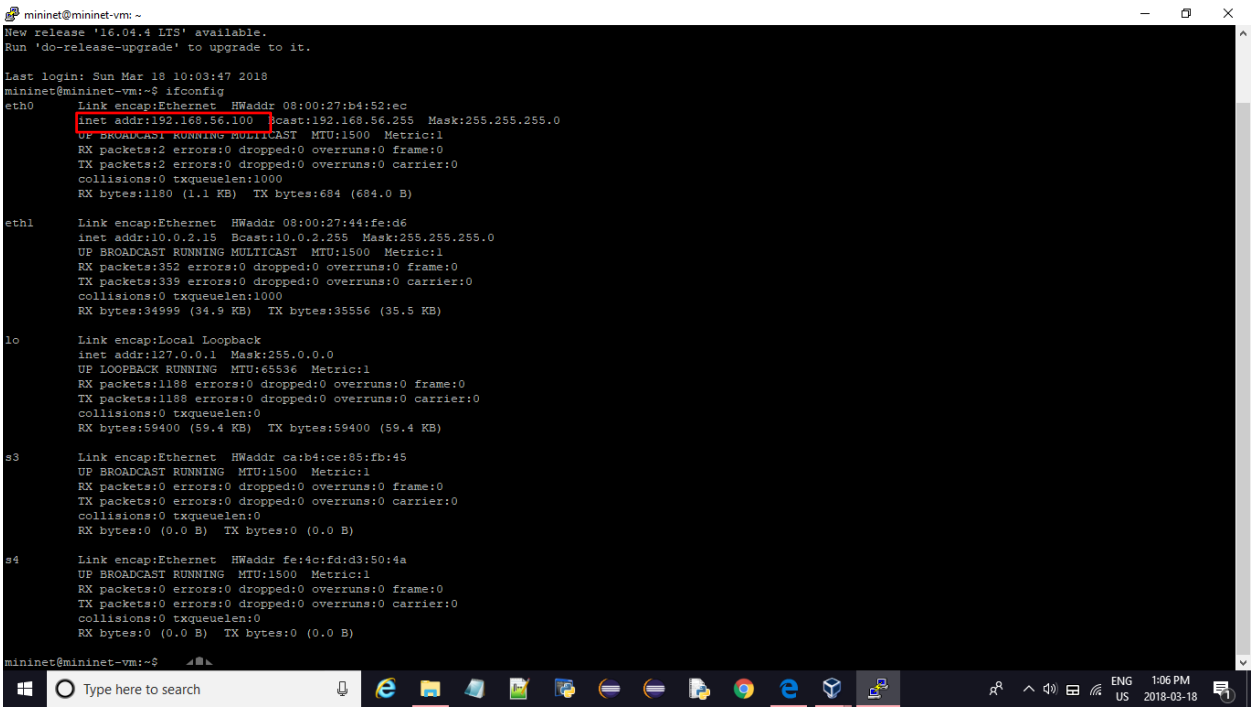
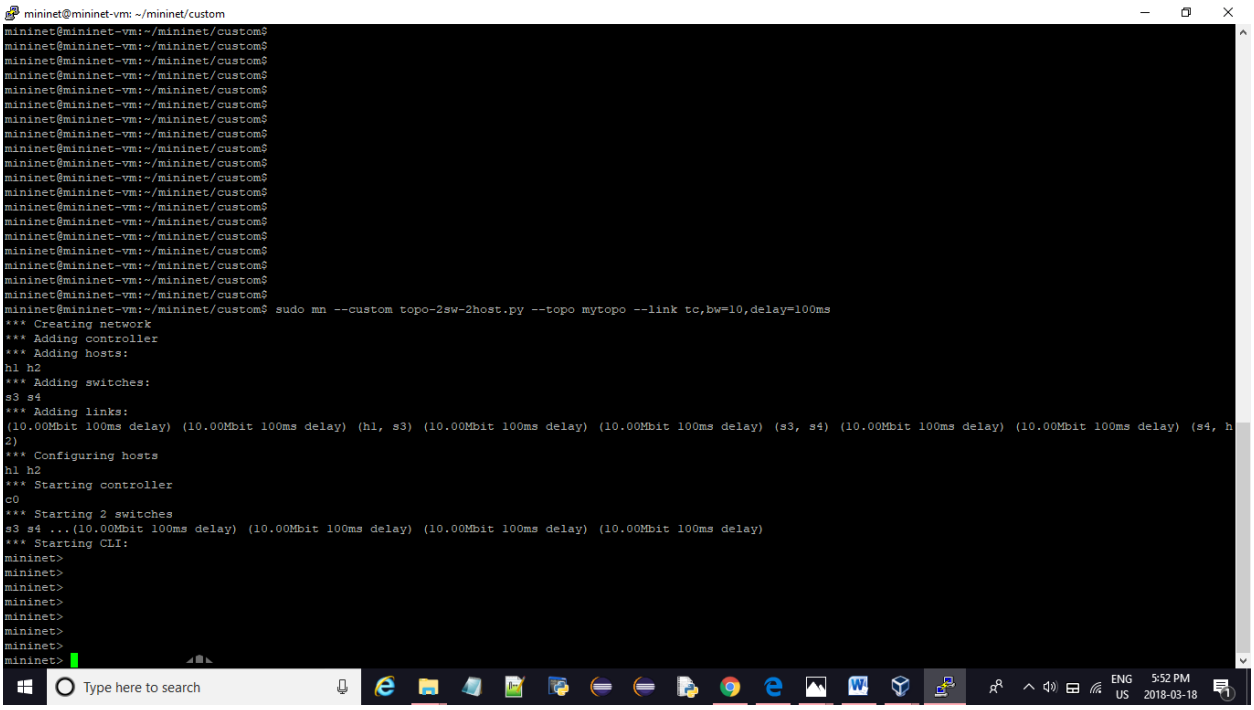
1.



