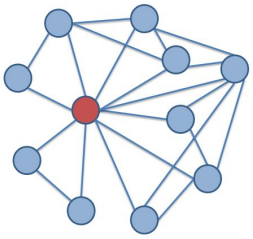


CMPE 150 - Lab 1

Kevin Romero Peces-Barba - 1635745

kromero8@ucsc.edu

1. In Mininet change the default configuration to have 4 hosts connected to a switch.

The following script (also included in the submit) does it. It also can get a parameter representing the number of hosts to add.

```python
#!/usr/bin/python
import sys

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.cli import CLI

class Topology(Topo):
  "Basic topology"
  def __init__(self, n=2):
    Topo.__init__(self)
    # Set up topology
    switch = self.addSwitch('s1')
    for i in range(n):
      host = self.addHost('h%s' % (i+1))
      self.addLink(host, switch)

if __name__ == '__main__':
  hosts = 4;
  if len(sys.argv) == 2:
    hosts = int(sys.argv[1])
  topo = Topology(n=hosts)
  net = Mininet( topo=topo )
  net.start()

  CLI(net)

  net.stop()
```

2. Save a screenshot of *dump* and *pingall* output. Explain what is being shown in the screenshot.

DUMP

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1409>
<Host h2: h2-eth0:10.0.0.2 pid=1411>
<Host h3: h3-eth0:10.0.0.3 pid=1413>
<Host h4: h4-eth0:10.0.0.4 pid=1415>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None pid=1420>
<Controller c0: 127.0.0.1:6653 pid=1402>
```

Dump command shows the information of each node of the network. Therefore we see each host with its connection to eth0 and the ip (as well as the pid), the switch with its connection to the controller and the controller with its interface IP.

PINGALL

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

Pingall sends a ping from every host to each one of the others. In the screenshot we can see exactly this and that there were no packets lost.

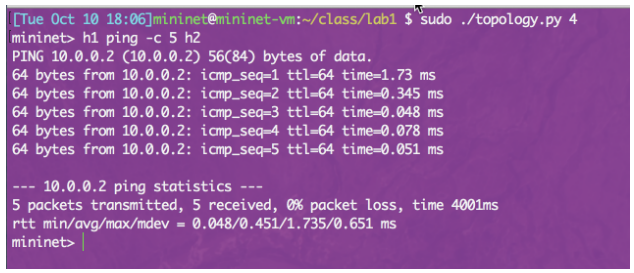
3. Run the iperf command as well, and screenshot the output, how fast is the connect?

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['36.3 Gbits/sec', '36.4 Gbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['35.9 Gbits/sec', '36.0 Gbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['35.2 Gbits/sec', '35.3 Gbits/sec']
```

We can see that the connection speed is around 35-36 Gbits/sec (symmetric).

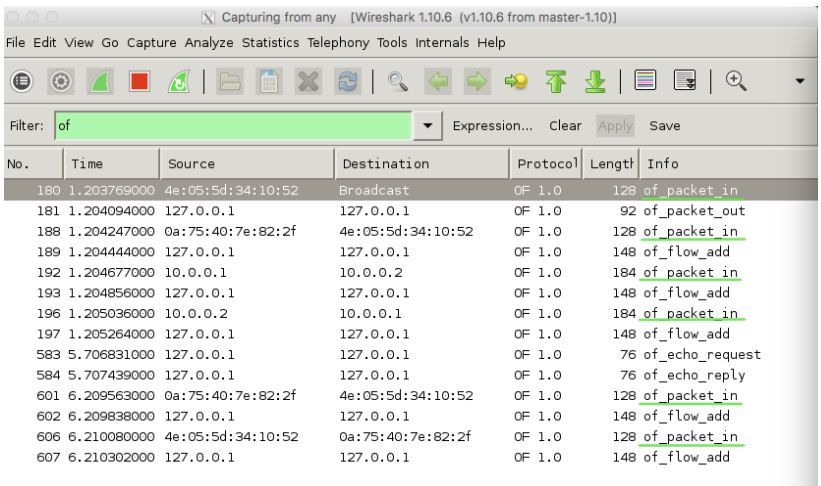
4. Run wireshark, and using the display filter, filter for “of”.

