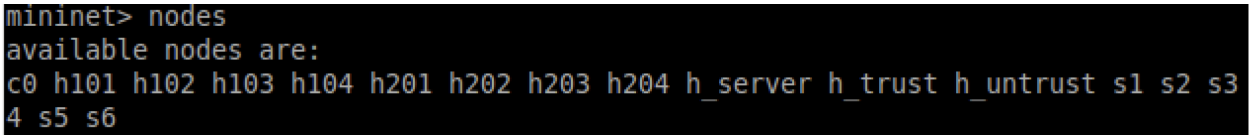
CSE-150 Final Project

I used the grading rubric for my report design and for prompts to demonstrate my code.

Mininet Topology

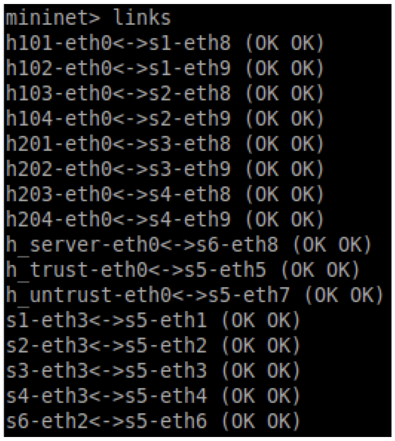
Devices are successfully created



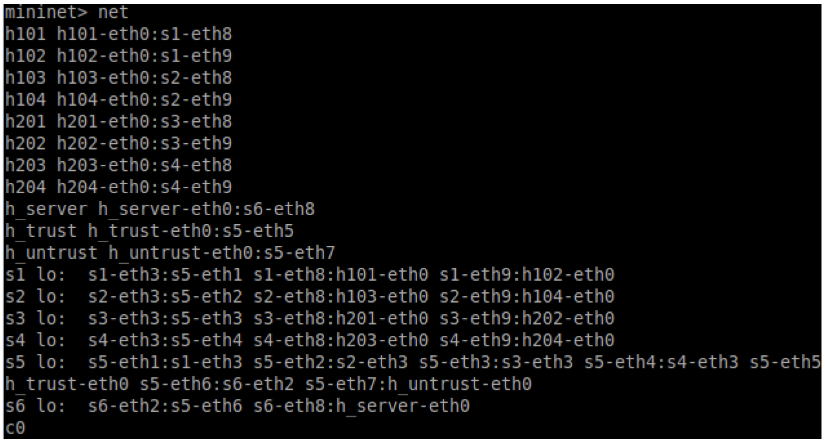
```
mininet> nodes
available nodes are:
c0 h101 h102 h103 h104 h201 h202 h203 h204 h_server h_trust h_untrust s1 s2 s3
4 s5 s6
```

‘nodes’ displays all hosts, switches, and controllers. We successfully create controller c0, our hosts, and s1 (floor 1 switch 1), s2 (floor 1 switch 2), s3 (floor 2 switch 1), s4 (floor 2 switch 2), s5 (core switch), and s6 (data center switch).

