Part1

1.

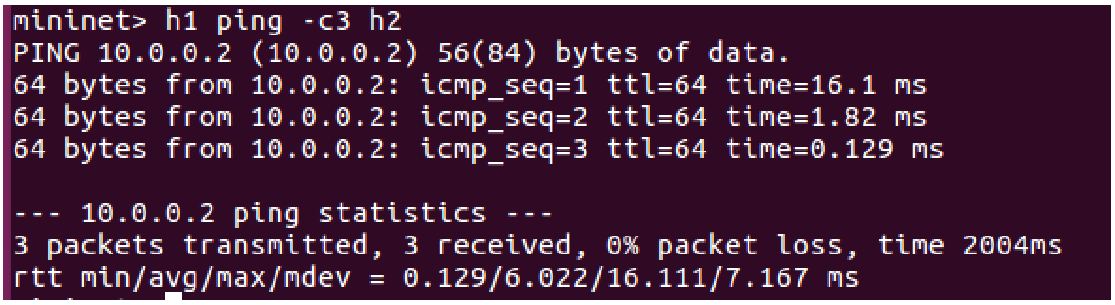
org.onosproject.optical-model

org.onosproject.hostprovider

org.onosproject.openflow-base

org.onosproject.lldpprovider

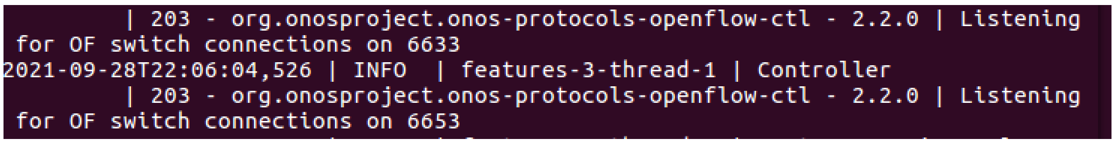
2.



```
mininet> h1 ping -c3 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=16.1 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.82 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.129 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.129/6.022/16.111/7.167 ms
```

只要 activate org.onosproject.fwd 後，就可以順利 ping(如上圖所示)，否則 data plane 上沒有 flow

3.



```
        | 203 - org.onosproject.onos-protocols-openflow-ctl - 2.2.0 | Listening
 for OF switch connections on 6633
2021-09-28T22:06:04,526 | INFO  | features-3-thread-1 | Controller
        | 203 - org.onosproject.onos-protocols-openflow-ctl - 2.2.0 | Listening
 for OF switch connections on 6653
```

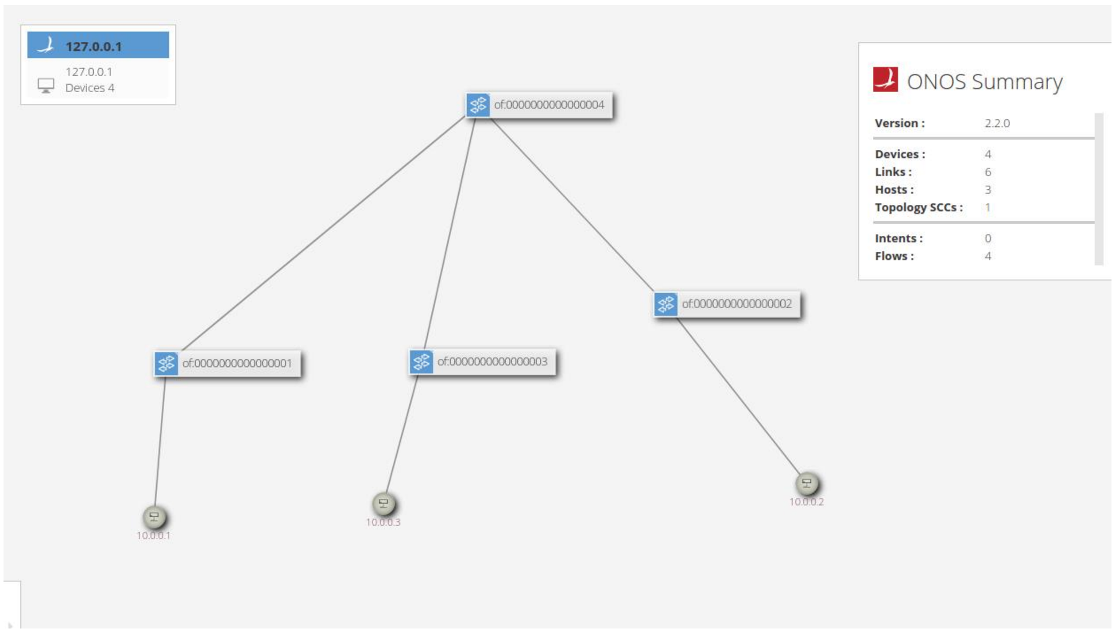
Activate org.onosproject.openflow 後，觀察 termanal 資訊，可發現上圖兩行，因此可知控制的 port 為 6653 或 6633

