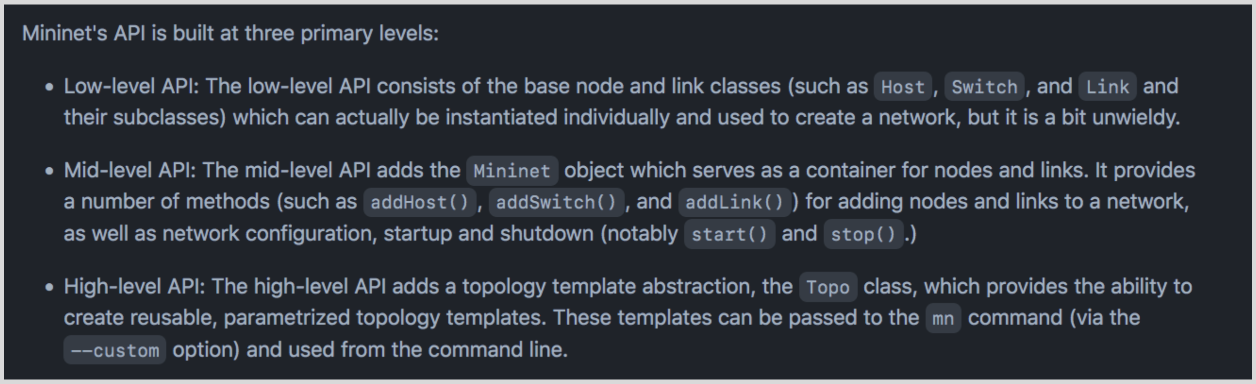
Assignment 2:

Murtadha Marzouq
ID: 800840999
Computer Comm & Networks ITCS 3166
Professor: Pu Wang

1. Create a Mininet topology script for the network topology given below

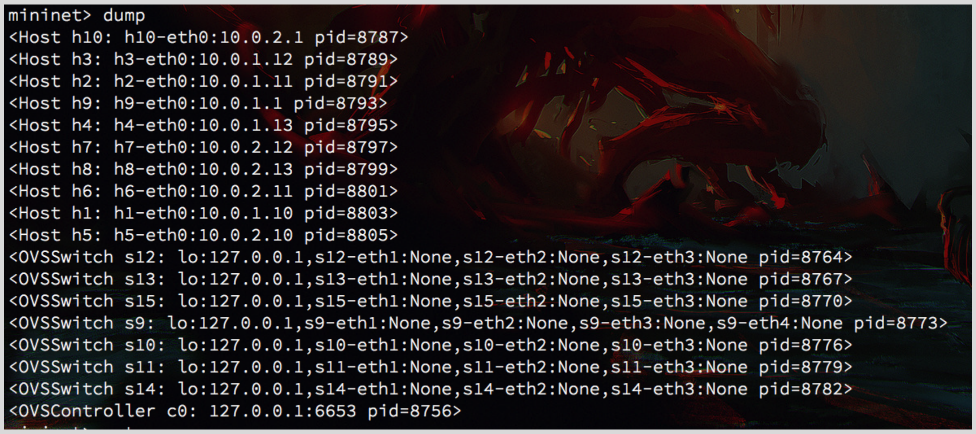
See the attached python files, but let me speak of something about Mininet:
Python code can be written in 3 ways
Low-level API: I have that version but did not submit it.
Mid-level API: I submitted this version.
High-level API: I first wrote this, but wanted to challenge myself.



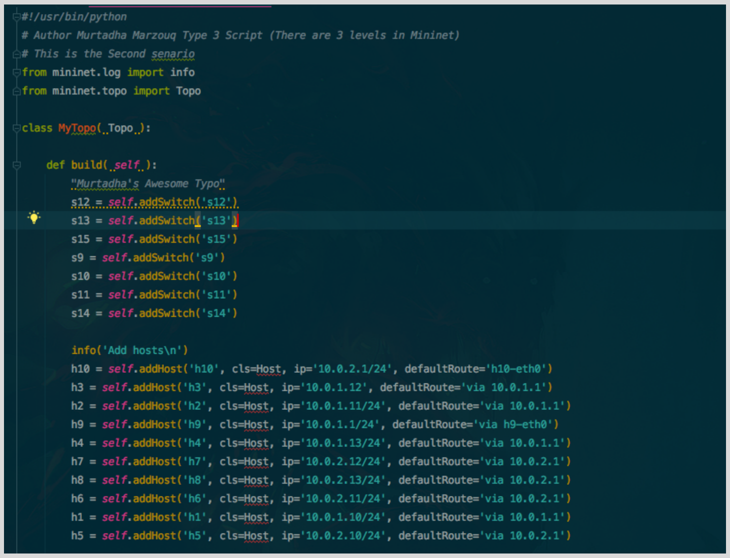
Mininet's API is built at three primary levels:

- Low-level API: The low-level API consists of the base node and link classes (such as `Host`, `Switch`, and `Link` and their subclasses) which can actually be instantiated individually and used to create a network, but it is a bit unwieldy.
- Mid-level API: The mid-level API adds the `Mininet` object which serves as a container for nodes and links. It provides a number of methods (such as `addHost()`, `addSwitch()`, and `addLink()`) for adding nodes and links to a network, as well as network configuration, startup and shutdown (notably `start()` and `stop()`.)
- High-level API: The high-level API adds a topology template abstraction, the `Topo` class, which provides the ability to create reusable, parametrized topology templates. These templates can be passed to the `mn` command (via the `--custom` option) and used from the command line.

