Elijah Cordova
CMPE 150
Due 3/19

Final Project Lab Report

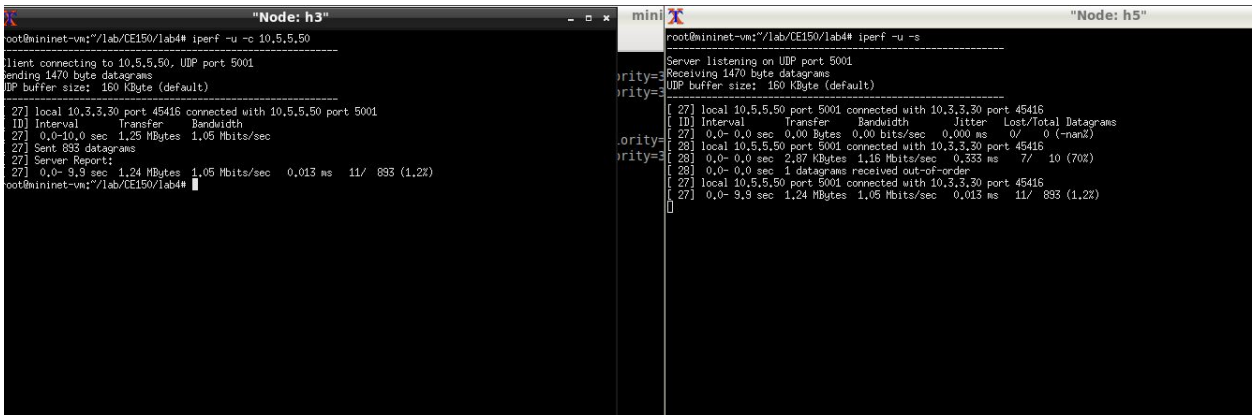
First, let's start with TCP/ipv4. As you can see, h1, h2, h3, and h5 can communicate just fine due to the rules installed in the flow table.

```
mininet@mininet-vm:~/lab/CE150/lab4$ sudo python final_skel.py
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['19.9 Gbits/sec', '19.9 Gbits/sec']
mininet> iperf h5 h3
*** Iperf: testing TCP bandwidth between h5 and h3
*** Results: ['30.9 Gbits/sec', '30.9 Gbits/sec']
mininet> iperf h4 h3
*** Iperf: testing TCP bandwidth between h4 and h3
```

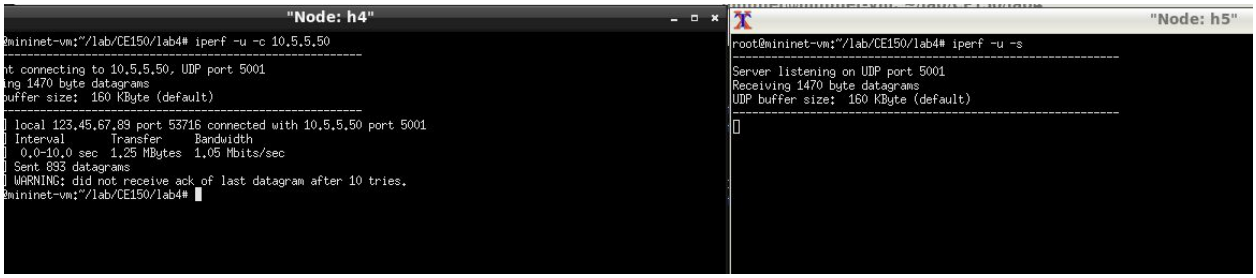
However, what happens when h4, an outsider, tries to connect? The connection completely freezes up, and you have to tell it to give up. Why? I designed the switches 'learn' that h4 is not a part of trusted hosted by ofpmod, as seen by dpctl dump-flows here: (Notice the actions=drop)

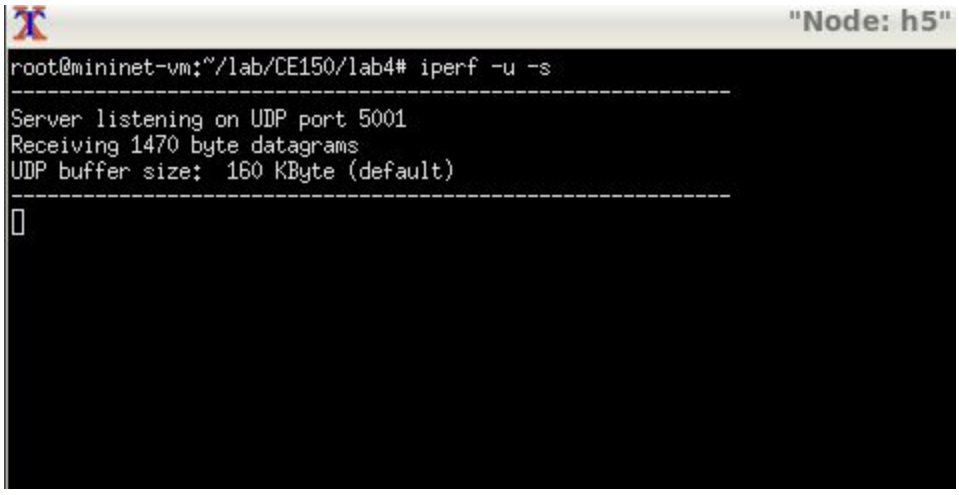
```
mininet> iperf h3 h4
*** Iperf: testing TCP bandwidth between h3 and h4
^C
Interrupt
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
*** s3 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=103.615s, table=0, n_packets=0, n_bytes=0, idle_age=103, priority=3,tcp,nw_src=123.45.67.89,nw_dst=10.3.3.30 actions=drop
 cookie=0x0, duration=103.614s, table=0, n_packets=6, n_bytes=444, idle_age=1, priority=3,tcp,nw_src=10.3.3.30,nw_dst=123.45.67.89 actions=drop
*** s4 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=103.656s, table=0, n_packets=6, n_bytes=444, idle_age=72, priority=3,tcp,nw_src=123.45.67.89,nw_dst=10.3.3.30 actions=drop
 cookie=0x0, duration=103.619s, table=0, n_packets=0, n_bytes=0, idle_age=103, priority=3,tcp,nw_src=10.3.3.30,nw_dst=123.45.67.89 actions=drop
*** s5 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
```