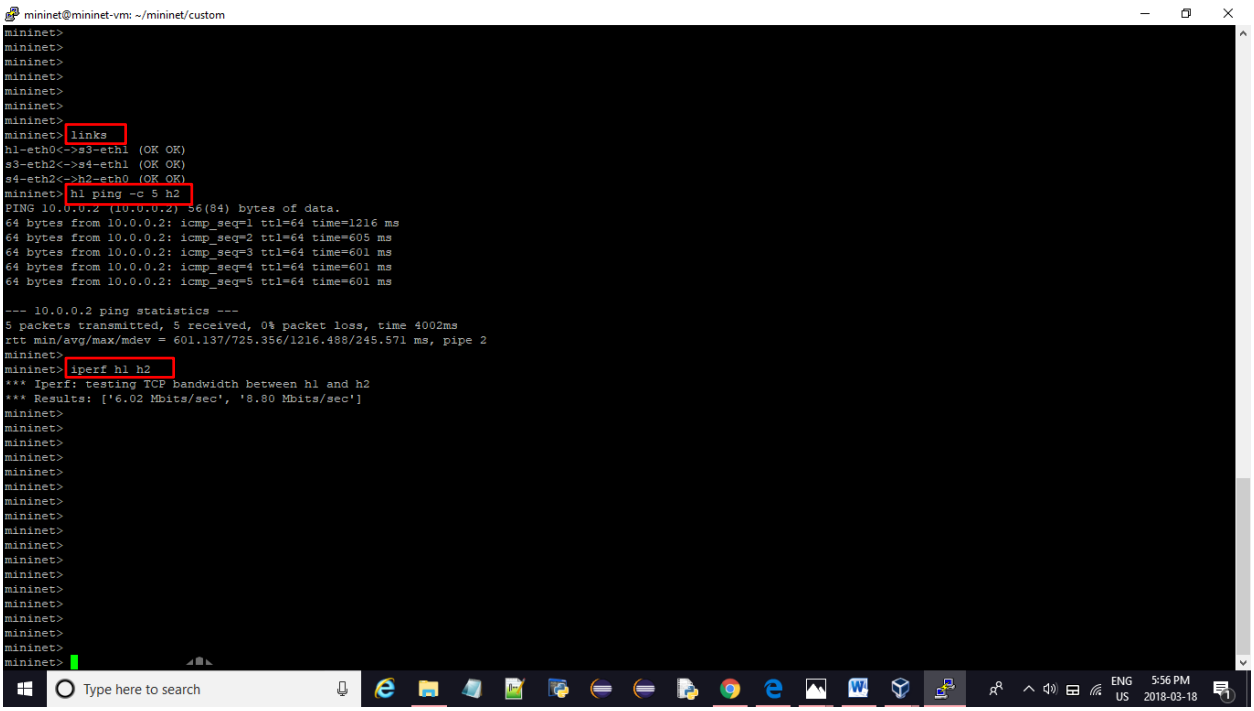
We have set up two network interfaces, one for connecting the VM to the outside through NAT, and one for the host only network. In this case, we can see eth0 is the interface for the latter.