Source: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#examples+

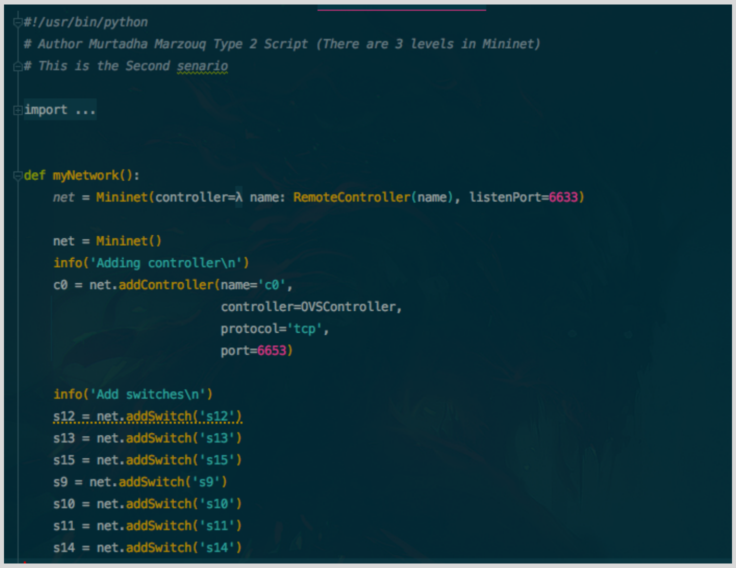


```
mininet> dump
<Host h10: h10-eth0:10.0.2.1 pid=8787>
<Host h3: h3-eth0:10.0.1.12 pid=8789>
<Host h2: h2-eth0:10.0.1.11 pid=8791>
<Host h9: h9-eth0:10.0.1.1 pid=8793>
<Host h4: h4-eth0:10.0.1.13 pid=8795>
<Host h7: h7-eth0:10.0.2.12 pid=8797>
<Host h8: h8-eth0:10.0.2.13 pid=8799>
<Host h6: h6-eth0:10.0.2.11 pid=8801>
<Host h1: h1-eth0:10.0.1.10 pid=8803>
<Host h5: h5-eth0:10.0.2.10 pid=8805>
<OVSSwitch s12: lo:127.0.0.1,s12-eth1:None,s12-eth2:None,s12-eth3:None pid=8764>
<OVSSwitch s13: lo:127.0.0.1,s13-eth1:None,s13-eth2:None,s13-eth3:None pid=8767>
<OVSSwitch s15: lo:127.0.0.1,s15-eth1:None,s15-eth2:None,s15-eth3:None pid=8770>
<OVSSwitch s9: lo:127.0.0.1,s9-eth1:None,s9-eth2:None,s9-eth3:None,s9-eth4:None pid=8773>
<OVSSwitch s10: lo:127.0.0.1,s10-eth1:None,s10-eth2:None,s10-eth3:None pid=8776>
<OVSSwitch s11: lo:127.0.0.1,s11-eth1:None,s11-eth2:None,s11-eth3:None pid=8779>
<OVSSwitch s14: lo:127.0.0.1,s14-eth1:None,s14-eth2:None,s14-eth3:None pid=8782>
<OVSController c0: 127.0.0.1:6653 pid=8756>
```



```python
#!/usr/bin/python
# Author Murtadha Marzouq Type 3 Script (There are 3 levels in Mininet)
# This is the Second senario
from mininet.log import info
from mininet.topo import Topo

class MyTopo( Topo ):

    def build( self ):
        "Murtadha's Awesome Typo"
        s12 = self.addSwitch('s12')
        s13 = self.addSwitch('s13')
        s15 = self.addSwitch('s15')
        s9 = self.addSwitch('s9')
        s10 = self.addSwitch('s10')
        s11 = self.addSwitch('s11')
        s14 = self.addSwitch('s14')

        info('Add hosts\n')
        h10 = self.addHost('h10', cls=Host, ip='10.0.2.1/24', defaultRoute='h10-eth0')
        h3 = self.addHost('h3', cls=Host, ip='10.0.1.12', defaultRoute='via 10.0.1.1')
        h2 = self.addHost('h2', cls=Host, ip='10.0.1.11/24', defaultRoute='via 10.0.1.1')
        h9 = self.addHost('h9', cls=Host, ip='10.0.1.1/24', defaultRoute='via h9-eth0')
        h4 = self.addHost('h4', cls=Host, ip='10.0.1.13/24', defaultRoute='via 10.0.1.1')
        h7 = self.addHost('h7', cls=Host, ip='10.0.2.12/24', defaultRoute='via 10.0.2.1')
        h8 = self.addHost('h8', cls=Host, ip='10.0.2.13/24', defaultRoute='via 10.0.2.1')
        h6 = self.addHost('h6', cls=Host, ip='10.0.2.11/24', defaultRoute='via 10.0.2.1')
        h1 = self.addHost('h1', cls=Host, ip='10.0.1.10/24', defaultRoute='via 10.0.1.1')
        h5 = self.addHost('h5', cls=Host, ip='10.0.2.10/24', defaultRoute='via 10.0.2.1')
```