Links are successfully created, and the topology is correct



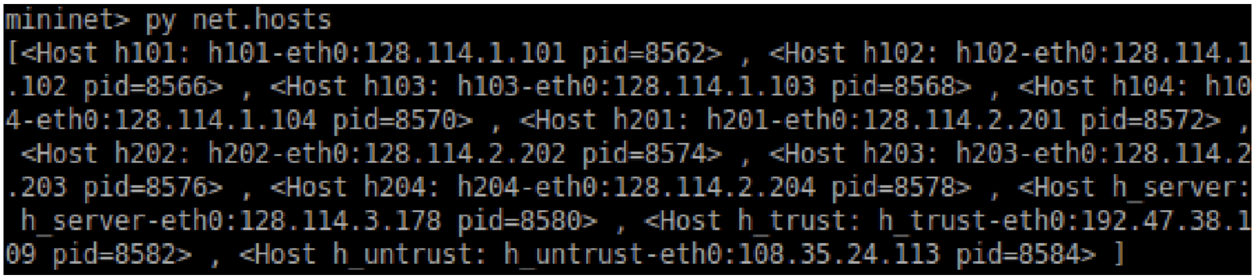
```
mininet> links
h101-eth0<->s1-eth8 (OK OK)
h102-eth0<->s1-eth9 (OK OK)
h103-eth0<->s2-eth8 (OK OK)
h104-eth0<->s2-eth9 (OK OK)
h201-eth0<->s3-eth8 (OK OK)
h202-eth0<->s3-eth9 (OK OK)
h203-eth0<->s4-eth8 (OK OK)
h204-eth0<->s4-eth9 (OK OK)
h_server-eth0<->s6-eth8 (OK OK)
h_trust-eth0<->s5-eth5 (OK OK)
h_untrust-eth0<->s5-eth7 (OK OK)
s1-eth3<->s5-eth1 (OK OK)
s2-eth3<->s5-eth2 (OK OK)
s3-eth3<->s5-eth3 (OK OK)
s4-eth3<->s5-eth4 (OK OK)
s6-eth2<->s5-eth6 (OK OK)
```



```
mininet> net
h101 h101-eth0:s1-eth8
h102 h102-eth0:s1-eth9
h103 h103-eth0:s2-eth8
h104 h104-eth0:s2-eth9
h201 h201-eth0:s3-eth8
h202 h202-eth0:s3-eth9
h203 h203-eth0:s4-eth8
h204 h204-eth0:s4-eth9
h_server h_server-eth0:s6-eth8
h_trust h_trust-eth0:s5-eth5
h_untrust h_untrust-eth0:s5-eth7
s1 lo:  s1-eth3:s5-eth1 s1-eth8:h101-eth0 s1-eth9:h102-eth0
s2 lo:  s2-eth3:s5-eth2 s2-eth8:h103-eth0 s2-eth9:h104-eth0
s3 lo:  s3-eth3:s5-eth3 s3-eth8:h201-eth0 s3-eth9:h202-eth0
s4 lo:  s4-eth3:s5-eth4 s4-eth8:h203-eth0 s4-eth9:h204-eth0
s5 lo:  s5-eth1:s1-eth3 s5-eth2:s2-eth3 s5-eth3:s3-eth3 s5-eth4:s4-eth3 s5-eth5
h_trust-eth0 s5-eth6:s6-eth2 s5-eth7:h_untrust-eth0
s6 lo:  s6-eth2:s5-eth6 s6-eth8:h_server-eth0
c0
```

‘links’ displays our topology, and we can see each host connected to the proper switch, and each switch connected to the core switch. ‘net’ is very similar.

