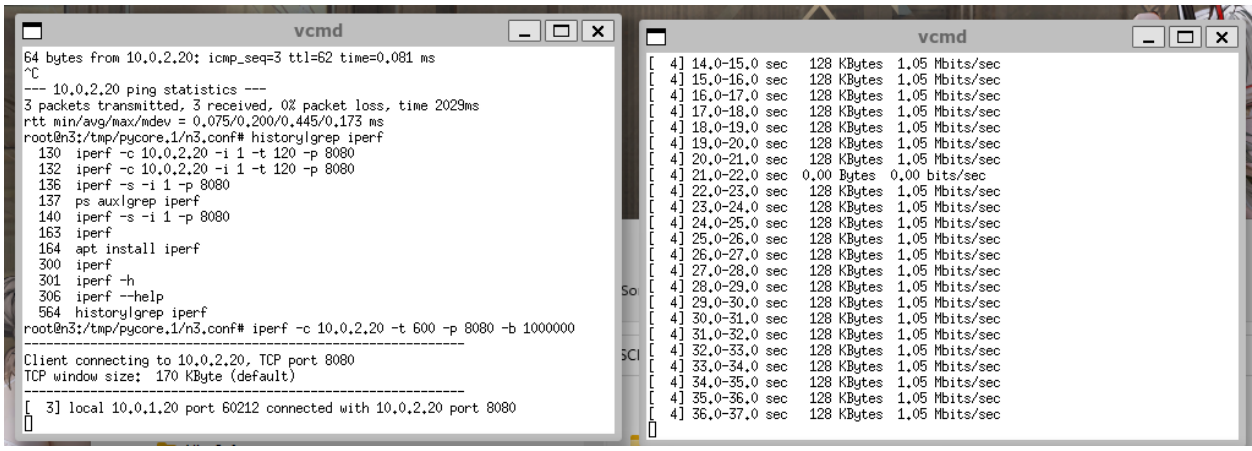
53) With that you should now be able to ping between nodes n1 and n2.

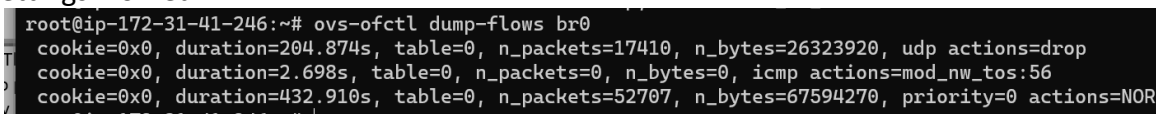
54) Start a UDP flow from n3 to n4 for 10 minutes with rate of 1Mbps

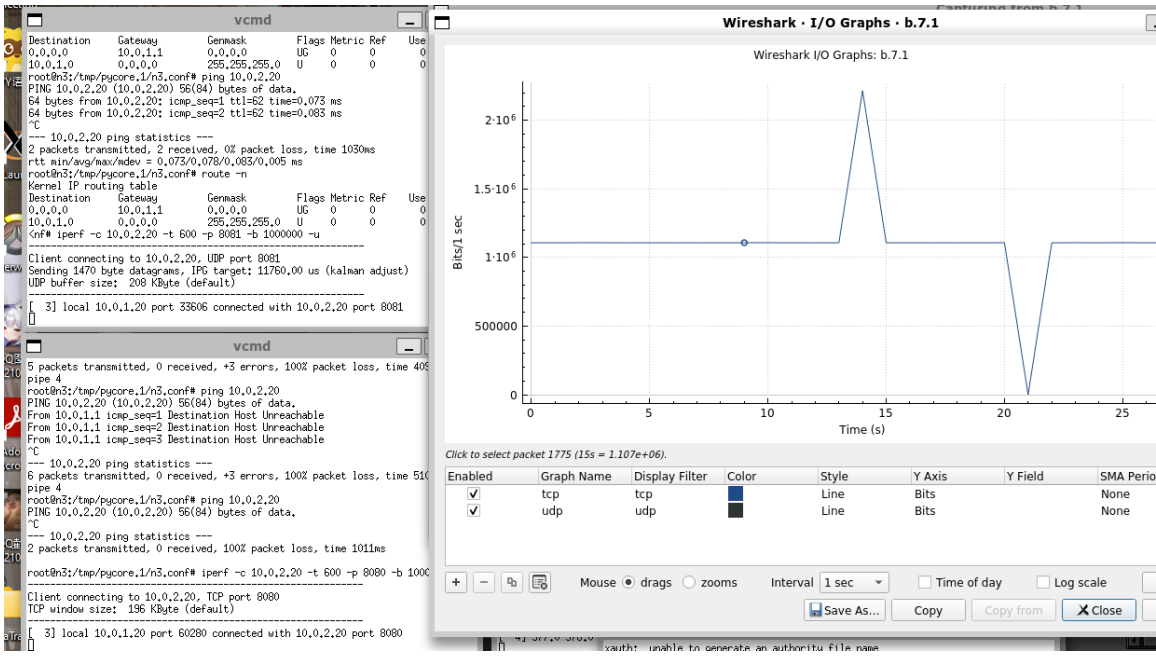
a.