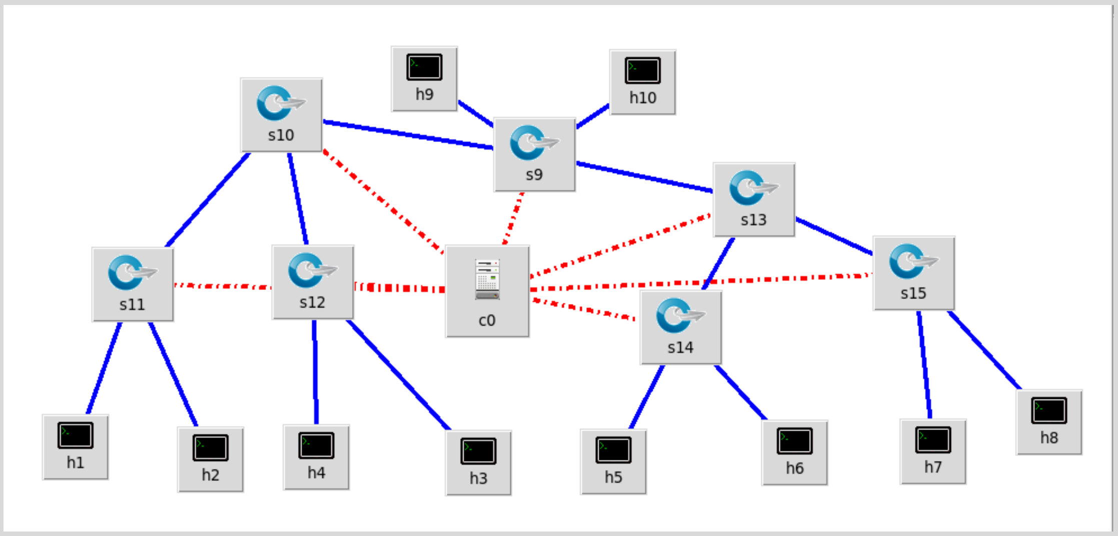
Snippet on TYPE 3 API I wrote



```python
#!/usr/bin/python
# Author Murtadha Marzouq Type 2 Script (There are 3 levels in Mininet)
# This is the Second senario

import ...


def myNetwork():
    net = Mininet(controller=λ name: RemoteController(name), listenPort=6633)

    net = Mininet()
    info('Adding controller\n')
    c0 = net.addController(name='c0',
                           controller=OVSController,
                           protocol='tcp',
                           port=6653)

    info('Add switches\n')
    s12 = net.addSwitch('s12')
    s13 = net.addSwitch('s13')
    s15 = net.addSwitch('s15')
    s9 = net.addSwitch('s9')
    s10 = net.addSwitch('s10')
    s11 = net.addSwitch('s11')
    s14 = net.addSwitch('s14')
```

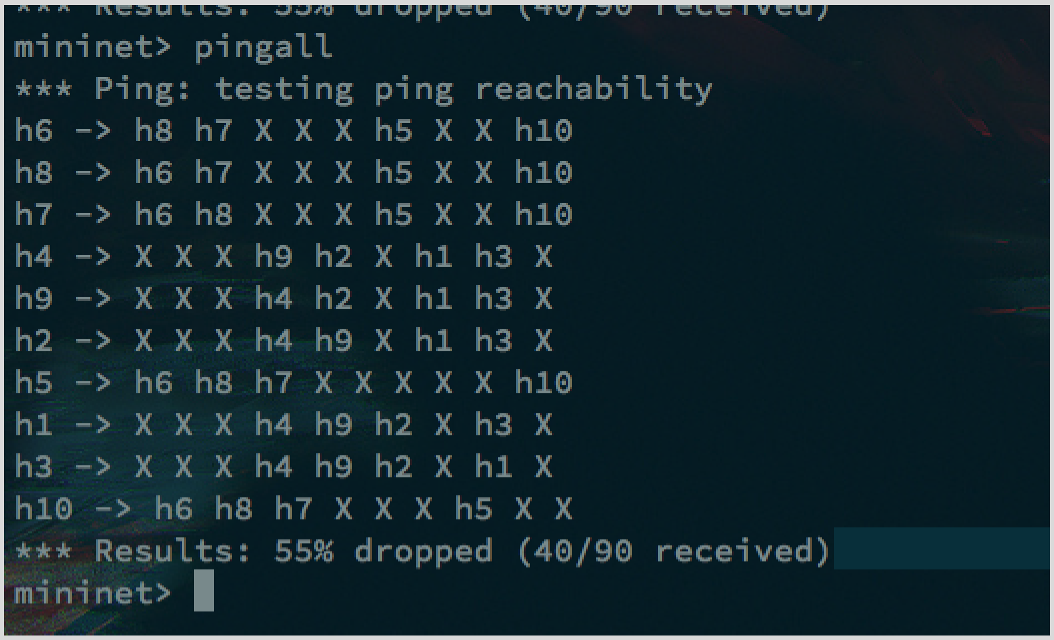
Snippet on TYPE 2 API I wrote

2. Run the topology script in the provided VM as shown below



Connectivity before enabling routing (See Bonus)