The IP address is determined by ifconfig. As we can see here, the IP address is 192.168.56.100 because this is what we set for the host address earlier, and this is also the address at which the guests can access the host.

2.



On the picture above, we can see that a network is created. A controller is added. Two hosts are added, h1 and h2. Two switches are added, s3 and s4. Three links are created; the first link is created between h1 and s3 with bandwidth 10 Mbit and 100ms delay, the second link is created between s3 and s4 with bandwidth 10Mbit and 100ms delay, the third link is created between s4 and h2 with bandwidth 10Mbit and 100ms delay. The hosts, controller and switches launched.



Links

Three links is created successfully

- One is created between h1 through eth0 and s3 through eth1
- One is created between s3 through eth2 and s4 through eth1
- One is created between s4 through eth2 and h2 through eth0

H1 ping –c 5 h2

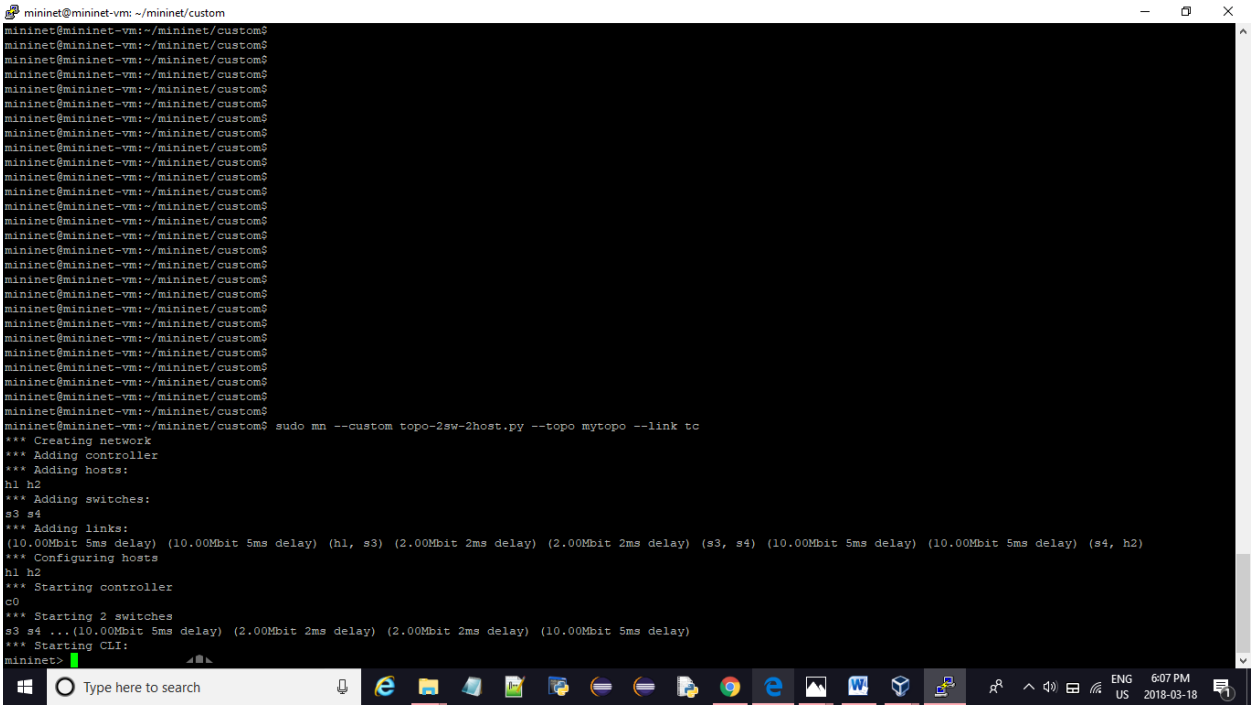
The ping command is to verify the connectivity between host 1 and host 2 is successful. As we can see, 5 packets were sent to verify the connectivity, and it all went successfully and therefore the connectivity between h1 and h2 is successful created.

We can see a much lower ping time at the second try. A flow entry covering ping traffic was previously installed in the switch, so no control traffic was generated, and the packets immediately pass through the switch.