4.

org.onosproject.openflow-base

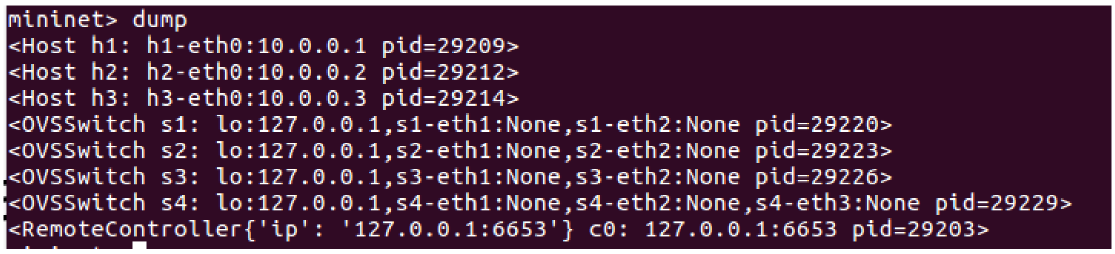
⇨ 雖然 org.onosproject.openflow 也可以看到 port 6653 的資訊，但仔細對幾次，就會發現 org.onosproject.openflow-base 才是主要決定 port 6653 出現與否的關鍵。

Part 2

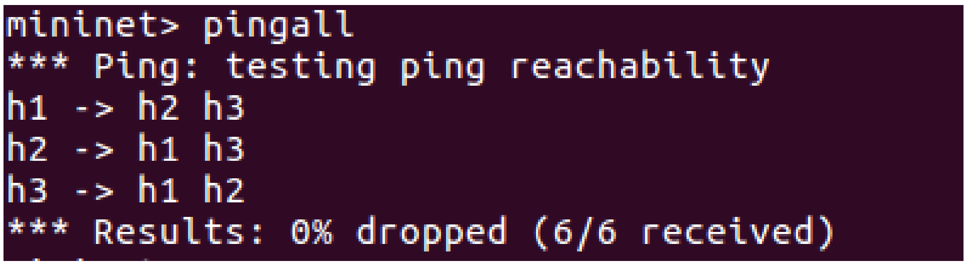


ONOS Summary

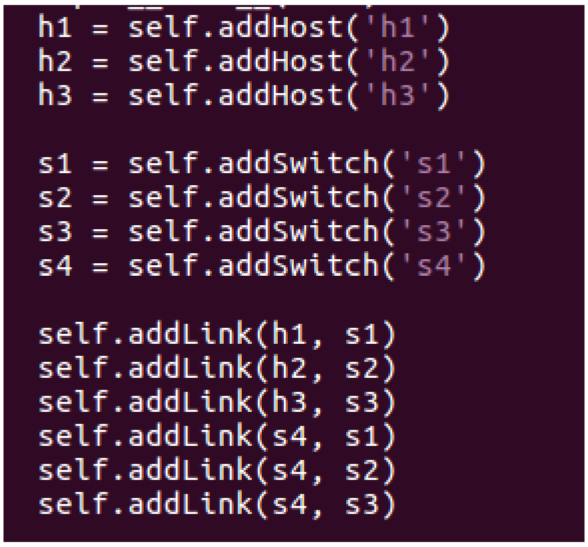
Version :	2.2.0
Devices :	4
Links :	6
Hosts :	3
Topology SCCs :	1
Intents :	0
Flows :	4



```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=29209>
<Host h2: h2-eth0:10.0.0.2 pid=29212>
<Host h3: h3-eth0:10.0.0.3 pid=29214>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=29220>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None pid=29223>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=29226>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=29229>
<RemoteController{'ip': '127.0.0.1:6653'} c0: 127.0.0.1:6653 pid=29203>
```