3. **Ping all commands in the Mininet terminal and check the connectivity (5 points)If you see a packet drop, please identify the reason for packet drop.**



```
*** Results: 55% dropped (40/90 received)
mininet> pingall
*** Ping: testing ping reachability
h6 -> h8 h7 X X X h5 X X h10
h8 -> h6 h7 X X X h5 X X h10
h7 -> h6 h8 X X X h5 X X h10
h4 -> X X X h9 h2 X h1 h3 X
h9 -> X X X h4 h2 X h1 h3 X
h2 -> X X X h4 h9 X h1 h3 X
h5 -> h6 h8 h7 X X X X X h10
h1 -> X X X h4 h9 h2 X h3 X
h3 -> X X X h4 h9 h2 X h1 X
h10 -> h6 h8 h7 X X X h5 X X
*** Results: 55% dropped (40/90 received)
mininet> 
```

Connectivity: The two subnets are not routed to each other with a routing table so they are unable to communicate with each other. According to documentation from Mininet, default

controllers make switches act as layer 2 switches by default so no IP routing, just MAC addresses. Luckily, there is a way around this ===> **Refer to Bonus Section at the bottom.**

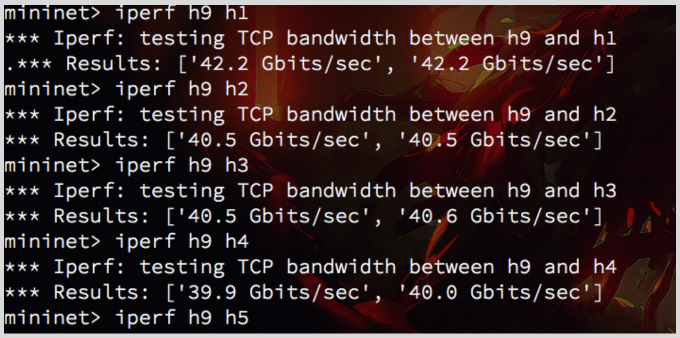
.4.You must think about the interconnection between each node and create a tabular format (30 points)below to analyze the network topology

The result reflects when all hosts are not routed together (2 subnets are unreachable) --See bonus section for screenshots otherwise, each host would only be able to measure bandwidth to its respective subnet :

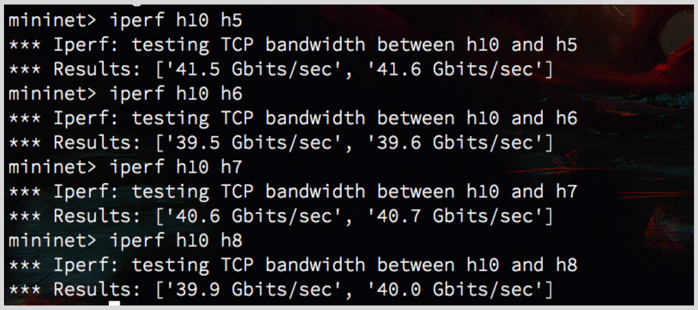
Host	IP	Is this host pingable from H9	Is this host pingable from H10	Scenario: 1 Measure bw and delay using iperf on H9	Scenario: 2 Measure bw and nd delay using iperf on H10
H1	10.0.1.10/24	True	False	42.2 Gbits/sec', '42.2 Gbits/sec PING 10.0.1.10 (10.0.1.10) 56(84) bytes of data. 64 bytes from 10.0.1.10: icmp_seq=1 ttl=64 time=10.7 ms	Not the same subnet
H2	10.0.1.11/24	True	False	40.5 Gbits/sec', '40.5 Gbits/sec mininet> h9 ping h2 PING 10.0.1.11 (10.0.1.11) 56(84) bytes of data. 64 bytes from 10.0.1.11: icmp_seq=1 ttl=64 time=3.82 ms	Not the same subnet
H3	10.0.1.12/24	True	False	40.5 Gbits/sec', '40.6 Gbits/sec' mininet> h9 ping h3 PING 10.0.1.12 (10.0.1.12) 56(84) bytes of data. 64 bytes from 10.0.1.12: icmp_seq=1 ttl=64 time=3.70 ms	Not the same subnet

H4	10.0.1.13/24	True	False	39.9 Gbits/sec', '40.0 Gbits/sec mininet> h9 ping h4 PING 10.0.1.13 (10.0.1.13) 56(84) bytes of data. 64 bytes from 10.0.1.13: icmp_seq=1 ttl=64 time=3.59 ms	Not the same subnet
H5	10.0.2.10/24	False	True	Not the same subnet	41.5 Gbits/sec', '41.6 Gbits/sec mininet> h10 ping h5 PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data. 64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=6.28 ms
H6	10.0.2.11/24	False	True	Not the same subnet	39.5 Gbits/sec', '39.6 Gbits/sec mininet> h10 ping h6 PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data. 64 bytes from 10.0.2.11: icmp_seq=1 ttl=64 time=4.79 ms
H7	10.0.2.12/24	False	True	Not the same subnet	40.6 Gbits/sec', '40.7 Gbits/sec mininet> h10 ping h7 PING 10.0.2.12 (10.0.2.12) 56(84) bytes of data. 64 bytes from 10.0.2.12: icmp_seq=1 ttl=64 time=17.4 ms
H8	10.0.2.13/24	False	True	Not the same subnet	39.9 Gbits/sec', '40.0 Gbits/sec mininet> h10 ping h8 PING 10.0.2.13