IP addresses are correct

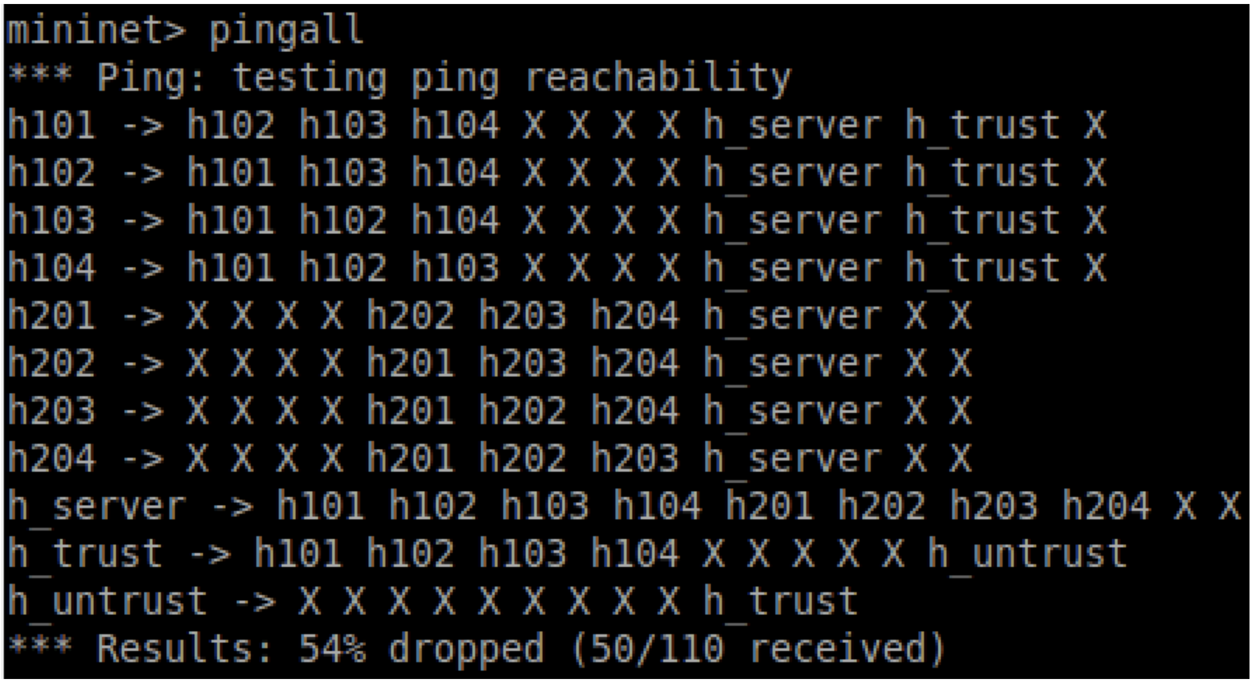


```
mininet> py net.hosts
[<Host h101: h101-eth0:128.114.1.101 pid=8562> , <Host h102: h102-eth0:128.114.1
.102 pid=8566> , <Host h103: h103-eth0:128.114.1.103 pid=8568> , <Host h104: h10
4-eth0:128.114.1.104 pid=8570> , <Host h201: h201-eth0:128.114.2.201 pid=8572> ,
 <Host h202: h202-eth0:128.114.2.202 pid=8574> , <Host h203: h203-eth0:128.114.2
.203 pid=8576> , <Host h204: h204-eth0:128.114.2.204 pid=8578> , <Host h_server:
 h_server-eth0:128.114.3.178 pid=8580> , <Host h_trust: h_trust-eth0:192.47.38.1
09 pid=8582> , <Host h_untrust: h_untrust-eth0:108.35.24.113 pid=8584> ]
```

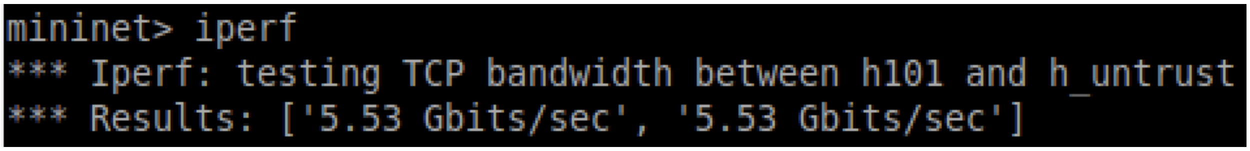
‘py net.hosts’ displays all ip addresses of each host, and we see that each one is correct.

Pox Controller

Hosts can communicate, Trusted Host cannot send ICMP traffic to Host 201 to 204, Trusted Host can send ICMP traffic to Host 101 to 104, Host 101 to 104 cannot send ICMP traffic to Host 201 to 204, Untrusted Host cannot send ICMP traffic to Host 101-104 or 201-204, Untrusted/Trusted Host cannot send any traffic to the LLM Server