a. Run ping from a host to any other host using *hX ping -c 5 hY*. How many “of_packet_in” messages show up. Take a screenshot of your results.



```
[Tue Oct 10 18:06]mininet@mininet-vm:~/class/lab1 $ sudo ./topology.py 4
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.73 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.345 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.051 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.048/0.451/1.735/0.651 ms
mininet>
```



Capturing from any [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

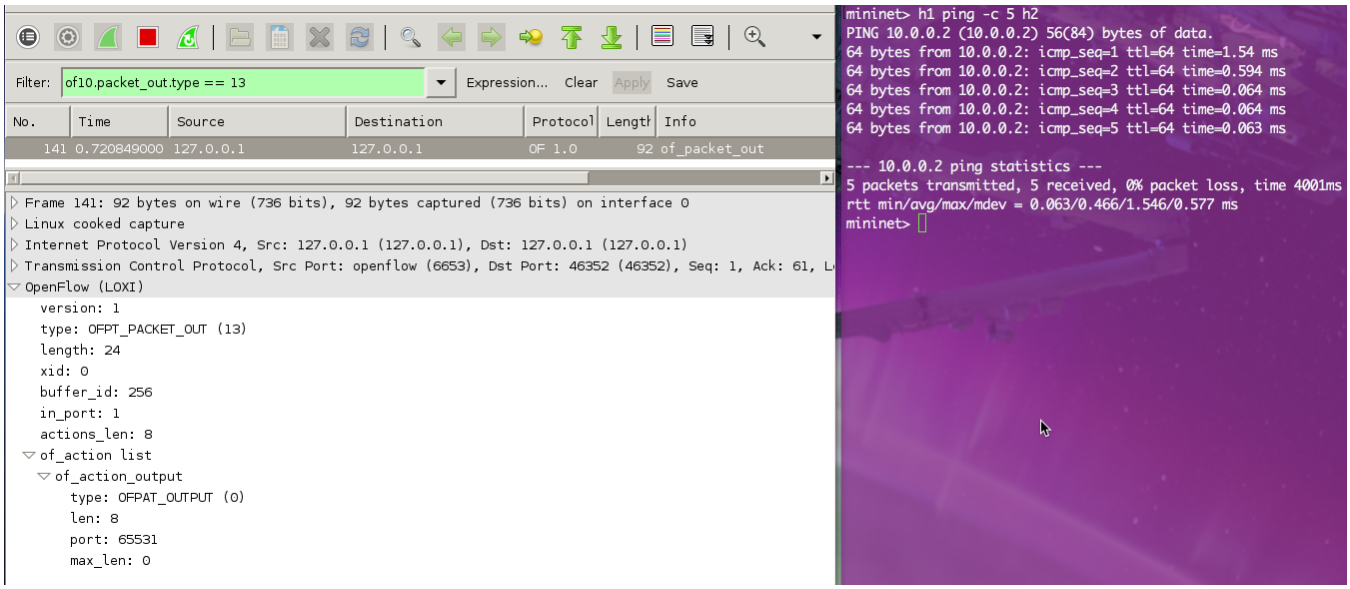
Filter: of | Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
180	1.203769000	4e:05:5d:34:10:52	Broadcast	OF 1.0	128	of_packet_in
181	1.204094000	127.0.0.1	127.0.0.1	OF 1.0	92	of_packet_out
188	1.204247000	0a:75:40:7e:82:2f	4e:05:5d:34:10:52	OF 1.0	128	of_packet_in
189	1.204444000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
192	1.204677000	10.0.0.1	10.0.0.2	OF 1.0	184	of_packet_in
193	1.204856000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
196	1.205036000	10.0.0.2	10.0.0.1	OF 1.0	184	of_packet_in
197	1.205264000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
583	5.706831000	127.0.0.1	127.0.0.1	OF 1.0	76	of_echo_request
584	5.707439000	127.0.0.1	127.0.0.1	OF 1.0	76	of_echo_reply
601	6.209563000	0a:75:40:7e:82:2f	4e:05:5d:34:10:52	OF 1.0	128	of_packet_in
602	6.209838000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
606	6.210080000	4e:05:5d:34:10:52	0a:75:40:7e:82:2f	OF 1.0	128	of_packet_in
607	6.210302000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add

