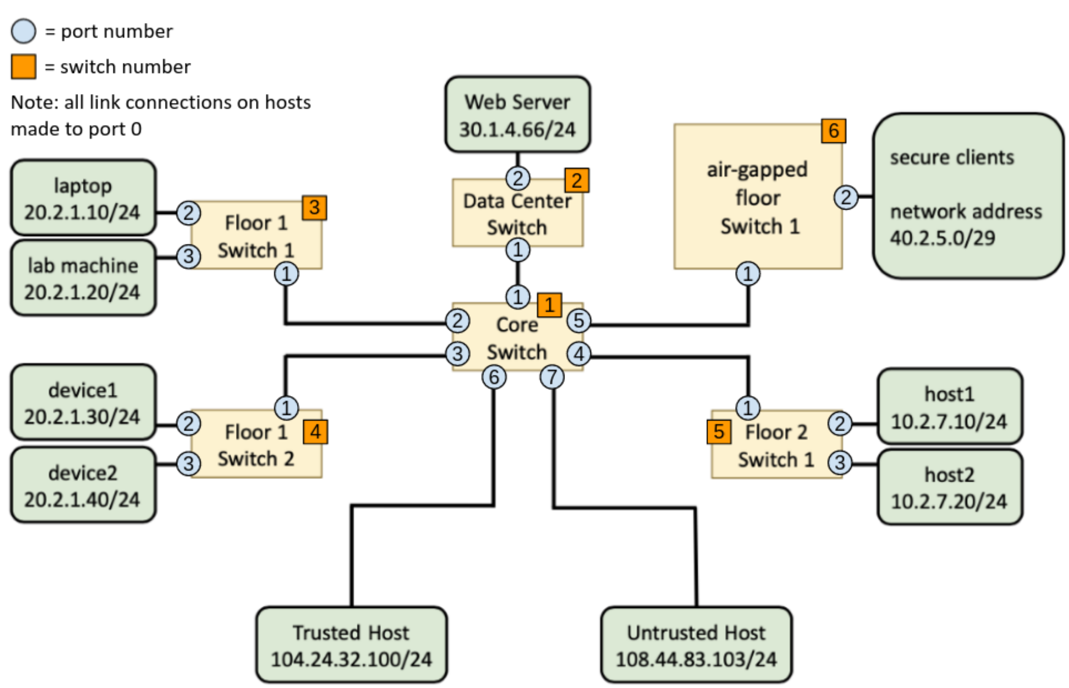
Kyle Won
Professor Parsa
CSE 150
27 November 2021

CSE 150 Final Project Testing/Review

POX Controller Rules

1	Hosts on the Air-Gapped floor can only communicate amongst themselves.
2	Untrusted Host (h_untrust) cannot send ICMP traffic to any devices on Floor 1, Floor 2, or the Web Server (h_server).
3	Untrusted Host (h_untrust) cannot send any IP traffic to the Web Server (h_server).
4	Trusted Host (h_trust) cannot send any ICMP traffic to devices on Floor 1, or the Web Server (h_server).
5	Trusted Host (h_trusted) cannot send any TCP traffic to the Web Server (h_server).
6	Devices on Floor 1 cannot send any ICMP traffic to devices Floor 2, and vice versa.

Network Topology and Links



Testing

1. pingall

Running ‘pingall’ on this network tests all the ICMP traffic rules. The output is as follows:

```
File  Edit  Tabs  Help
mininet@mininet-vm:~$ date
Mon Nov 22 20:51:54 PST 2021
                                                    mininet@mininet-vm: ~/CSE_150_Intr
File  Edit  Tabs  Help
mininet@min...  mininet@min...  mininet@min...  mininet@min...
mininet@mininet-vm:~/CSE_150_Intro_Comp_Networks/final_project/POX-router$ sudo python final_skel.py
Hello network
mininet> pingall
*** Ping: testing ping reachability
client1 -> client2 client3 client4 client5 client6 X X X X X X X X X
client2 -> client1 client3 client4 client5 client6 X X X X X X X X X
client3 -> client1 client2 client4 client5 client6 X X X X X X X X X
client4 -> client1 client2 client3 client5 client6 X X X X X X X X X
client5 -> client1 client2 client3 client4 client6 X X X X X X X X X
client6 -> client1 client2 client3 client4 client5 X X X X X X X X X
device1 -> X X X X X X device2 h_server X X X X lab_mac laptop
device2 -> X X X X X X device1 h_server X X X X lab_mac laptop
h_server -> X X X X X X device1 device2 X X host1 host2 lab_mac laptop
h_trust -> X X X X X X X X X h_untrust host1 host2 X X
h_untrust -> X X X X X X X X X h_trust X X X X
host1 -> X X X X X X X X h_server h_trust X host2 X X
host2 -> X X X X X X X X h_server h_trust X host1 X X
lab_mac -> X X X X X X device1 device2 h_server X X X X laptop
laptop -> X X X X X X device1 device2 h_server X X X X lab_mac
*** Results: 70% dropped (62/210 received)
mininet>
```