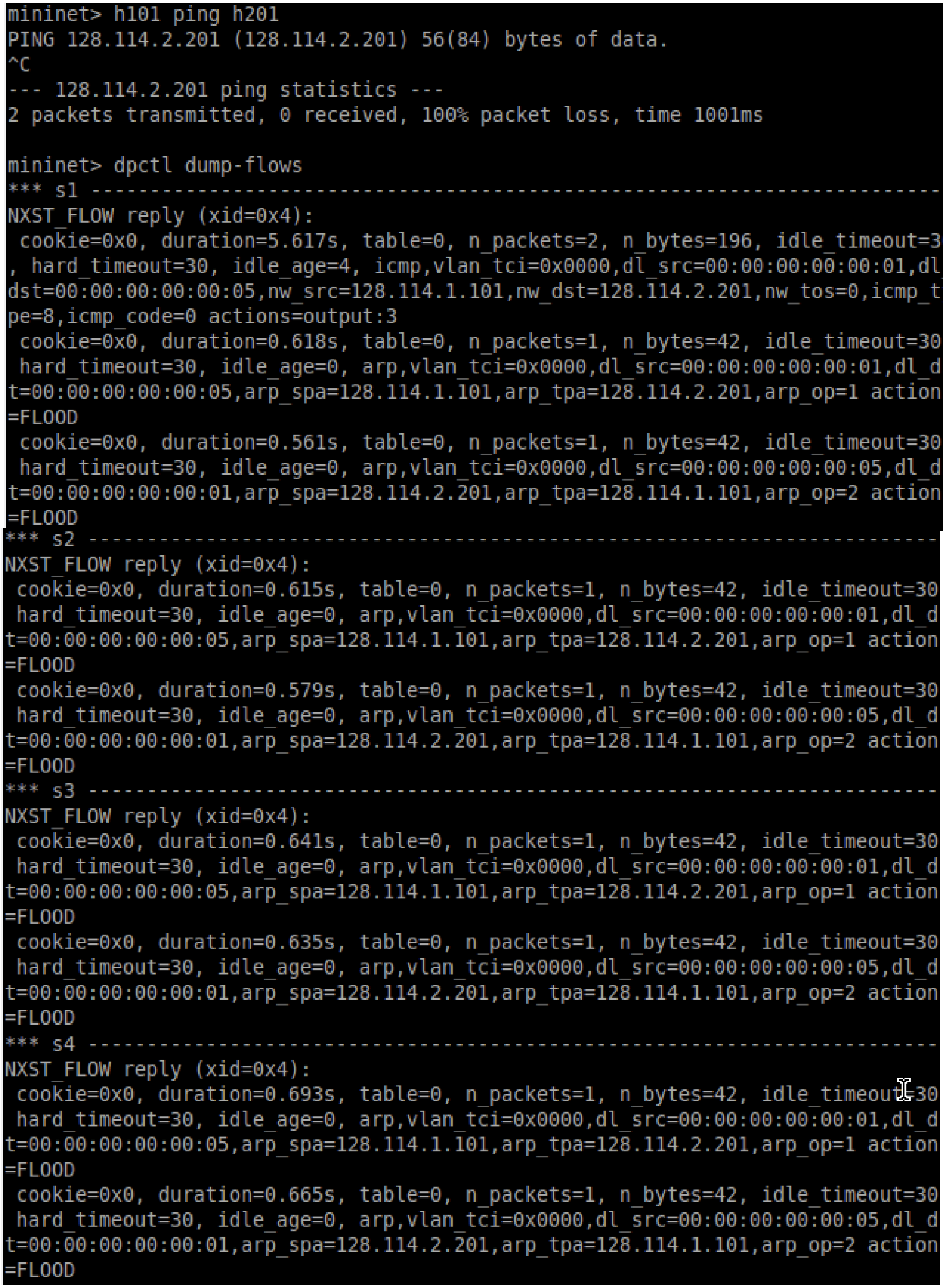
```
mininet> pingall
*** Ping: testing ping reachability
h101 -> h102 h103 h104 X X X X h_server h_trust X
h102 -> h101 h103 h104 X X X X h_server h_trust X
h103 -> h101 h102 h104 X X X X h_server h_trust X
h104 -> h101 h102 h103 X X X X h_server h_trust X
h201 -> X X X X h202 h203 h204 h_server X X
h202 -> X X X X h201 h203 h204 h_server X X
h203 -> X X X X h201 h202 h204 h_server X X
h204 -> X X X X h201 h202 h203 h_server X X
h_server -> h101 h102 h103 h104 h201 h202 h203 h204 X X
h_trust -> h101 h102 h103 h104 X X X X X h_untrust
h_untrust -> X X X X X X X X X h_trust
*** Results: 54% dropped (50/110 received)
```