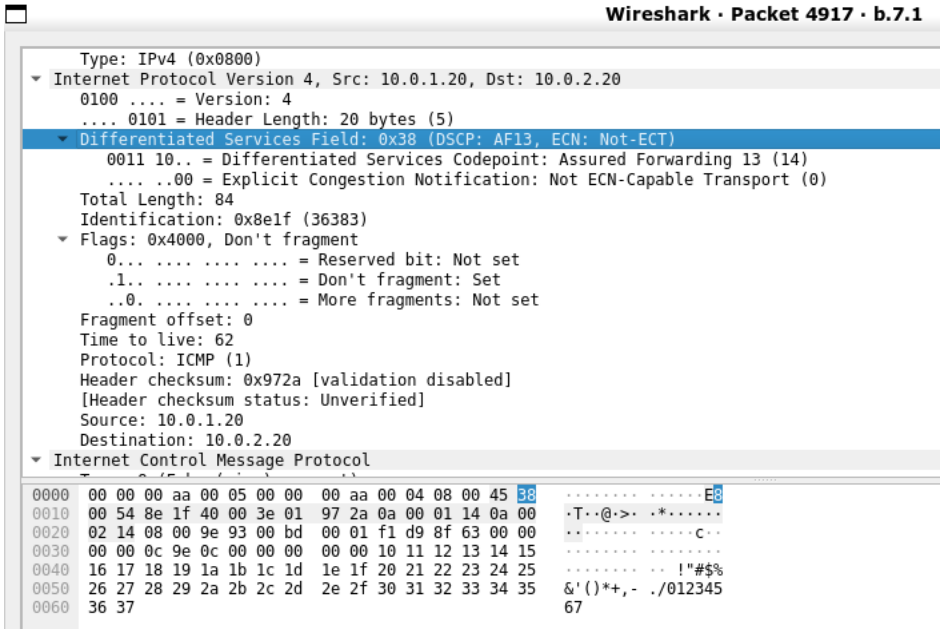
55) Start a TCPP flow from n3 to n4 for 10 minutes with rate of 1Mbps

a. 

56) We will now learn about programming the switch. The switch can be managed by a controller which uses a protocol called OpenFlow to program the switch. Since we do not have a controller, we will program the switch by hand
57) We will use a tool called ovs-ofctl. Learn about it here:
[Open vSwitch Documentation Contents — Open vSwitch 3.0.90 documentation]
[ovs-actions — Open vSwitch 3.0.90 documentation]
58) A rule link this: ovs-ofctl add-flow br0 “in_port=veth1.0.1,ip,icmp actions=drop” will drop all ICMP packets coming in from the veth1.0.1
59) (1.5pts) Add the following rules to the bridge:
    a. Drop all UDP packets between n3 to n4 (look at actions drop)
        i. ovs-ofctl add-flow br0 "ip,udp actions=drop"
    b. Allow all TCP packets between n3 to n4
        i. ovs-ofctl add-flow br0 "ip,tcp actions=normal"
    c. Add a ToS value of 56 to ICMP packets from n3 to n4 (look at mod_tos action)
        i. ovs-ofctl add-flow br0 "icmp,actions=mod_nw_tos:56,normal
    d. "Use screenshot from ovs-ofctl dump-flows br0 AND wireshark to show me that these settings worked
        i. 

ii. 

iii. Only TCP has traffic on n2 to n4 after setting drop action.

iv. 