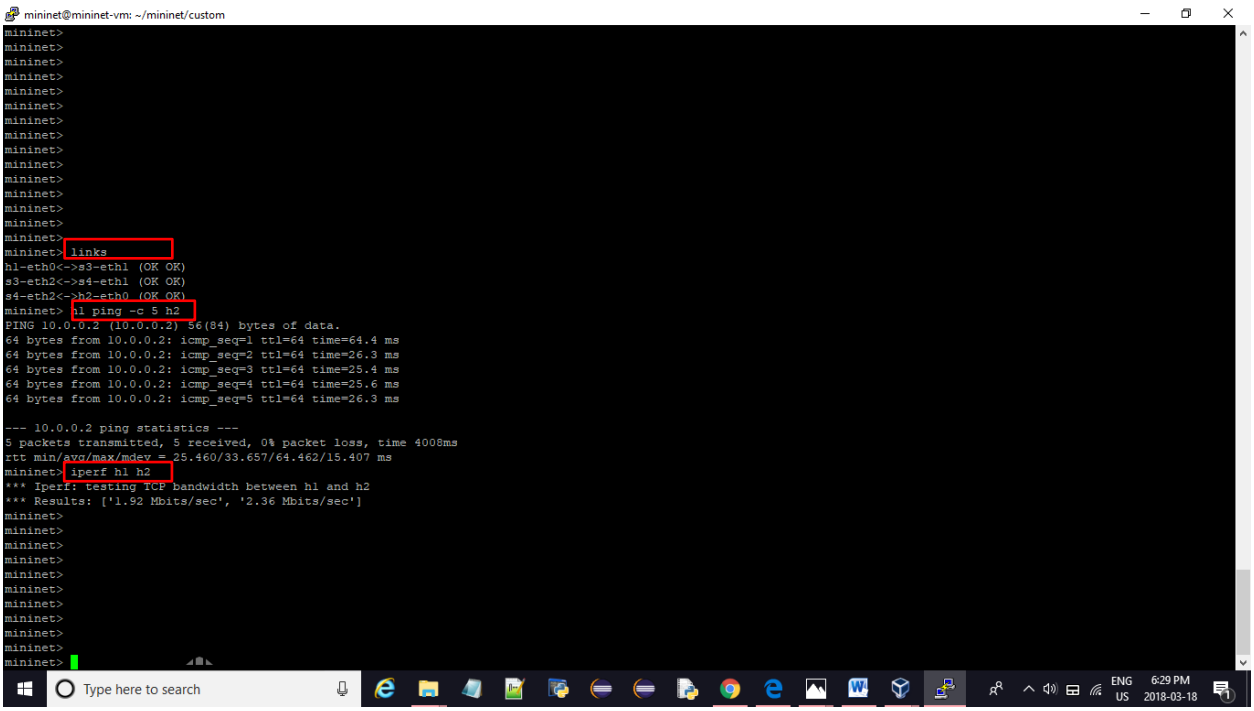
Iperf h1 h2

Iperf measured the maximum achieved bandwidth between h1 and h2. This indicates that the through put in the forward direction is 6.02Mbits/sec and the reverse direction is 8.80Mbits/sec.

3.



On the picture above, we can see that a network is created. A controller is added. Two hosts are added, h1 and h2. Two switches are added, s3 and s4. Three links are created; the first link is created between h1 and s3 with bandwidth 10 Mbit and 5ms delay, the second link is created between s3 and s4 with bandwidth 2Mbit and 2ms delay, the third link is created between s4 and h2 with bandwidth 10Mbit and 5ms delay. The hosts, controller and switches launched.



Links

Three links is created successfully

- One is created between h1 through eth0 and s3 through eth1
- One is created between s3 through eth2 and s4 through eth1
- One is created between s4 through eth2 and h2 through eth0

H1 ping –c 5 h2

The ping command is to verify the connectivity between host 1 and host 2 is successful. As we can see, 5 packets were sent to verify the connectivity, and it all went successfully and therefore the connectivity between h1 and h2 is successful created.

We can see a much lower ping time at the second try(26.3ms). A flow entry covering ping traffic was previously installed in the switch, so no control traffic was generated, and the packets immediately pass through the switch.