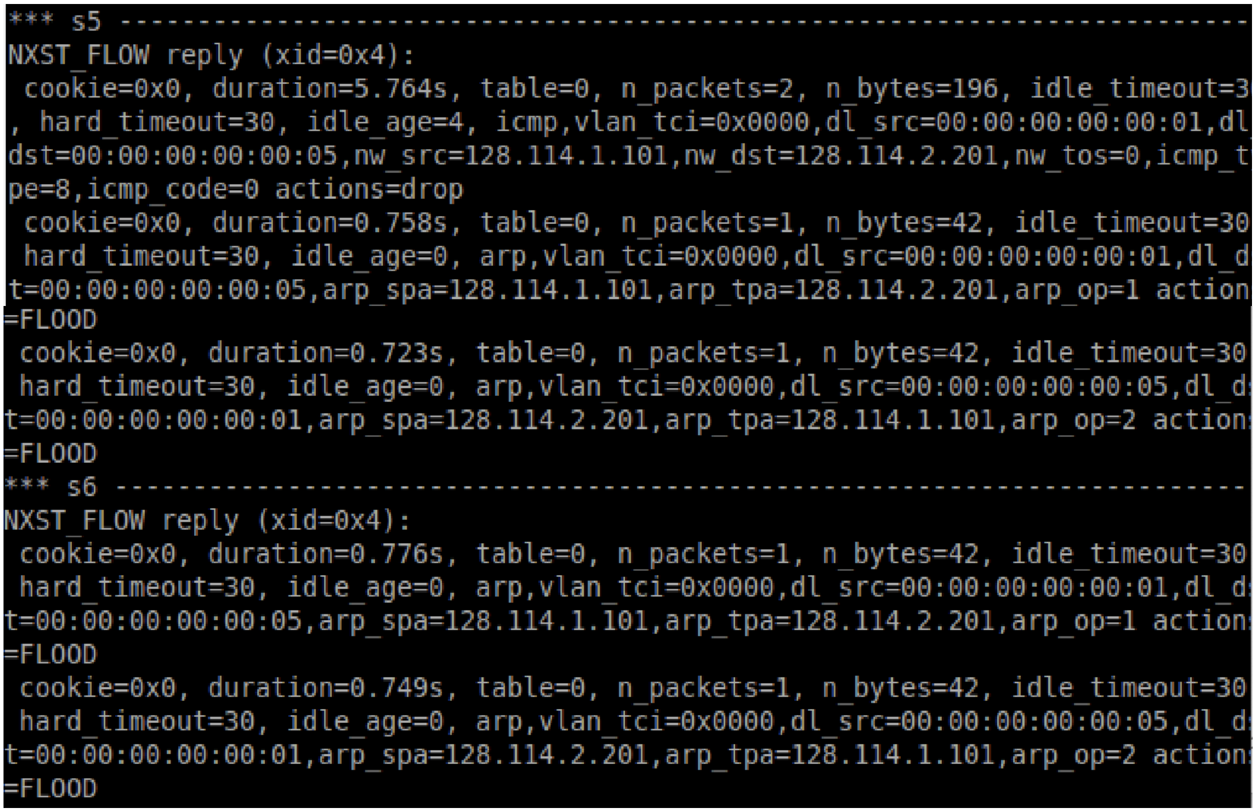
```
mininet> iperf
*** Iperf: testing TCP bandwidth between h101 and h_untrust
*** Results: ['5.53 Gbits/sec', '5.53 Gbits/sec']
```

Pingall shows proper communication and ICMP connectivity between hosts as described in the assignment. Iperf is used to show our maximum TCP/UDP bandwidth performance of the network.

