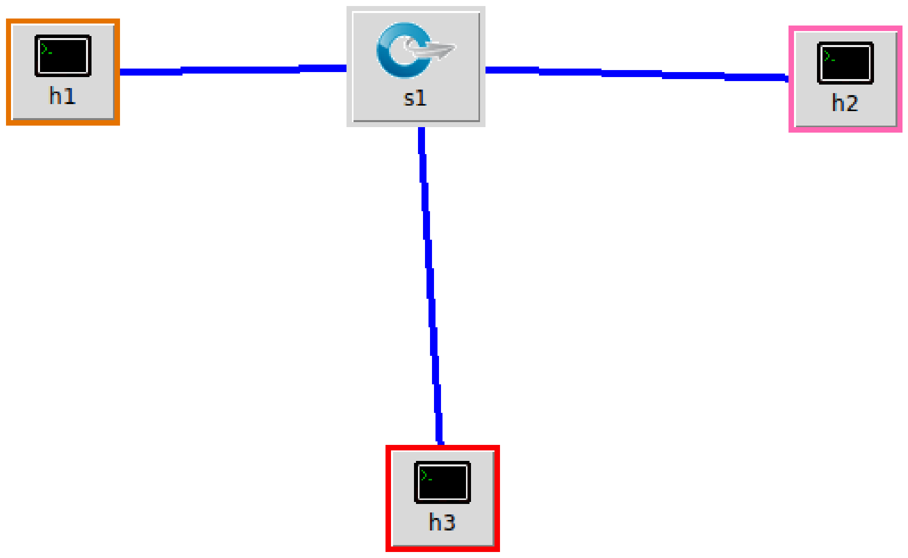
LAB 8: Simple NFV

Network function virtualization (NFV) is a network architecture concept that uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services.

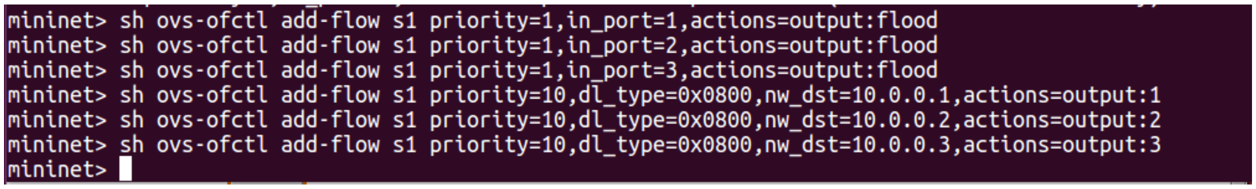
OBJECTIVE:

In this example, H1 will send "Hello, World!" to H2. If the link between S1 and H2 is a lossy (unreliable link). The message during transmission may be lost. If we can add the NFV at the node 3 and route the packet from H1 to H3 first, duplicate the packets and send back to H2, the message has a higher probability of reaching H2.

Step 1: Start a 3 hosts star topology as given below (**Use single line command** instead of writing python code) and **attach the screenshot of the both**. No need to start a controller since we will be manually adding the flows.

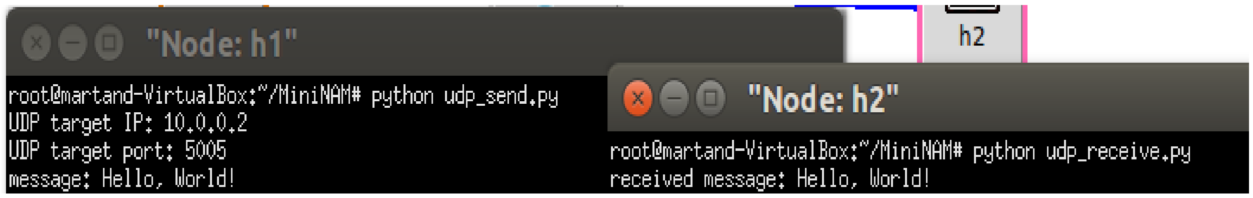


Step 2: Copy the code given in the files (udp_send, udp_receive, and udp_forward) into the correct MiniNAM folder. Now either using xterm or Mininet CLI add flows on the switch.



```
mininet> sh ovs-ofctl add-flow s1 priority=1,in_port=1,actions=output:flood
mininet> sh ovs-ofctl add-flow s1 priority=1,in_port=2,actions=output:flood
mininet> sh ovs-ofctl add-flow s1 priority=1,in_port=3,actions=output:flood
mininet> sh ovs-ofctl add-flow s1 priority=10,dl_type=0x0800,nw_dst=10.0.0.1,actions=output:1
mininet> sh ovs-ofctl add-flow s1 priority=10,dl_type=0x0800,nw_dst=10.0.0.2,actions=output:2
mininet> sh ovs-ofctl add-flow s1 priority=10,dl_type=0x0800,nw_dst=10.0.0.3,actions=output:3
mininet>
```