Similar behavior for UDP, another ipv4 protocol. h3 -> h5 udp connection by xterm is sucessful below.

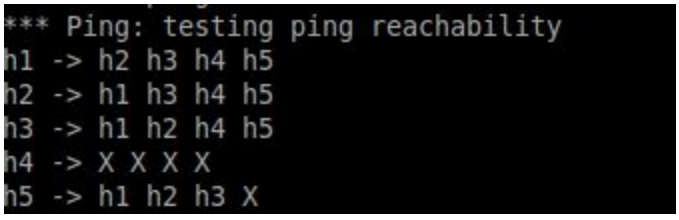


Now let's try UDP on h4 -> h5. H5 Refuses to acknowledge h4 at all, like with TCP!

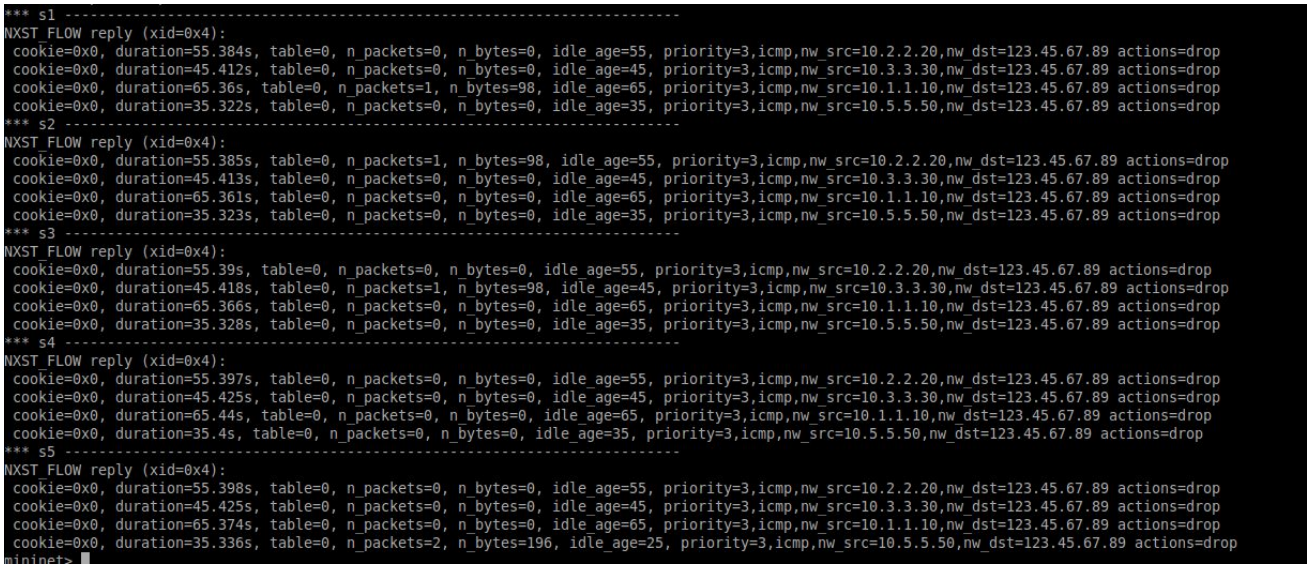



Now the attacker tries to use pingall. What happens?




As you can see, h4 can still attempt to send out ARP packets. I have all non-ipv4 packets to of.OFPP_FLOOD. h1 is still pinging h4! However, moment that h4 even attempts to try ICMP packets through ping, it is blocked out from the network. Check out all the actions= 'drop' rules installed.




Overall, I learned a lot about how packet handling works in openflow. You absolutely have to broadcast rules to the appropriate switches when forwarding ports. If I were to expand upon this assignment, I would try to see how to handle a DDoS attack or Trojan attack, as ARP leaves a hole open in our network.