Rules are installed in flow table



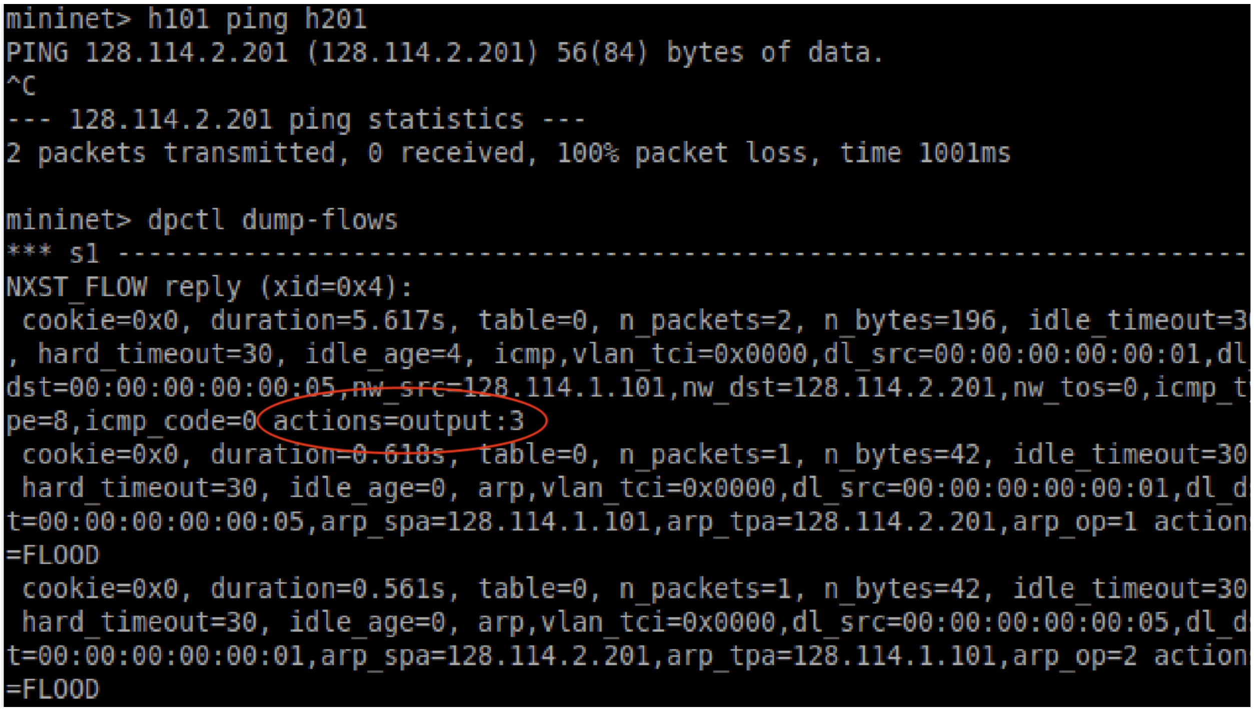
```
mininet> h101 ping h201
PING 128.114.2.201 (128.114.2.201) 56(84) bytes of data.
^C
--- 128.114.2.201 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1001ms

mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=5.617s, table=0, n_packets=2, n_bytes=196, idle_timeout=3
, hard_timeout=30, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_
dst=00:00:00:00:00:05,nw_src=128.114.1.101,nw_dst=128.114.2.201,nw_tos=0,icmp_t
pe=8,icmp_code=0 actions=output:3
 cookie=0x0, duration=0.618s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_d
t=00:00:00:00:00:05,arp_spa=128.114.1.101,arp_tpa=128.114.2.201,arp_op=1 action
=FLOOD
 cookie=0x0, duration=0.561s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:05,dl_d
t=00:00:00:00:00:01,arp_spa=128.114.2.201,arp_tpa=128.114.1.101,arp_op=2 action
=FLOOD
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=0.615s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_d
t=00:00:00:00:00:05,arp_spa=128.114.1.101,arp_tpa=128.114.2.201,arp_op=1 action
=FLOOD
 cookie=0x0, duration=0.579s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:05,dl_d
t=00:00:00:00:00:01,arp_spa=128.114.2.201,arp_tpa=128.114.1.101,arp_op=2 action
=FLOOD
*** s3 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=0.641s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_d
t=00:00:00:00:00:05,arp_spa=128.114.1.101,arp_tpa=128.114.2.201,arp_op=1 action
=FLOOD
 cookie=0x0, duration=0.635s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:05,dl_d