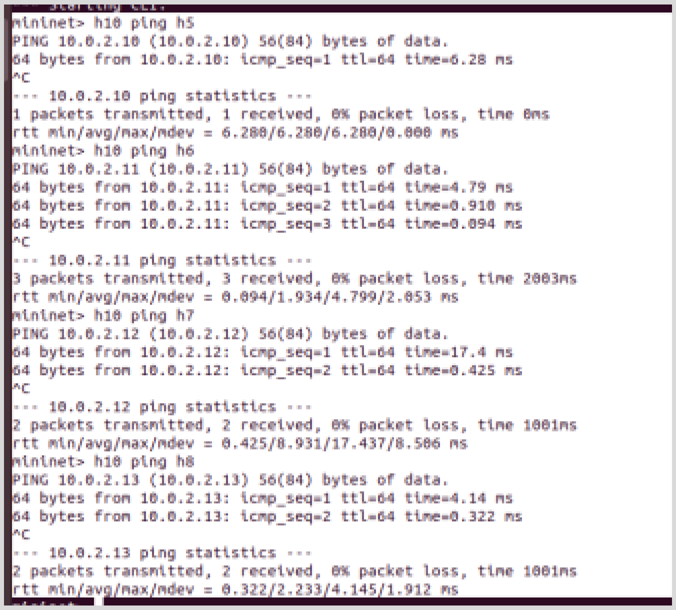
					(10.0.2.13) 56(84) bytes of data. 64 bytes from 10.0.2.13: icmp_seq=1 ttl=64 time=4.14 ms
H9	10.0.1.1	True	False		
H10	10.0.2.1	False	True		



```
mininet> iperf h9 h1
*** Iperf: testing TCP bandwidth between h9 and h1
.*** Results: ['42.2 Gbits/sec', '42.2 Gbits/sec']
mininet> iperf h9 h2
*** Iperf: testing TCP bandwidth between h9 and h2
*** Results: ['40.5 Gbits/sec', '40.5 Gbits/sec']
mininet> iperf h9 h3
*** Iperf: testing TCP bandwidth between h9 and h3
*** Results: ['40.5 Gbits/sec', '40.6 Gbits/sec']
mininet> iperf h9 h4
*** Iperf: testing TCP bandwidth between h9 and h4
*** Results: ['39.9 Gbits/sec', '40.0 Gbits/sec']
mininet> iperf h9 h5
```



```
mininet> iperf h10 h5
*** Iperf: testing TCP bandwidth between h10 and h5
*** Results: ['41.5 Gbits/sec', '41.6 Gbits/sec']
mininet> iperf h10 h6
*** Iperf: testing TCP bandwidth between h10 and h6
*** Results: ['39.5 Gbits/sec', '39.6 Gbits/sec']
mininet> iperf h10 h7
*** Iperf: testing TCP bandwidth between h10 and h7
*** Results: ['40.6 Gbits/sec', '40.7 Gbits/sec']
mininet> iperf h10 h8
*** Iperf: testing TCP bandwidth between h10 and h8
*** Results: ['39.9 Gbits/sec', '40.0 Gbits/sec']
```



```
mininet> h10 ping h5
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=6.28 ms
^C
--- 10.0.2.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 6.280/6.280/6.280/0.000 ms
mininet> h10 ping h6
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data.
64 bytes from 10.0.2.11: icmp_seq=1 ttl=64 time=4.79 ms
64 bytes from 10.0.2.11: icmp_seq=2 ttl=64 time=0.910 ms
64 bytes from 10.0.2.11: icmp_seq=3 ttl=64 time=0.094 ms
^C
--- 10.0.2.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.094/1.934/4.799/2.053 ms
mininet> h10 ping h7
PING 10.0.2.12 (10.0.2.12) 56(84) bytes of data.
64 bytes from 10.0.2.12: icmp_seq=1 ttl=64 time=17.4 ms
64 bytes from 10.0.2.12: icmp_seq=2 ttl=64 time=0.425 ms
^C
--- 10.0.2.12 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.425/8.931/17.437/8.506 ms
mininet> h10 ping h8
PING 10.0.2.13 (10.0.2.13) 56(84) bytes of data.
64 bytes from 10.0.2.13: icmp_seq=1 ttl=64 time=4.14 ms
64 bytes from 10.0.2.13: icmp_seq=2 ttl=64 time=0.322 ms
^C
--- 10.0.2.13 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.322/2.233/4.145/1.912 ms
```

5. In Short, explain why some nodes are not reachable from h9 and h10 in the given topology.

There are many reasons why the ICMB requests were not being sent so I will touch over 3 equally important concepts about networking to lo analyze the situation:

A. I will begin with the most relevant: I have no clue what Mininet is doing. I understand abstraction and the good things it enables, however it changes years

of perception. For example, according to Mininet’s documentation, static switches are not capable of routing because they operate on the mac address level (Layer 2 vs Layer 3(. This is true until you realize that each network node with Mininet is a mini VM)

B. If we assume that it is operating on Layer 2, then the switches need the hosts to have MAC addresses to do some basic routing (beware of flooding)