We can capture six *of_packet_in* messages.

b. What is the source and destination IP addresses for these entries? Find another packet that patches de “of” filter with the OpenFlow typefield set to “OFPT_PACKET_OUT”. What is the source and destination address for this entry? Take screenshots showing your results.

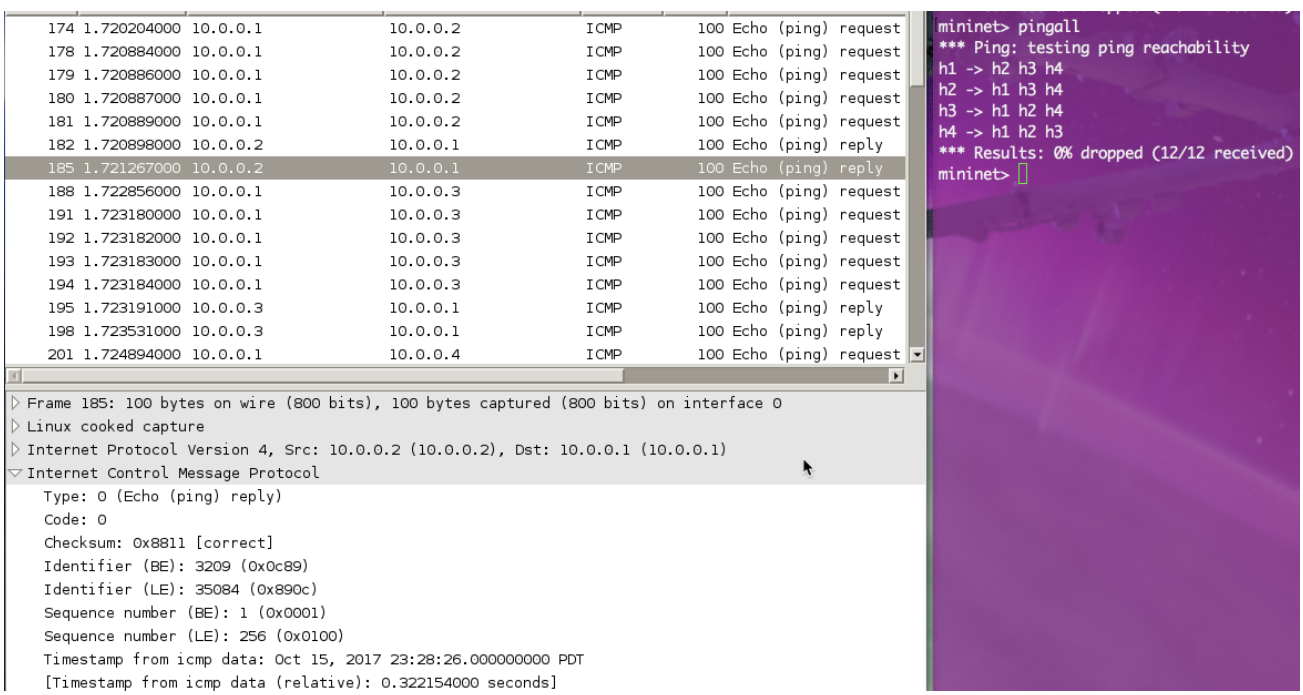
The first one is from h1 to the switch (the controller) to make it broadcast to all hosts. The rest are between h1 and h2.



It's a packet with source and destination the switch interface for the controller.

c. Replace the display filter for "of" to "icmp && not of". Run pingall again, how many entries are generated in wireshark? What types of icmp entries show up? Take a screenshot of your results.

Wireshark shows 57 packets:



There're two different icmp entries: echo reply and echo request.