t=00:00:00:00:00:01,arp_spa=128.114.2.201,arp_tpa=128.114.1.101,arp_op=2 action
=FLOOD
*** s4 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=0.693s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_d
t=00:00:00:00:00:05,arp_spa=128.114.1.101,arp_tpa=128.114.2.201,arp_op=1 action
=FLOOD
 cookie=0x0, duration=0.665s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:05,dl_d
t=00:00:00:00:00:01,arp_spa=128.114.2.201,arp_tpa=128.114.1.101,arp_op=2 action
=FLOOD
```



```
*** s5 ----------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=5.764s, table=0, n_packets=2, n_bytes=196, idle_timeout=3
, hard_timeout=30, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
dst=00:00:00:00:00:05,nw_src=128.114.1.101,nw_dst=128.114.2.201,nw_tos=0,icmp_t
pe=8,icmp_code=0 actions=drop
 cookie=0x0, duration=0.758s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_d
t=00:00:00:00:00:05,arp_spa=128.114.1.101,arp_tpa=128.114.2.201,arp_op=1 action
=FLOOD
 cookie=0x0, duration=0.723s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:05,dl_d
t=00:00:00:00:00:01,arp_spa=128.114.2.201,arp_tpa=128.114.1.101,arp_op=2 action
=FLOOD
*** s6 ----------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=0.776s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_d
t=00:00:00:00:00:05,arp_spa=128.114.1.101,arp_tpa=128.114.2.201,arp_op=1 action
=FLOOD
 cookie=0x0, duration=0.749s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:05,dl_d
t=00:00:00:00:00:01,arp_spa=128.114.2.201,arp_tpa=128.114.1.101,arp_op=2 action
=FLOOD
```

