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).

Graphical user interface

Description automatically generated

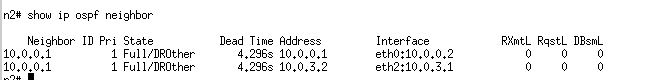
## Part 1: Observing Adjacencies (2points)

1. Start the CORE scenario.
   1. Chart

      Description automatically generated
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)
   1. Text

      Description automatically generated
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?
   1. 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
   1. Text

      Description automatically generated
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.
   1. A picture containing text

      Description automatically generated
   2. 
10. (0.5pts) What is an adjacency? Where does the neighbor ID in the adjacency come from?
    1. 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)
    1. 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)

1. 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)
2. (0.5pts) Show a screenshot of the output from 12 highlighting the link costs
   1. Graphical user interface, text, application

      Description automatically generated
3. Run the command “show ip ospf route” to show the routing table
   1. Text

      Description automatically generated
4. Do a ping between the two hosts n3 and n4
   1. Text

      Description automatically generated
5. (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
   1. It’s using the upper line(10.0.0.1)
   2. Text

      Description automatically generated
6. (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
   1. It’s also the upper line(10.0.2.1)
   2. Text

      Description automatically generated
7. (0.5pts) Can you justify the routes that OSPF picked?
   1. It could because the cost of two link is equal. So it picked the one in higher alphabetical order.

## Part 3: Modifying Costs (7points)

1. Do the below on node n1
2. 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)”
3. Let’s assume the top link connecting n1 and n2 is called eth100 (get actual interface name from 8)
4. Run the command “interface eth100” to edit config of the top link
5. Run “ip ospf cost 10” to set the interface cost to 10
6. Run “exit” to exit configure mode of the link
7. Repeat 22-24 so you have a setting where top link cost is 10 and bottom link cost is 20 on node n1
8. 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
9. 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)
10. (2pts) Show screenshot of the output of “show run” no n1 and n2 highlighting the costs you assigned to the links
    1. Text

       Description automatically generated
11. Do a ping between the two hosts n3 and n4
12. (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
    1. Text, letter

       Description automatically generated
    2. It’s using 10.0.2.1(upper line) as request going over
13. (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
    1. Text, letter

       Description automatically generated
    2. It’s using 10.0.3.1(bottom line) as reply traffic going over.
14. (0.5pts) Can you justify the routes the OSPF picked?
    1. It picked all routes with lowest cost in the OSPF table.
15. 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
    1. Text

       Description automatically generated
    2. Request is using 10.0.3.2 using the bottom line
       1. Text, letter

          Description automatically generated
    3. Reply is replying to 10.0.1.1 using the upper line.
       1. Text, letter

          Description automatically generated
    4. As the cost swapped, the line choice is also swapped
16. Do you now see how routes can be influenced through costs?
    1. Yes, the ospf chooses the line with lower cost.

## Part 4: OSPF Traffic Analysis (3 points)

1. Now do a Wireshark capture to observe OSPF traffic from on the top link on n1
2. (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)
   1. Table

      Description automatically generated
   2. Every 2 second.
   3. Yes the configuration is also 2 second.
3. Now change the cost of the bottom link on node n2 to anything different
4. (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.
   1. Yes
   2. Table

      Description automatically generated
   3. The new value shows in the LS update LSA as Type PTP.
5. (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
   1. Text

      Description automatically generated
6. (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.
   1. 224.0.0.5

## Part 5: SDN Configuration (4 points)

1. Disable OSPF and Zebra on both nodes n1 and n2 as shown above
2. Install the following software in Ubuntu: openvswitch and bridge-utils
   1. sudo apt-get install bridge-utils openvswitch-switch
3. 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



1. 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.
2. (0.5pts) Show the output of “brctl show”
   1. Text

      Description automatically generated with medium confidence
3. We will remove these bridges and replacing with a software programmable switch, which is what is used in an SDN architecture
   1. Remove interfaces from bridge b.5.1
      1. brctl delif b.5.1 veth1.0.1
      2. brctl delif b.5.1 veth2.0.1
   2. Remove interfaces from bridge b.8.1
      1. brctl delif b.8.1 veth1.2.1
      2. brctl delif b.8.1 veth2.2.1
   3. Bring both bridge interfaces down
      1. ifconfig b.5.1 down
      2. ifconfig b.8.1 down
   4. Delete both bridges
      1. brctl delbr b.5.1
      2. brctl delbr b.8.1
4. (0.5pts) Show the output of “brctl show”
   1. Text

      Description automatically generated
5. If you try to ping n4 from n3 you should not be able to. The links are disconnected
6. Now we add the virtual switch and attach the interfaces to it
7. To add a virtual switch:
   1. ovs-vsctl add-br br0
8. To add interfaces to the switch
   1. ovs-vsctl add-port br0 veth1.0.1
   2. ovs-vsctl add-port br0 veth2.0.1
   3. ovs-vsctl add-port br0 veth1.2.1
   4. ovs-vsctl add-port br0 veth2.1.1
9. (0.5pts) Show the output of: ovs-vsctl dump-ports br0
   1. Text

      Description automatically generated with medium confidence
   2. No dump-ports in this version. Using list-ports instead.
   3. Text

      Description automatically generated
10. With that you should now be able to ping between nodes n1 and n2.
11. Start a UDP flow from n3 to n4 for 10 minutes with rate of 1Mbps
12. Start a TCPP flow from n3 to n4 for 10 minutes with rate of 1Mbps
13. 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
14. We will use a tool called ovs-ofctl. Learn about it here:   
    [Open vSwitch Documentation Contents — Open vSwitch 3.0.90 documentation](https://docs.openvswitch.org/en/latest/contents/)  
    [ovs-actions — Open vSwitch 3.0.90 documentation](https://docs.openvswitch.org/en/latest/ref/ovs-actions.7/)
15. 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
16. (1.5pts) Add the following rules to the bridge:
    1. Drop all UDP packets between n3 to n4 (look at actions drop)
    2. Allow all TCP packets between n3 to n4
    3. Add a ToS value of 56 to ICMP packets from n3 to n4 (look at mod\_tos action)
    4. Use screenshot from ovs-ofctl dump-flows br0 AND wireshark to show me that these settings worked
17. Stop both TCP and UDP flows clients
18. Change the destination port configured in the iperf TCP client to 5002
19. (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.
    1. Use screenshot from ovs-ofctl dump-flows br0 AND wireshark to show me that these settings worked