Step 3: Now open xterm of hosts h1 and h2. First, start udp_receive on the H2 and then start udp_send on the H1 and check if you are receiving packets on H2 or not. (**Paste the screenshot of the H1 and H2 xterm window**)

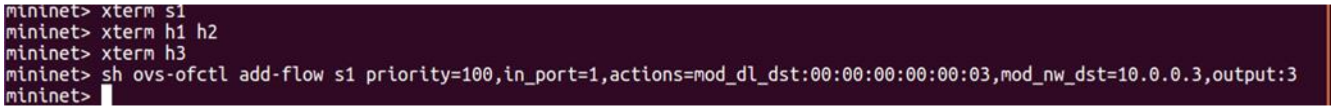


```
root@martand-VirtualBox:~/MiniNAM# python udp_send.py
UDP target IP: 10.0.0.2
UDP target port: 5005
message: Hello, World!
```

```
root@martand-VirtualBox:~/MiniNAM# python udp_receive.py
received message: Hello, World!
```

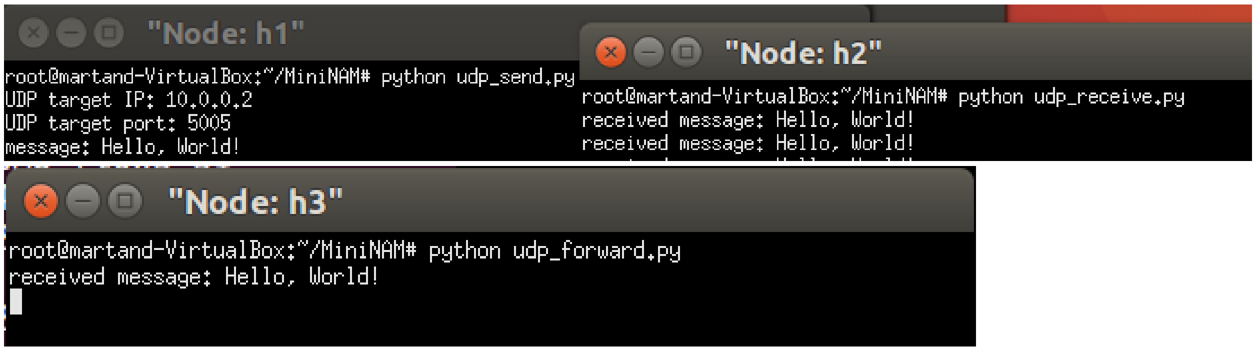
Note you will receive one packet for each time you start udp_send on the H1.

Step 4: Now, we will duplicate the packet on the H3 and send it to H2 just if the path is busy. So, add one more flow on the switch (using mininet CLI or xterm).



```
mininet> xterm s1
mininet> xterm h1 h2
mininet> xterm h3
mininet> sh ovs-ofctl add-flow s1 priority=100,in_port=1,actions=mod_dl_dst:00:00:00:00:00:03,mod_nw_dst=10.0.0.3,output:3
mininet>
```

Now, start the H3's xterm and start udp_forward code on it. Now, again, when you will begin to the udp_send and send packets from H1 to H2, first, it will send the packet to H3. Then H3 will duplicate packet and send both the packets to H2.



```
root@martand-VirtualBox:~/MiniNAM# python udp_send.py
UDP target IP: 10.0.0.2
UDP target port: 5005
message: Hello, World!
```

```
root@martand-VirtualBox:~/MiniNAM# python udp_receive.py
received message: Hello, World!
received message: Hello, World!
```

```
root@martand-VirtualBox:~/MiniNAM# python udp_forward.py
received message: Hello, World!
```

(**Paste the screenshot of the All xterm window**)

TASK:

Now, shut-down the topology and clear all the cache. Start the same three host topology and edit the python code given to you for sending, receiving, and forwarding the packets in such a way that **H2 should send packet**, **H1 should receive and duplicate packets** and **H3 should receive two packets**.
(**Paste the screenshot of all the xterm windows and the flows.)**
Hint: All the initial flows would be the same, you just must change the final flow accordingly, and the IP addresses in the respective python files.

This concludes Lab 8

Thank you!!