(I use ‘h101 ping h102’ as an example to trigger the pox controller to install flow rules in the switches) I call ‘dpctl dump-flows’ to show the flow table and installed rules (corresponding to ‘h101 ping h102’ in this example) for the switches.

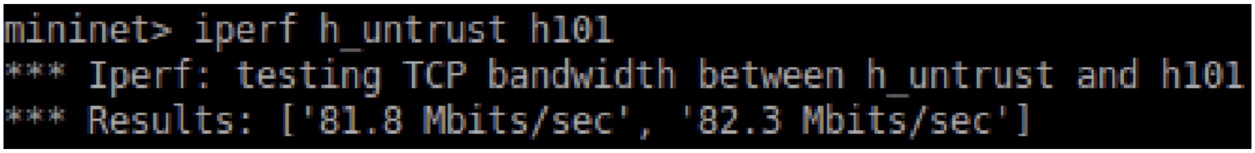
IP traffic is implemented without using OFPP_FLOOD



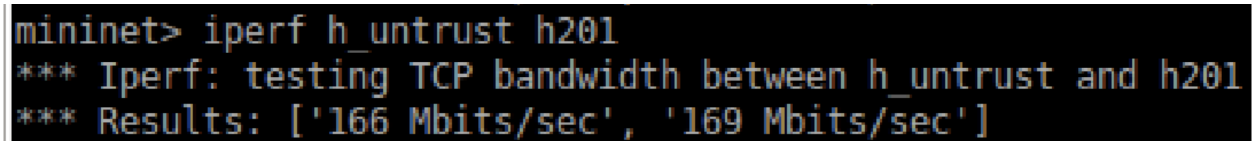
```
mininet> h101 ping h201
PING 128.114.2.201 (128.114.2.201) 56(84) bytes of data.
^C
--- 128.114.2.201 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1001ms

mininet> dpctl dump-flows
*** s1 ----------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=5.617s, table=0, n_packets=2, n_bytes=196, idle_timeout=3
, hard_timeout=30, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
dst=00:00:00:00:00:05,nw_src=128.114.1.101,nw_dst=128.114.2.201,nw_tos=0,icmp_t
pe=8,icmp_code=0 actions=output:3
 cookie=0x0, duration=0.618s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_d
t=00:00:00:00:00:05,arp_spa=128.114.1.101,arp_tpa=128.114.2.201,arp_op=1 action
=FLOOD
 cookie=0x0, duration=0.561s, table=0, n_packets=1, n_bytes=42, idle_timeout=30
 hard_timeout=30, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:05,dl_d
t=00:00:00:00:00:01,arp_spa=128.114.2.201,arp_tpa=128.114.1.101,arp_op=2 action
=FLOOD
```

As seen earlier in ‘dpctl dump-flows’, the IP traffic is handled by porting to specific ports, not to OFPP_FLOOD. All IP traffic is specifically ported to reach the proper host. Additionally, my finalcontroller_skel.py can be seen to use OFPP_FLOOD only twice, first for IP traffic that *doesn't* need to be entirely blocked/ported to a specific port, and second to flood all other non-IP traffic.

Untrusted Host can send traffic (not ICMP) to the hosts



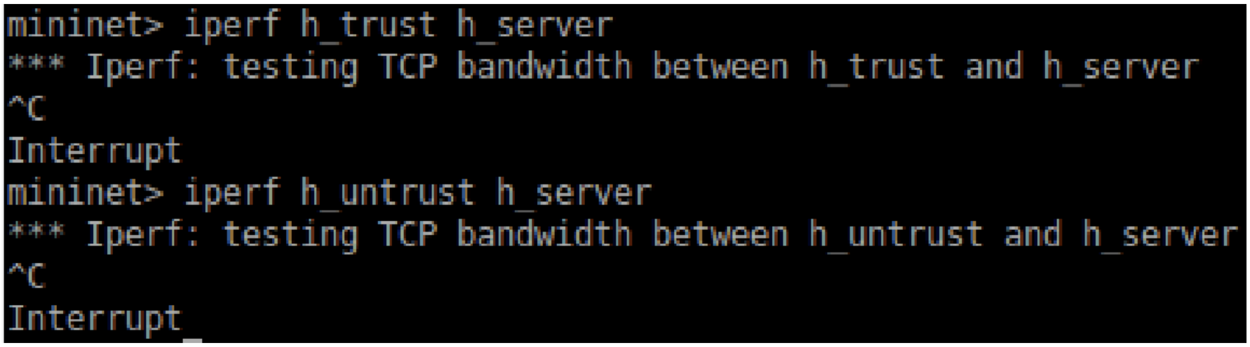
```
mininet> iperf h_untrust h101
*** Iperf: testing TCP bandwidth between h_untrust and h101
*** Results: ['81.8 Mbits/sec', '82.3 Mbits/sec']
```



```
mininet> iperf h_untrust h201
*** Iperf: testing TCP bandwidth between h_untrust and h201
*** Results: ['166 Mbits/sec', '169 Mbits/sec']
```

iperf shows a TCP connection between h_untrust and h101, which wouldn't be possible if all traffic was blocked from h_untrust to h101. This can be replicated for each host (201 is included as well as an example) except for h_server.

Untrusted/Trust Host cannot send any traffic to the LLM Server



```
mininet> iperf h_trust h_server
*** Iperf: testing TCP bandwidth between h_trust and h_server
^C
Interrupt
mininet> iperf h_untrust h_server
*** Iperf: testing TCP bandwidth between h_untrust and h_server
^C
Interrupt_
```

I ran iperf to attempt to establish a TCP connection between trusted and untrusted hosts and the server, and after plenty of time I ran control+C to Interrupt. This failure of iperf suggests no traffic can be sent from untrusted and trusted hosts to the server.