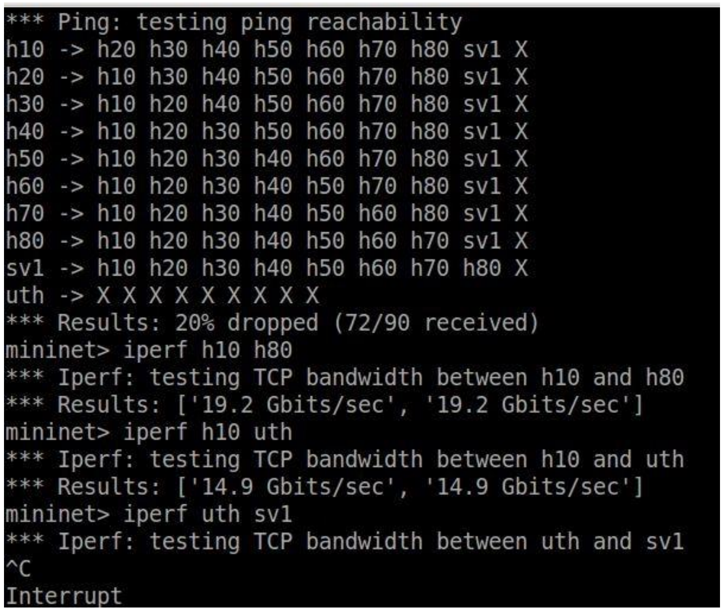
Elad Zohar
ezohar 1745121
CSE 150 Spring 2020