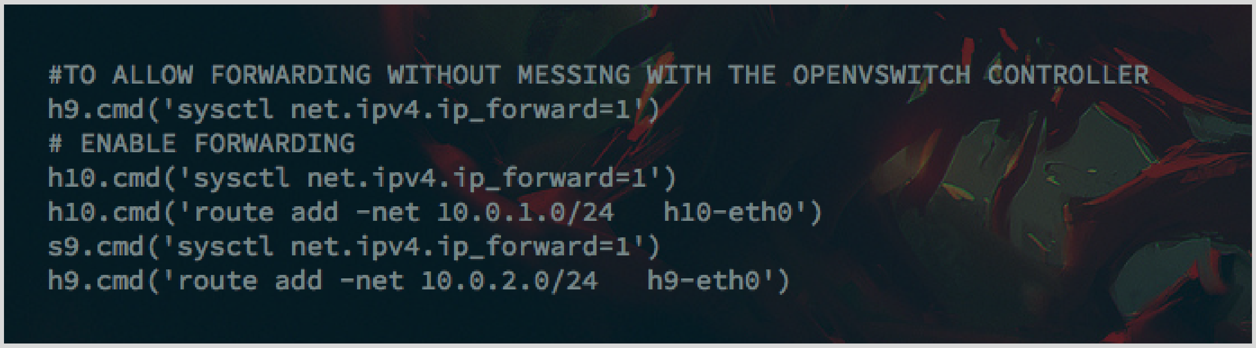
C. another equally likely scenario is because Host 9 and Host 8 have the same network ID, that is a different diagram, but on a completely separate **VLAN**. I assumed that H9 and H10 are on the same network of their designated subnet.

D. I will touch slightly on this because it is beyond the scope of this assignment. This network if introduces loops that you might need a controller that can handle **Spanning Tree Protocol** in perhaps **Ryu Controller**

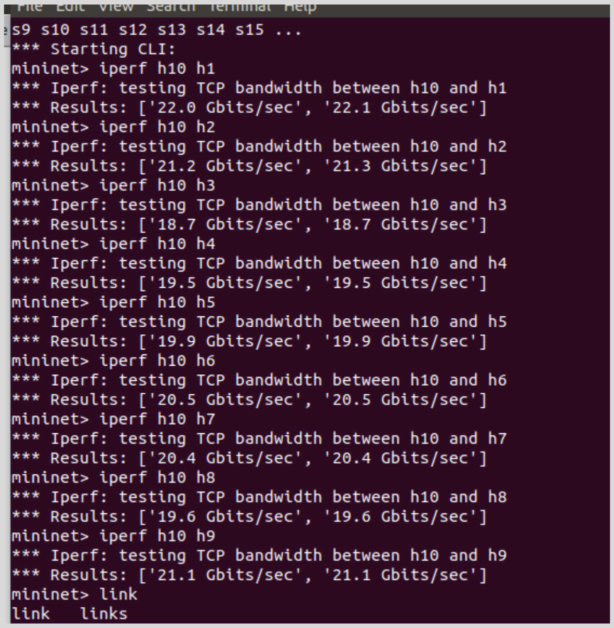
BONUS:

I was able to fix the routing between the two networks by adding a routing rule to the switches routing table.

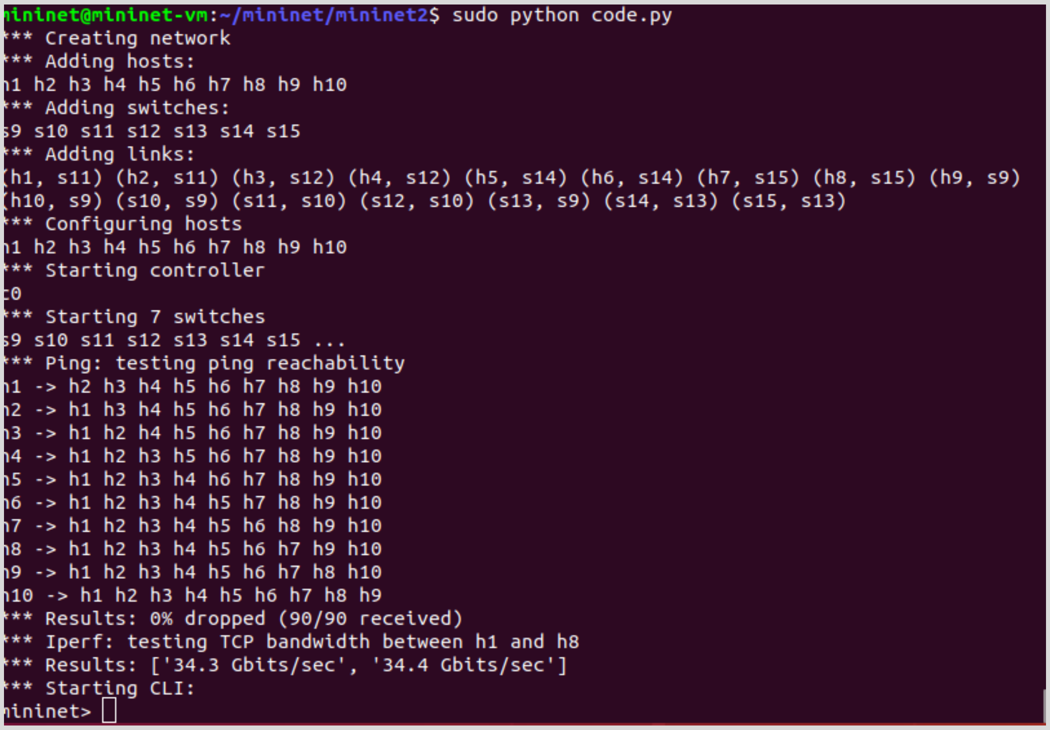
It works now, but it refused to work for hours. It turns out that Mininet by default makes switches Layer 2 switches that can not route between subnets because Layer 2 see only Mac Addresses, not IP's, so I went to each switch and manually enabled IP forwarding:



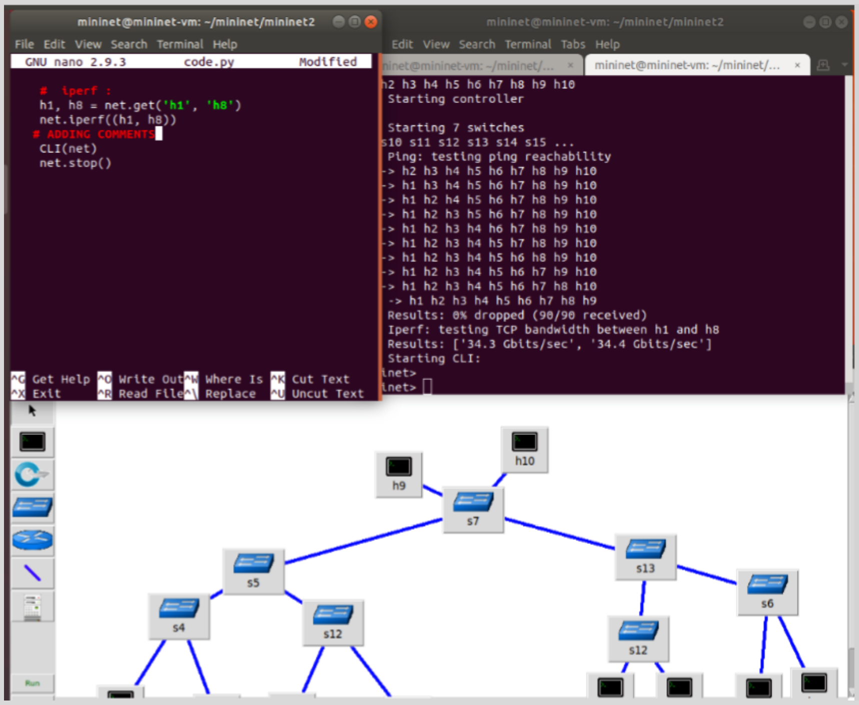
```
#TO ALLOW FORWARDING WITHOUT MESSING WITH THE OPENVSWITCH CONTROLLER
h9.cmd('sysctl net.ipv4.ip_forward=1')
# ENABLE FORWARDING
h10.cmd('sysctl net.ipv4.ip_forward=1')
h10.cmd('route add -net 10.0.1.0/24   h10-eth0')
s9.cmd('sysctl net.ipv4.ip_forward=1')
h9.cmd('route add -net 10.0.2.0/24   h9-eth0')
```



```
s9 s10 s11 s12 s13 s14 s15 ...
*** Starting CLI:
mininet> iperf h10 h1
*** Iperf: testing TCP bandwidth between h10 and h1
*** Results: ['22.0 Gbits/sec', '22.1 Gbits/sec']
mininet> iperf h10 h2
*** Iperf: testing TCP bandwidth between h10 and h2
*** Results: ['21.2 Gbits/sec', '21.3 Gbits/sec']
mininet> iperf h10 h3
*** Iperf: testing TCP bandwidth between h10 and h3
*** Results: ['18.7 Gbits/sec', '18.7 Gbits/sec']
mininet> iperf h10 h4
*** Iperf: testing TCP bandwidth between h10 and h4
*** Results: ['19.5 Gbits/sec', '19.5 Gbits/sec']
mininet> iperf h10 h5
*** Iperf: testing TCP bandwidth between h10 and h5
*** Results: ['19.9 Gbits/sec', '19.9 Gbits/sec']
mininet> iperf h10 h6
*** Iperf: testing TCP bandwidth between h10 and h6
*** Results: ['20.5 Gbits/sec', '20.5 Gbits/sec']
mininet> iperf h10 h7
*** Iperf: testing TCP bandwidth between h10 and h7
*** Results: ['20.4 Gbits/sec', '20.4 Gbits/sec']
mininet> iperf h10 h8
*** Iperf: testing TCP bandwidth between h10 and h8
*** Results: ['19.6 Gbits/sec', '19.6 Gbits/sec']
mininet> iperf h10 h9
*** Iperf: testing TCP bandwidth between h10 and h9
*** Results: ['21.1 Gbits/sec', '21.1 Gbits/sec']
mininet> link
link    links
```