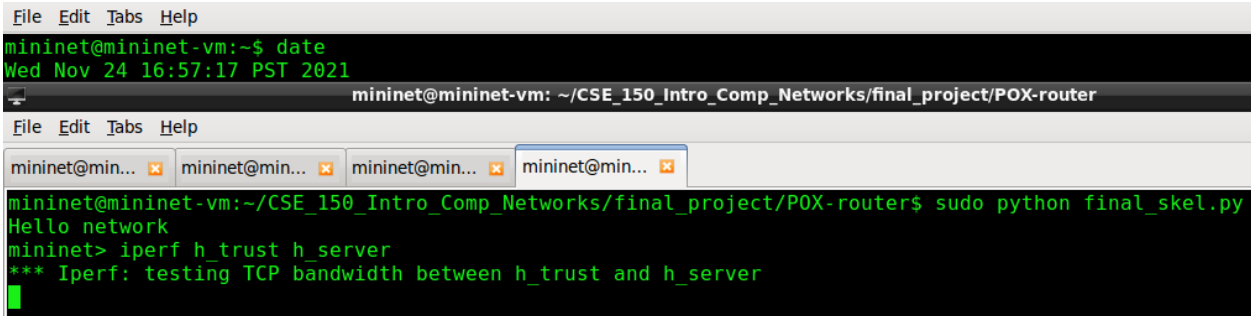
Air-Gapped Clients 1-6 > can only send and receive traffic to and from other Clients on the Air-Gapped floor (rule 1).
Floor 1 > Device1, device2, lab_mac, and laptop cannot reach Untrusted Host (rule 2), Trusted Host (rule 4), or any devices on Floor 2 (rule 6).
Floor 2 > Host1 and host2 cannot reach Untrusted Host (rule 2) or any devices on Floor 1 (rule 6). Unlike the devices on Floor 1, devices on Floor 2 can communicate with Trusted Host (Trusted Host owned by someone on Floor 2).
Web Server > The web server can only communicate via ICMP with devices on Floor 1 and Floor 2. It cannot communicate with Untrusted Host (rule 2) or Trusted Host (rule 4).
Trusted Host > Can only reach Untrusted Host and devices on Floor 2 (rule 4).
Untrusted Host > Can only reach Trusted Host (rule 2).

2. iperf

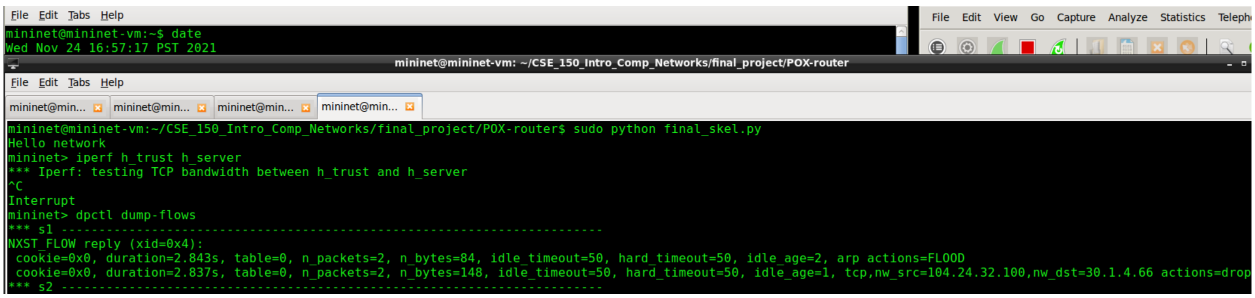
Running ‘iperf deviceX deviceY’ on this network tests the TCP connection between two devices. We can use iperf to observe the effect of the network rules on TCP traffic for the following scenarios:

a) iperf h_trust h_server:

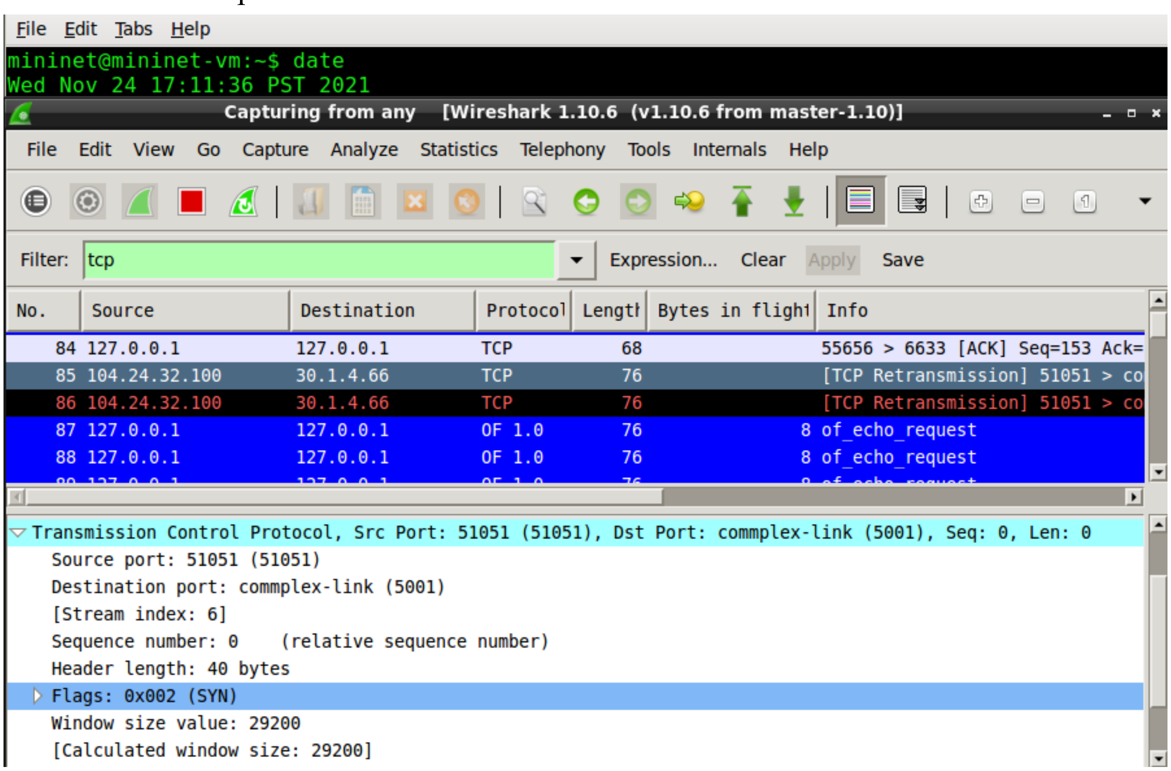
This scenario tests the TCP connection between Trusted Host and Web Server which, by rule 5, should not be allowed. In the mininet console, running ‘iperf h_trust h_server’ results in no output, indicating that the TCP traffic is being dropped.



Dumping the flow table confirms this, as we can see that the Core Switch (s1) which is connected to Trusted Host is using a rule that drops all TCP traffic traveling from Trusted Host's IP (104.24.32.100) to the Web Server's IP (30.1.4.66).

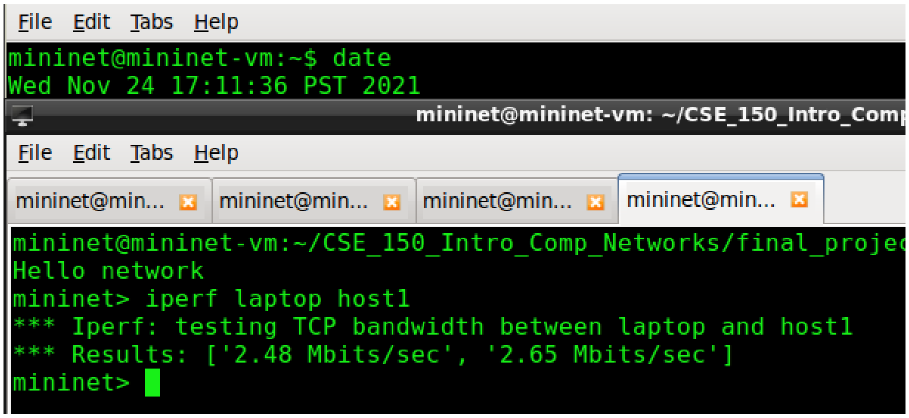


In Wireshark, we can confirm this result as well by filtering by TCP traffic and observing the packets sent from Trusted Host's IP (104.24.32.100) to the Web Server's IP (30.1.4.66). We can see that Trusted Host repeatedly transmits a SYN connection request with no response.