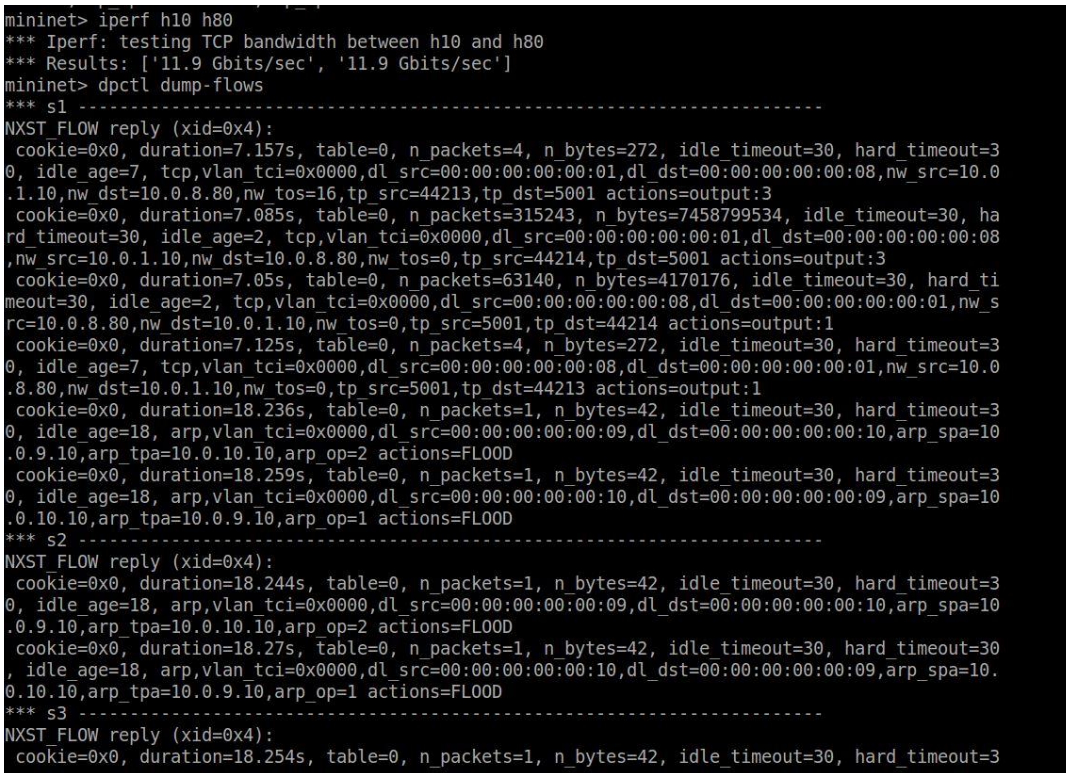
Final Lab

<u>Screen Shots:</u>



```
*** Ping: testing ping reachability
h10 -> h20 h30 h40 h50 h60 h70 h80 sv1 X
h20 -> h10 h30 h40 h50 h60 h70 h80 sv1 X
h30 -> h10 h20 h40 h50 h60 h70 h80 sv1 X
h40 -> h10 h20 h30 h50 h60 h70 h80 sv1 X
h50 -> h10 h20 h30 h40 h60 h70 h80 sv1 X
h60 -> h10 h20 h30 h40 h50 h70 h80 sv1 X
h70 -> h10 h20 h30 h40 h50 h60 h80 sv1 X
h80 -> h10 h20 h30 h40 h50 h60 h70 sv1 X
sv1 -> h10 h20 h30 h40 h50 h60 h70 h80 X
uth -> X X X X X X X X X
*** Results: 20% dropped (72/90 received)
mininet> iperf h10 h80
*** Iperf: testing TCP bandwidth between h10 and h80
*** Results: ['19.2 Gbits/sec', '19.2 Gbits/sec']
mininet> iperf h10 uth
*** Iperf: testing TCP bandwidth between h10 and uth
*** Results: ['14.9 Gbits/sec', '14.9 Gbits/sec']
mininet> iperf uth sv1
*** Iperf: testing TCP bandwidth between uth and sv1
^C
Interrupt
```

pingall and iperf



```
mininet> iperf h10 h80
*** Iperf: testing TCP bandwidth between h10 and h80
*** Results: ['11.9 Gbits/sec', '11.9 Gbits/sec']
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=7.157s, table=0, n_packets=4, n_bytes=272, idle_timeout=30, hard_timeout=3
0, idle_age=7, tcp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:08,nw_src=10.0
.1.10,nw_dst=10.0.8.80,nw_tos=16,tp_src=44213,tp_dst=5001 actions=output:3
 cookie=0x0, duration=7.085s, table=0, n_packets=315243, n_bytes=7458799534, idle_timeout=30, ha
rd_timeout=30, idle_age=2, tcp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:08
,nw_src=10.0.1.10,nw_dst=10.0.8.80,nw_tos=0,tp_src=44214,tp_dst=5001 actions=output:3
 cookie=0x0, duration=7.05s, table=0, n_packets=63140, n_bytes=4170176, idle_timeout=30, hard_ti
meout=30, idle_age=2, tcp,vlan_tci=0x0000,dl_src=00:00:00:00:00:08,dl_dst=00:00:00:00:00:01,nw_s
rc=10.0.8.80,nw_dst=10.0.1.10,nw_tos=0,tp_src=5001,tp_dst=44214 actions=output:1
 cookie=0x0, duration=7.125s, table=0, n_packets=4, n_bytes=272, idle_timeout=30, hard_timeout=3
0, idle_age=7, tcp,vlan_tci=0x0000,dl_src=00:00:00:00:00:08,dl_dst=00:00:00:00:00:01,nw_src=10.0
.8.80,nw_dst=10.0.1.10,nw_tos=0,tp_src=5001,tp_dst=44213 actions=output:1
 cookie=0x0, duration=18.236s, table=0, n_packets=1, n_bytes=42, idle_timeout=30, hard_timeout=3
0, idle_age=18, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:09,dl_dst=00:00:00:00:00:10,arp_spa=10
.0.9.10,arp_tpa=10.0.10.10,arp_op=2 actions=FLOOD
 cookie=0x0, duration=18.259s, table=0, n_packets=1, n_bytes=42, idle_timeout=30, hard_timeout=3
0, idle_age=18, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:10,dl_dst=00:00:00:00:00:09,arp_spa=10
.0.10.10,arp_tpa=10.0.9.10,arp_op=1 actions=FLOOD
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=18.244s, table=0, n_packets=1, n_bytes=42, idle_timeout=30, hard_timeout=3
0, idle_age=18, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:09,dl_dst=00:00:00:00:00:10,arp_spa=10
.0.9.10,arp_tpa=10.0.10.10,arp_op=2 actions=FLOOD
 cookie=0x0, duration=18.27s, table=0, n_packets=1, n_bytes=42, idle_timeout=30, hard_timeout=30
, idle_age=18, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:10,dl_dst=00:00:00:00:00:09,arp_spa=10.
0.10.10,arp_tpa=10.0.9.10,arp_op=1 actions=FLOOD
*** s3 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=18.254s, table=0, n_packets=1, n_bytes=42, idle_timeout=30, hard_timeout=3
```