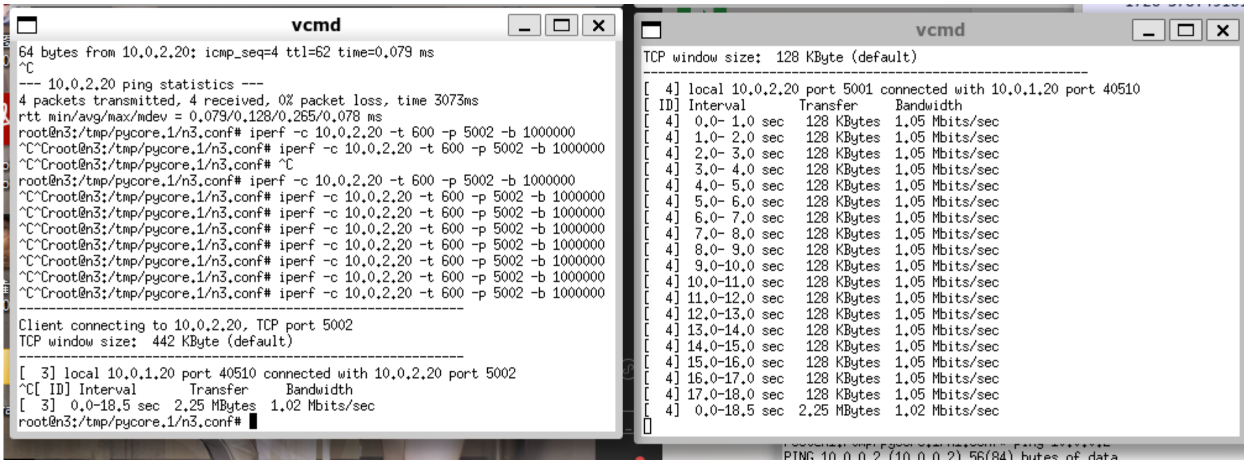
v. You can see icmp tos is 0x38->56

60) Stop both TCP and UDP flows clients

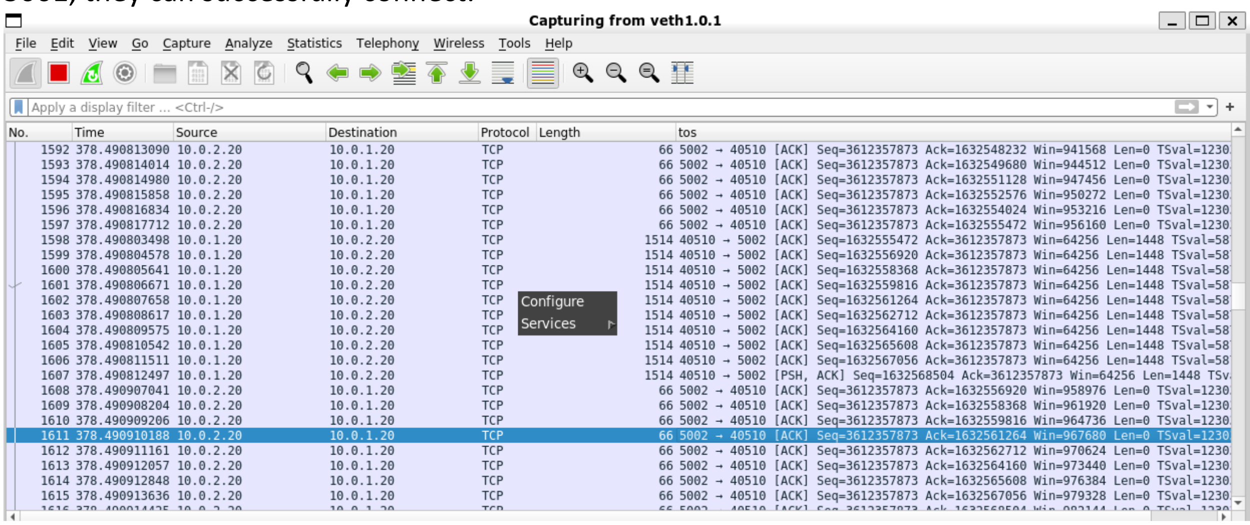
61) Change the destination port configured in the iperf TCP client to 5002

62) (1.5pts) Now add ovs-ofctl rules to change the destination port of the packets to 5001 as they leave n1, and change them back to 5002 as they leave n2. Look at actions mod_tp_src and mod_tp_dst.