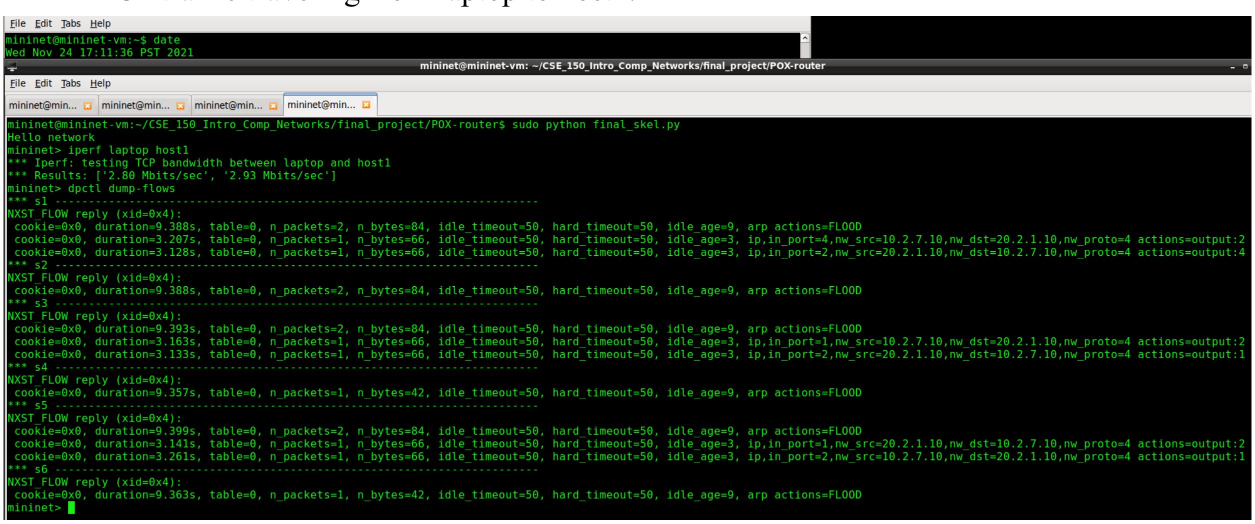


b) iperf laptop host1

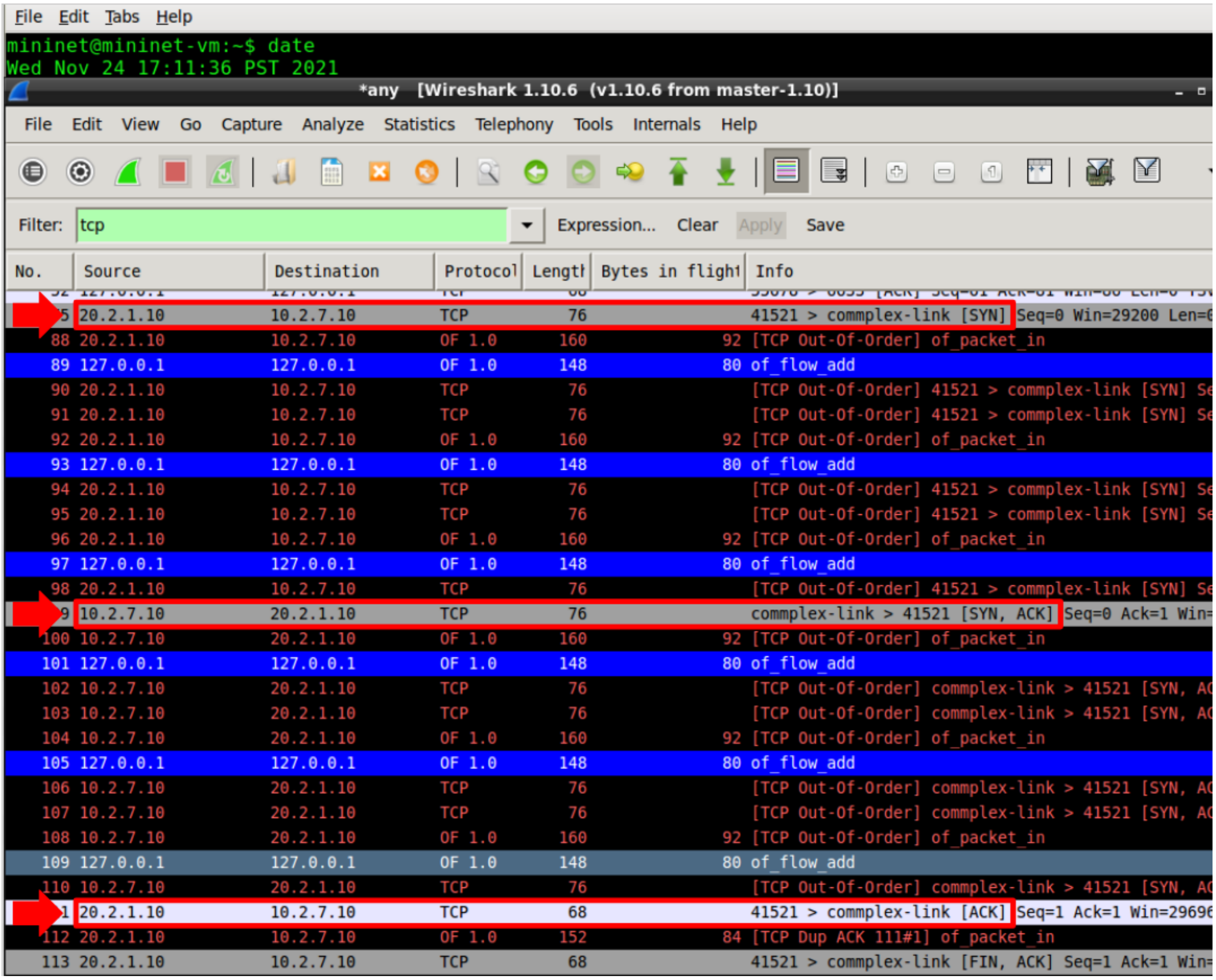
This scenario tests the TCP connection between laptop (on Floor 1) and host1 (on Floor 2). These two devices should be allowed to make a TCP connection by the network rules. In the mininet network console, the command ‘iperf laptop host1’ completes successfully:



Dumping the flow table shows the path that the directed IP traffic took through the network. Looking at the network topology, laptop is connected to Floor 1 Switch 1 which has a switch label of 3. The flow table of s3 shows that packets sourced from laptop (20.2.1.10) and destined for host1 (10.2.7.10) were sent out port 1. Port 1 on Floor 1 Switch 1 links to the Core Switch which has a label of 1. The flow table of s1 shows that packets going from 20.2.1.10 to 10.2.7.10 were sent out port 4. Port 4 on the Core Switch links to Floor 2 Switch 1 which has a switch label of 5. The flow table of s5 shows that packets going from 20.2.1.10 to 10.2.7.10 were sent out port 2. Port 2 on Floor 2 Switch 1 connects to host1. Further analysis of the flow table reveals that there is a path allowing TCP packets to flow in the return direction (host1 to laptop) as well. The Network Topology and Links diagram confirms that the path s3 -> s1 -> s5 is the expected path for TCP traffic traveling from laptop to host1.



In Wireshark, we can confirm that the TCP connection was successful by filtering by TCP packets and observing the traffic between laptop (20.2.1.10) and host1 (10.2.7.10). We can see that laptop transmits a SYN connection request which is followed by a SYN-ACK response from host1 and, finally, an ACK from laptop.



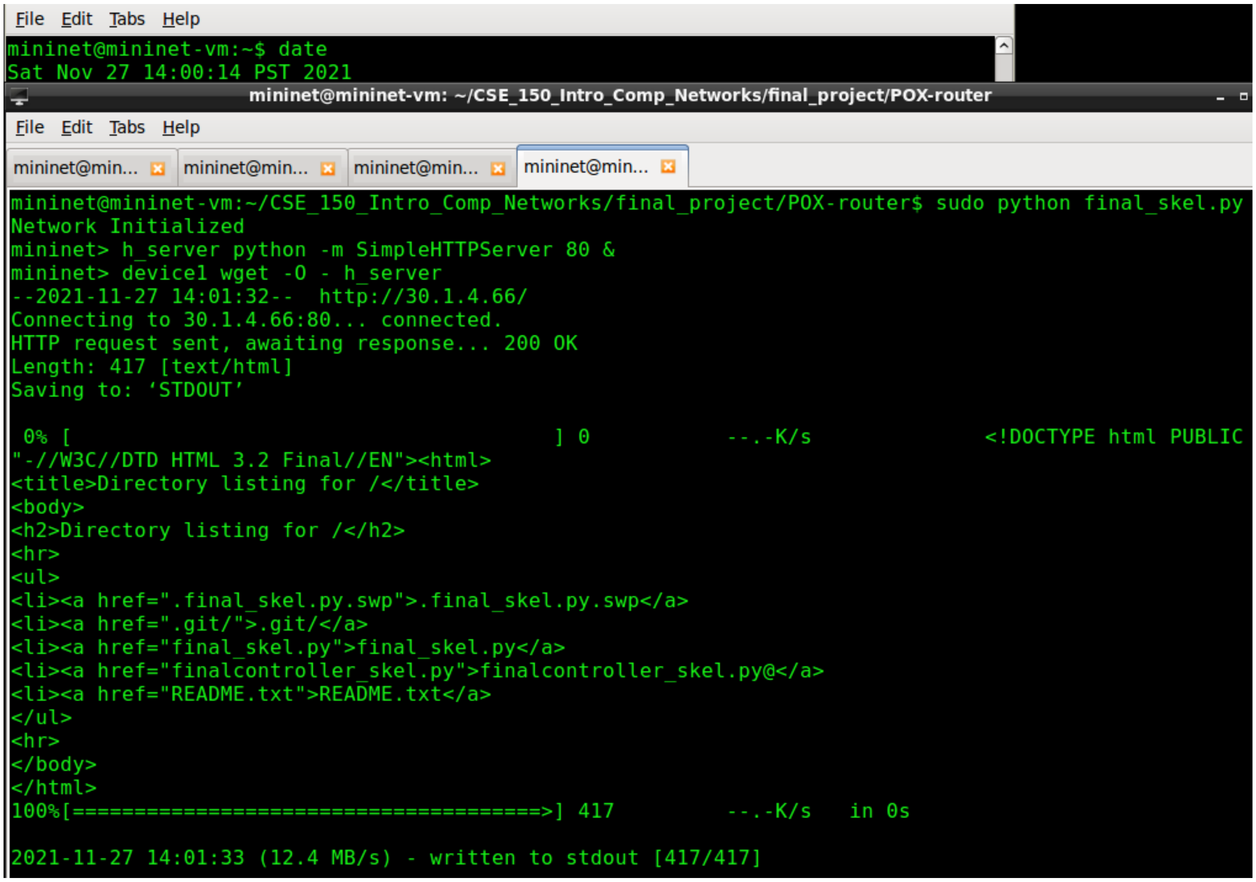
Successful TCP connections should be able to be made between all hosts except Trusted Host or Untrusted Host and Web Server, and any two hosts on different sides of the Air-Gapped Switch.

3. HTTP server

To test the network's routing of HTTP traffic, we can run a simple HTTP server on Web Server using the command 'h_server python -m SimpleHTTPServer 80 &'. We can use 'wget' to observe the network's behavior with HTTP traffic in the following scenarios:

a) device1 wget -O - h_server

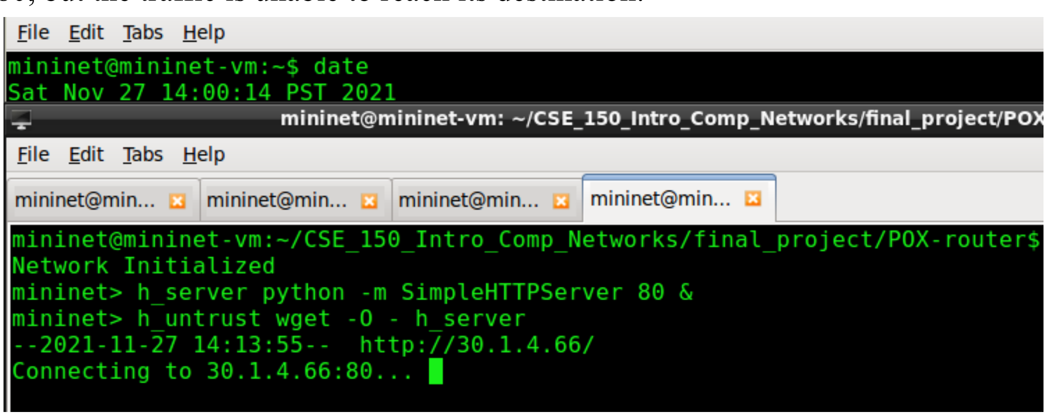
With this command, device1 makes a HTTP request to the Web Server and prints the response to the console. We can see in the output of the Mininet console that device1 makes a successful connection to the web server. This is expected since the network rules allow TCP traffic between these two hosts.