dpctl dump-flows

Elad Zohar
ezohar 1745121
CSE 150 Spring 2020

Explanations:

- **Pingall** – will return a list of the reachability of all the network hosts, the server and the untrusted host. Inside the network we definitely want network hosts to be able to reach each other and the server. Because of the ports I designed, all hosts are available from one another, and the server is also reachable. That said we want all IP traffic directed at the server and ICMP packets going to the network to be blocked from the untrusted host. This should result with drops related to all untrusted host connection. The "X" characted seen in the *pingall and iperf* screenshot shows the attempted connection to the untrusted host that drops because of the controller's designed purpose.

- **Iperf** – will return the TCP bandwidth. For the secure host to secure host and secure host to untrusted host connectivity, tangible bandwidth is returned (as seen in *pingall and iperf*) because within the network hosts are designed to send all packets freely, and the untrusted host is only blocked from sending and receiving ICMP packets, so TCP packets make it through just fine. When iperf is run between the server and the untrusted host all packets are dropped, causing iperf to get stuck because it has no packets to analyze bandwidth with. That's why in the screenshot above I had to interrupt it.

- **Dpctl dump-flows** – should respond with all the flows that passed in the network. In the case shown in the image above each switch (s1 – s6) has a few cases being executed and displayed. The "actions:" response at the end of each segment shows how the packet passed through, and will vary between "output:x" (x = {0,1,2,3,4,5,6}) in cases where the ports I programmed were used and "output:FLOOD" when the packets sent weren't "ipv4" packets.

- **Controller –** The final_controller.py file is designed to help monitor and keep the final.py topology secure from hosts outside the trusted network. "do_final" contains all of my code.
  - To start I have a series of variable listed to help keep track and clarify which switches and which hosts are being connected with which ports.
  - The idle and hard timeout is set up per the recommended skeleton.

Elad Zohar
ezohar 1745121
CSE 150 Spring 2020

- The packet filtering section begins by isolating whether the packet being transferred is an “ipv4” packet. If it isn’t then the packet is sent through freely with the port set to of.OFPP_FLOOD in the else statement below. Otherwise, the packet is “ipv4” and we proceed to sort whether it will be let through or dropped.
    - **i.** In the case that the **floor switches** are being used we’ll assign the correct port (1, 2, or 3) as defined by the topology for each respective network host.
    - **ii.** In the case that the **Data Center Switch** is being used we need to assign the correct port value (1 or 3). We also make sure that the source of the packet isn’t the untrusted host. If the source is the untrusted host then the port will remain 1, but because that doesn’t match the assigned port of the topology the packet will be dropped as expected.
    - **iii.** In the case that the **Core Switch** is being used then we have to make sure we traffic the packet to the correct port (1, 2, 3, 4, 5, or 6). That said if the packet is “icmp” AND is coming from the untrusted host, we want to drop the packet. For that reason I set the port to 0 before the if statement so that the if the packet is an “icmp” packet and the source is the untrusted host the packet will be dropped. Otherwise the port will be assigned to direct the packet to either the appropriate switch or to the untrusted host.