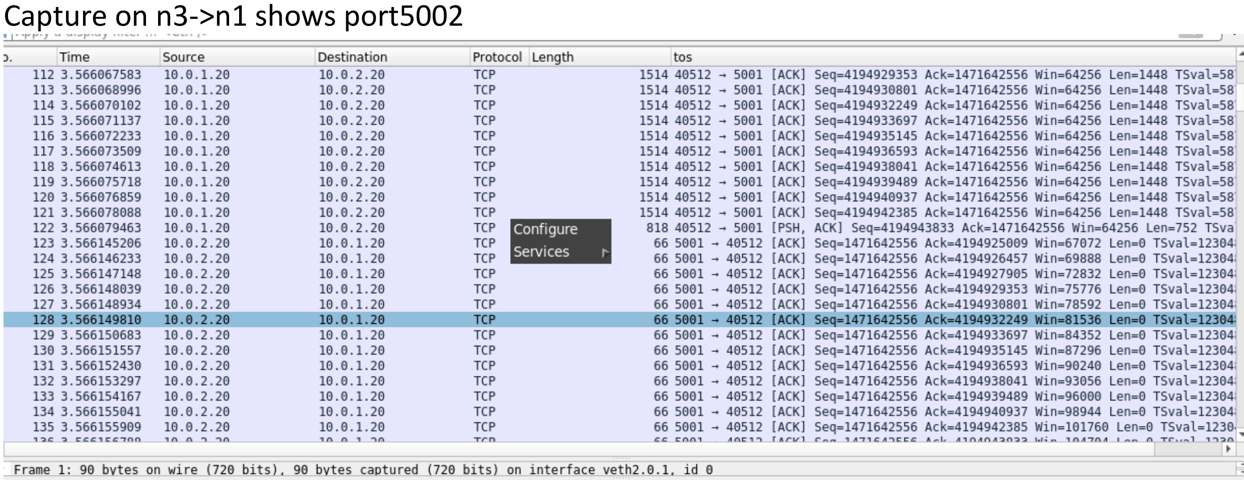
    a. Use screenshot from ovs-ofctl dump-flows br0 AND wireshark to show me that these settings worked

b. 

c. You can see the n3 is trying to connecting to port 5002 while n4 is hosting at port 5001, they can successfully connect.

d. 

e. Capture on n3->n1 shows port5002

f. 

g. Capture on n2->n4 shows port 5001.

h.

```
root@ip-172-31-41-246:~# ovs-ofctl dump-flows br0
 cookie=0x0, duration=3281.401s, table=0, n_packets=5236, n_bytes=7817024, priority=65535,tcp,in_port="veth1.0.1",t
p_dst=5002 actions=mod_tp_dst:5001,NORMAL
 cookie=0x0, duration=3262.345s, table=0, n_packets=3, n_bytes=222, priority=65535,tcp,in_port="veth2.0.1",tp_dst=5
001 actions=mod_tp_dst:5002,NORMAL
 cookie=0x0, duration=3548.695s, table=0, n_packets=68, n_bytes=4132, tcp actions=NORMAL
 cookie=0x0, duration=2497.012s, table=0, n_packets=16, n_bytes=1568, icmp actions=mod_nw_tos:56,NORMAL
 cookie=0x0, duration=226.452s, table=0, n_packets=2221, n_bytes=146602, priority=65534,tcp actions=mod_tp_src:5002
,NORMAL
 cookie=0x0, duration=3886.455s, table=0, n_packets=9877, n_bytes=1712690, actions=NORMAL
```

i. You can see both flow rules has packets coming through.