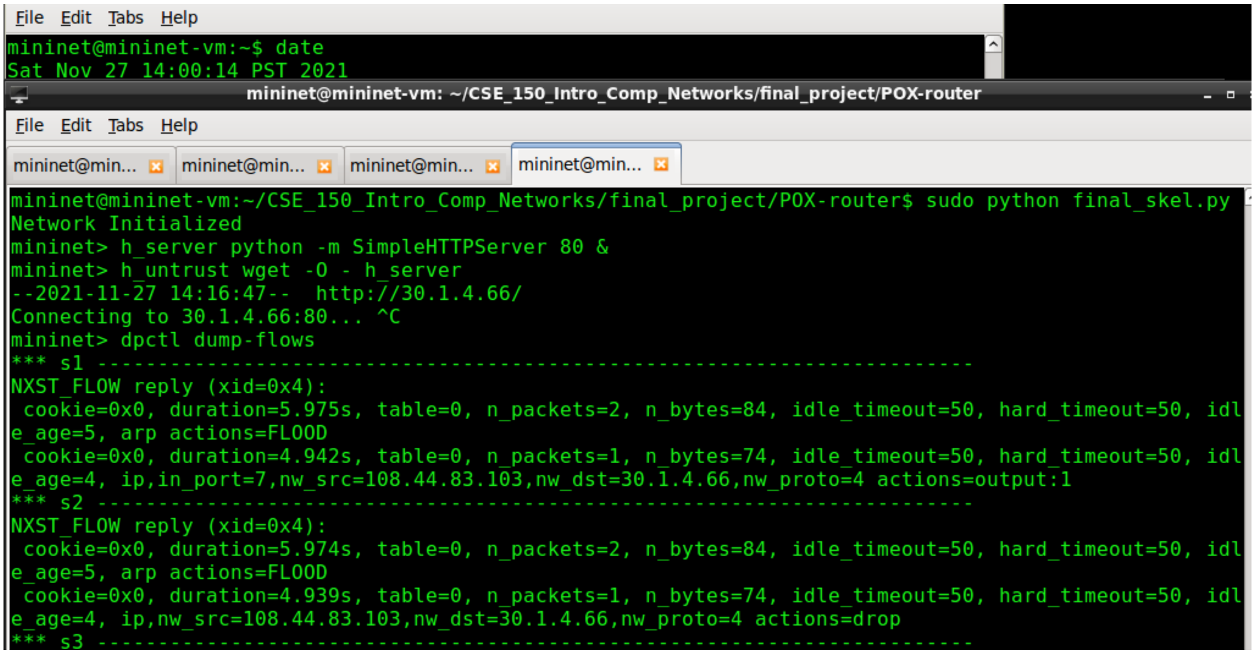
In Wireshark, we can confirm this result by filtering by HTTP traffic. We can see that device1 (20.2.1.30) makes a request to the Web Server (30.1.4.66) via TCP and gets an HTTP 200 OK response from the server along with some html data.

b) h_untrust wget -O - h_server

With this command, Untrusted Host makes a HTTP request to the Web Server and prints the response to the console. By the network rules, Untrusted Host is not allowed to transmit any IP traffic to Web Server, so the traffic should be dropped. In the Mininet console, we can see that Untrusted Host attempts to connect to the Web Server on port 80, but the traffic is unable to reach its destination:

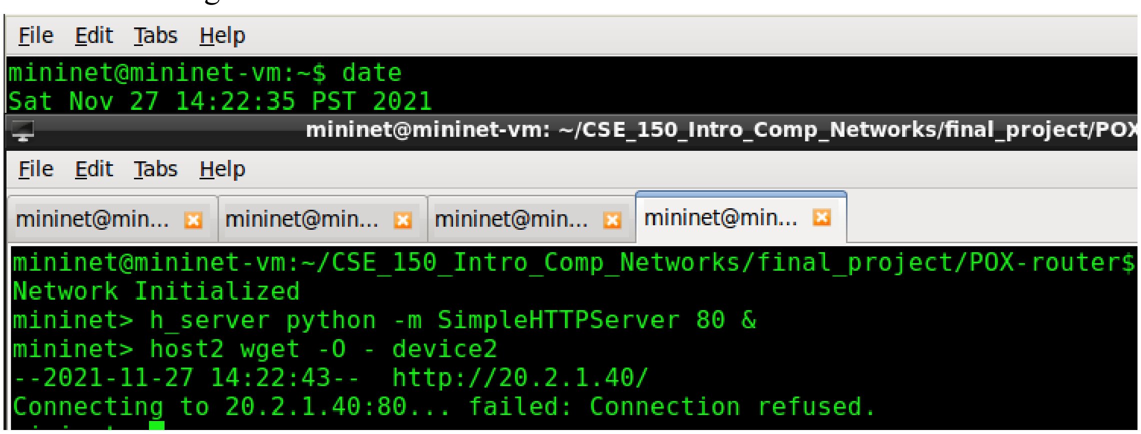


This result is confirmed by looking at the flow table which shows that the packets from Untrusted Host are routed through the Core Switch (s1) to the Data Center Switch (s2) where they are subsequently dropped.