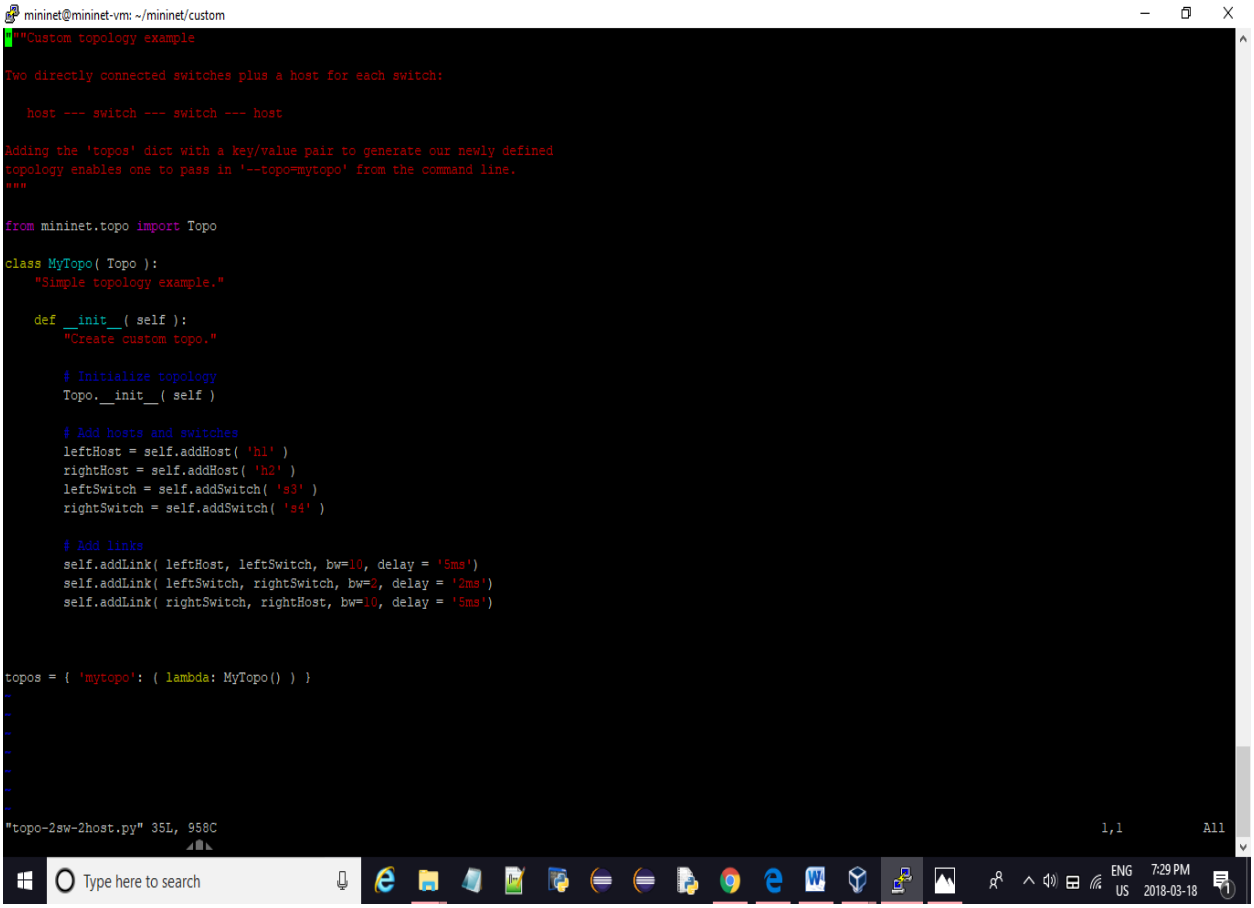
As we can see, the time is much lower compared to question 2. This happened because we have set the delay time a lot lower compared to Question 2.

Iperf h1 h2

Iperf measured the maximum achieved bandwidth between h1 and h2. This indicates that the through put in the forward direction is 1.92Mbits/sec and the reverse direction is 2.36Mbits/sec. As we can see

the throughput is slower in this case compared to Question 2 because we have set the bandwidth between switch 3 and switch 4 to 2Mbit, which is slower compared to Question2.

Modified topo-2sw-2host.py




```
"""Custom topology example

Two directly connected switches plus a host for each switch:

   host --- switch --- switch --- host

Adding the 'topos' dict with a key/value pair to generate our newly defined
topology enables one to pass in '--topo=mytopo' from the command line.
"""

from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch, bw=10, delay = '5ms')
        self.addLink( leftSwitch, rightSwitch, bw=2, delay = '2ms')
        self.addLink( rightSwitch, rightHost, bw=10, delay = '5ms')



topos = { 'mytopo': ( lambda: MyTopo() ) }
```