

```

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)
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=4872>
<Host h2: h2-eth0:10.0.0.2 pid=4876>
<Host h3: h3-eth0:10.0.0.3 pid=4878>
<Host h4: h4-eth0:10.0.0.4 pid=4880>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=4885>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=4888>
<Controller c0: 127.0.0.1:6633 pid=4865>
mininet>

```

1. The pingall command shows that there is a valid connection between the 4 different hosts where each host can ping to the other 3 hosts. Out of the 12 pings sent, 12 of them were received meaning there is a stable connection. The dump command shows all the information about the nodes, which produces information about the 4 hosts, the 2 switches, and the controller which is typically outside of the VM. It also shows the IP address for each of the hosts.

```

mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['30.4 Gbits/sec', '30.5 Gbits/sec']

```

2. Running the iperf command on our given topography gives the bandwidth speed, which is 30.4 Gbits/sec and 30.5 Gbits/sec, respectively.

60	25.018067000	06:f8:dc:e9:b3:4a	Broadcast	0F 1.0	128 of_packet_in
68	25.019060000	06:f8:dc:e9:b3:4a	Broadcast	0F 1.0	128 of_packet_in
69	25.019075000	1e:72:90:a2:f0:63	06:f8:dc:e9:b3:4a	0F 1.0	128 of_packet_in
78	25.020808000	10.0.0.1	10.0.0.2	0F 1.0	184 of_packet_in
82	25.021702000	10.0.0.2	10.0.0.1	0F 1.0	184 of_packet_in
109	30.038056000	1e:72:90:a2:f0:63	06:f8:dc:e9:b3:4a	0F 1.0	128 of_packet_in
114	30.041200000	06:f8:dc:e9:b3:4a	1e:72:90:a2:f0:63	0F 1.0	128 of_packet_in

3. After running the ping command between host1 and host2, we see that we have 7 messages regarding of_packet_in.

25.01873900	127.0.0.1	127.0.0.1	0F 1.0	92 of packet_out
70 25.01958800	127.0.0.1	127.0.0.1	0F 1.0	92 of packet_out

4. You'll notice that the source and destination IP addresses of the packets coming in are relative to the IP addresses of the host1 and host2 which are 10.0.0.1 and 10.0.0.2, and you will also notice that the rest have MAC addresses that match up with source and destination. From this, we have 2 out packets, one from each host, which are host1 and host2 that have the same destination IP address as the source IP address.

Filter: `icmp && not of` Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
81	21.85628500	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ed8, seq=3/768, ttl=64
80	21.85627900	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ed8, seq=3/768, ttl=64 (request in 79)
85	22.85734700	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ed8, seq=4/1024, ttl=64
84	22.85734100	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ed8, seq=4/1024, ttl=64 (request in 83)
89	23.85663900	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ed8, seq=5/1280, ttl=64
88	23.85663300	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ed8, seq=5/1280, ttl=64 (request in 87)
109	27.54347400	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ede, seq=1/256, ttl=64
108	27.54323200	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ede, seq=1/256, ttl=64 (request in 107)
143	27.55296900	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0edf, seq=1/256, ttl=64
144	27.55297100	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0edf, seq=1/256, ttl=64
147	27.55382900	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0edf, seq=1/256, ttl=64
140	27.55224400	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0edf, seq=1/256, ttl=64 (request in 139)
179	27.56337000	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ee0, seq=1/256, ttl=64
180	27.56339900	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0ee0, seq=1/256, ttl=64

Frame 109: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.0.2 (10.0.0.2), Dst: 10.0.0.1 (10.0.0.1)
Internet Control Message Protocol

0010 45 00 00 54 e5 1e 00 00 40 01 81 88 0a 00 00 02 E..T...@.....
0020 0a 00 00 01 00 00 d6 5a 0e de 00 01 2e 7a 3e 5cZ....>\
0030 b6 ec 0c 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13
0040 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23!..#

Internet Protocol Version 4 (ip)... Packets: 444 · Displayed: 100 (22.5%) · Dropped: 0 (0.0%) Profile: Default

5. Out of 444 packets, only 100 were displayed while using the filter. 50 of these packets were replies and 50 were requests. This showed the pingall results along with the h1 ping -c 5 h2 results.