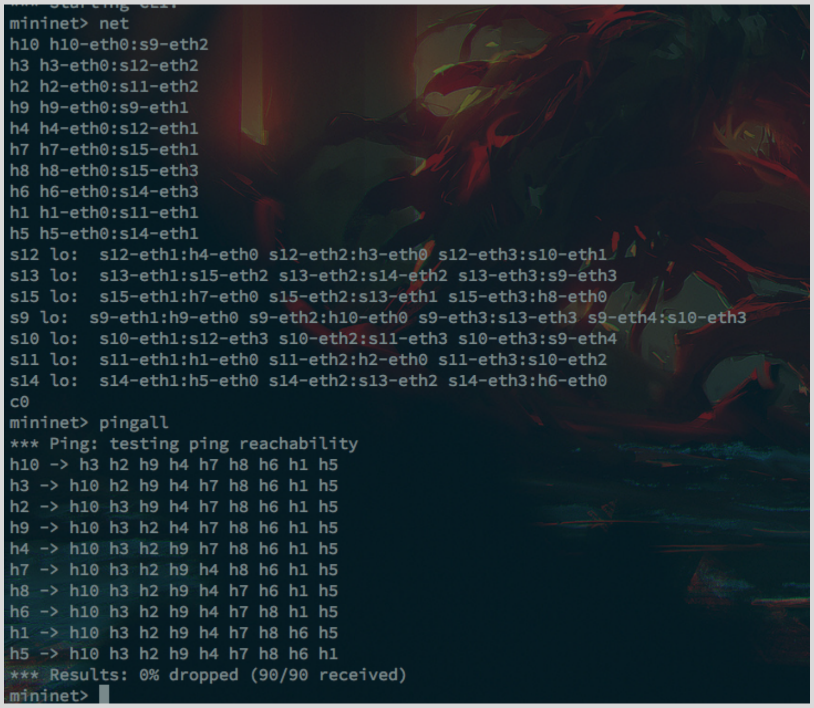
```
ininet@mininet-vm:~/mininet/mininet2$ sudo python code.py
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s9 s10 s11 s12 s13 s14 s15
*** Adding links:
(h1, s11) (h2, s11) (h3, s12) (h4, s12) (h5, s14) (h6, s14) (h7, s15) (h8, s15) (h9, s9)
(h10, s9) (s10, s9) (s11, s10) (s12, s10) (s13, s9) (s14, s13) (s15, s13)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c0
*** Starting 7 switches
s9 s10 s11 s12 s13 s14 s15 ...
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['34.3 Gbits/sec', '34.4 Gbits/sec']
*** Starting CLI:
ininet>
```



```
GNU nano 2.9.3            code.py            Modified

   #  iperf :
   h1, h8 = net.get('h1', 'h8')
   net.iperf((h1, h8))
  # ADDING COMMENTS
   CLI(net)
   net.stop()

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text
```

```
h2 h3 h4 h5 h6 h7 h8 h9 h10
 Starting controller

 Starting 7 switches
s10 s11 s12 s13 s14 s15 ...
 Ping: testing ping reachability
-> h2 h3 h4 h5 h6 h7 h8 h9 h10
-> h1 h3 h4 h5 h6 h7 h8 h9 h10
-> h1 h2 h4 h5 h6 h7 h8 h9 h10
-> h1 h2 h3 h5 h6 h7 h8 h9 h10
-> h1 h2 h3 h4 h6 h7 h8 h9 h10
-> h1 h2 h3 h4 h5 h7 h8 h9 h10
-> h1 h2 h3 h4 h5 h6 h8 h9 h10
-> h1 h2 h3 h4 h5 h6 h7 h9 h10
-> h1 h2 h3 h4 h5 h6 h7 h8 h10
 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
 Results: 0% dropped (90/90 received)
 Iperf: testing TCP bandwidth between h1 and h8
 Results: ['34.3 Gbits/sec', '34.4 Gbits/sec']
 Starting CLI:
inet>
inet>
```

mininet@mininet-vm: ~/mininet/mininet2
File Edit View Search Terminal Help
Edit View Search Terminal Tabs Help
h9
h10
s7
s5
s4
s12
s13
s6
s12
Run



```
*** Starting CLI:
mininet> net
h10 h10-eth0:s9-eth2
h3 h3-eth0:s12-eth2
h2 h2-eth0:s11-eth2
h9 h9-eth0:s9-eth1
h4 h4-eth0:s12-eth1
h7 h7-eth0:s15-eth1
h8 h8-eth0:s15-eth3
h6 h6-eth0:s14-eth3
h1 h1-eth0:s11-eth1
h5 h5-eth0:s14-eth1
s12 lo:  s12-eth1:h4-eth0 s12-eth2:h3-eth0 s12-eth3:s10-eth1
s13 lo:  s13-eth1:s15-eth2 s13-eth2:s14-eth2 s13-eth3:s9-eth3
s15 lo:  s15-eth1:h7-eth0 s15-eth2:s13-eth1 s15-eth3:h8-eth0
s9 lo:  s9-eth1:h9-eth0 s9-eth2:h10-eth0 s9-eth3:s13-eth3 s9-eth4:s10-eth3
s10 lo:  s10-eth1:s12-eth3 s10-eth2:s11-eth3 s10-eth3:s9-eth4
s11 lo:  s11-eth1:h1-eth0 s11-eth2:h2-eth0 s11-eth3:s10-eth2
s14 lo:  s14-eth1:h5-eth0 s14-eth2:s13-eth2 s14-eth3:h6-eth0
c0
mininet> pingall
*** Ping: testing ping reachability
h10 -> h3 h2 h9 h4 h7 h8 h6 h1 h5
h3 -> h10 h2 h9 h4 h7 h8 h6 h1 h5
h2 -> h10 h3 h9 h4 h7 h8 h6 h1 h5
h9 -> h10 h3 h2 h4 h7 h8 h6 h1 h5
h4 -> h10 h3 h2 h9 h7 h8 h6 h1 h5
h7 -> h10 h3 h2 h9 h4 h8 h6 h1 h5
h8 -> h10 h3 h2 h9 h4 h7 h6 h1 h5
h6 -> h10 h3 h2 h9 h4 h7 h8 h1 h5
h1 -> h10 h3 h2 h9 h4 h7 h8 h6 h5
h5 -> h10 h3 h2 h9 h4 h7 h8 h6 h1
*** Results: 0% dropped (90/90 received)
mininet> 
```