c) host2 wget -O - device2

With this command, host2 makes a HTTP request to device2 and prints the response to the console. By the network rules, host2 and device2 can exchange IP traffic, but device2 is not running an HTTP server so a HTTP connection cannot not be made.



The result is verified by the flow table, which shows that IP traffic was allowed to transmit from host2 to device2 via the path Floor 2 Switch 1 (s5) <-> Core Switch (s1) <-> Floor 1 Switch 2 (s4).

File Edit Tabs Help

```
mininet@mininet-vm:~$ date
Sat Nov 27 14:22:35 PST 2021
```

mininet@mininet-vm: ~/CSE_150_Intro_Comp_Networks/final_project/POX-router

File Edit Tabs Help

mininet@min... | mininet@min... | mininet@min... | mininet@min...

```
mininet@mininet-vm:~/CSE_150_Intro_Comp_Networks/final_project/POX-router$ sudo python final_skel.py
Network Initialized
mininet> h_server python -m SimpleHTTPServer 80 &
mininet> host2 wget -O - device2
--2021-11-27 14:22:43--  http://20.2.1.40/
Connecting to 20.2.1.40:80... failed: Connection refused.
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=21.665s, table=0, n_packets=4, n_bytes=168, idle_timeout=50, hard_timeout=50, i
dle_age=16, arp actions=FLOOD
 cookie=0x0, duration=21.649s, table=0, n_packets=1, n_bytes=74, idle_timeout=50, hard_timeout=50, id
le_age=21, ip,in_port=4,nw_src=10.2.7.20,nw_dst=20.2.1.40,nw_proto=4 actions=output:3
 cookie=0x0, duration=21.625s, table=0, n_packets=1, n_bytes=54, idle_timeout=50, hard_timeout=50, id
le_age=21, ip,in_port=3,nw_src=20.2.1.40,nw_dst=10.2.7.20,nw_proto=4 actions=output:4
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=21.661s, table=0, n_packets=3, n_bytes=126, idle_timeout=50, hard_timeout=50, i
dle_age=16, arp actions=FLOOD
*** s3 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=21.661s, table=0, n_packets=3, n_bytes=126, idle_timeout=50, hard_timeout=50, i
dle_age=16, arp actions=FLOOD
*** s4 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=21.67s, table=0, n_packets=4, n_bytes=168, idle_timeout=50, hard_timeout=50, id
le_age=16, arp actions=FLOOD
 cookie=0x0, duration=21.654s, table=0, n_packets=1, n_bytes=74, idle_timeout=50, hard_timeout=50, id
le_age=21, ip,in_port=1,nw_src=10.2.7.20,nw_dst=20.2.1.40,nw_proto=4 actions=output:3
 cookie=0x0, duration=21.638s, table=0, n_packets=1, n_bytes=54, idle_timeout=50, hard_timeout=50, id
le_age=21, ip,in_port=3,nw_src=20.2.1.40,nw_dst=10.2.7.20,nw_proto=4 actions=output:1
*** s5 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=21.68s, table=0, n_packets=4, n_bytes=168, idle_timeout=50, hard_timeout=50, id
le_age=16, arp actions=FLOOD
 cookie=0x0, duration=21.632s, table=0, n_packets=1, n_bytes=54, idle_timeout=50, hard_timeout=50, id
le_age=21, ip,in_port=1,nw_src=20.2.1.40,nw_dst=10.2.7.20,nw_proto=4 actions=output:3
 cookie=0x0, duration=21.664s, table=0, n_packets=1, n_bytes=74, idle_timeout=50, hard_timeout=50, id
le_age=21, ip,in_port=3,nw_src=10.2.7.20,nw_dst=20.2.1.40,nw_proto=4 actions=output:1
*** s6 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=21.671s, table=0, n_packets=3, n_bytes=126, idle_timeout=50, hard_timeout=50, i
dle_age=16, arp actions=FLOOD
```