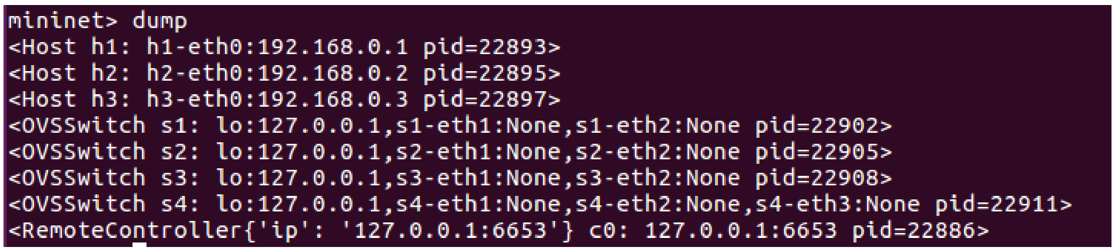
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```



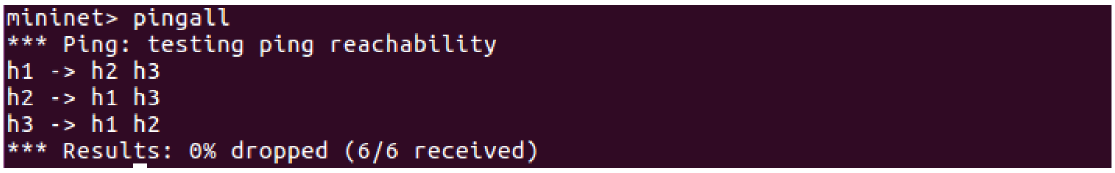
```
  h1 = self.addHost('h1')
  h2 = self.addHost('h2')
  h3 = self.addHost('h3')

  s1 = self.addSwitch('s1')
  s2 = self.addSwitch('s2')
  s3 = self.addSwitch('s3')
  s4 = self.addSwitch('s4')

  self.addLink(h1, s1)
  self.addLink(h2, s2)
  self.addLink(h3, s3)
  self.addLink(s4, s1)
  self.addLink(s4, s2)
  self.addLink(s4, s3)
```

⇨ 在原先的 script 上，多加幾個 host 和 switch，然後按照題目給的 topology 給定 link，並在 GUI 上面顯示出如題目所要的 topology

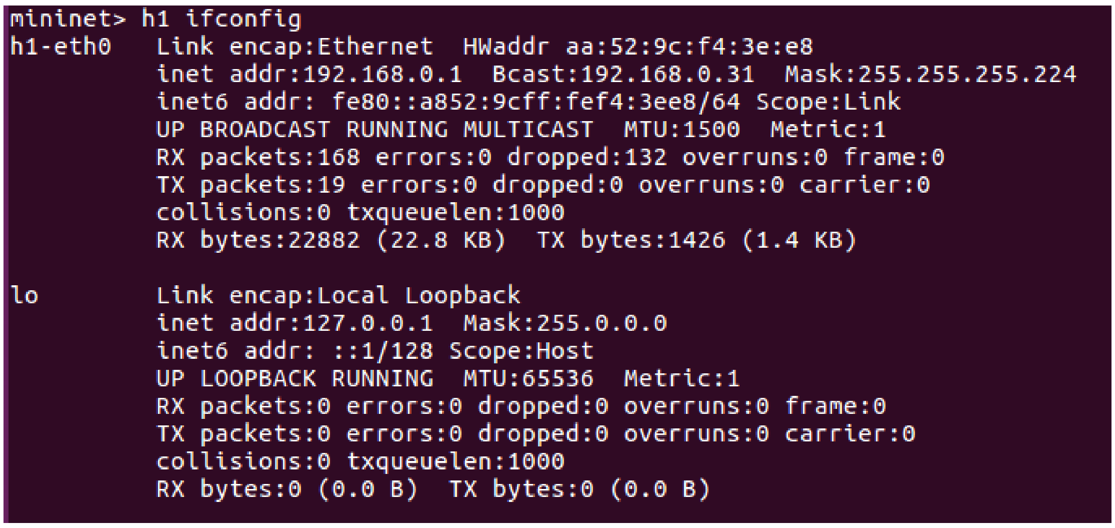
Part 3



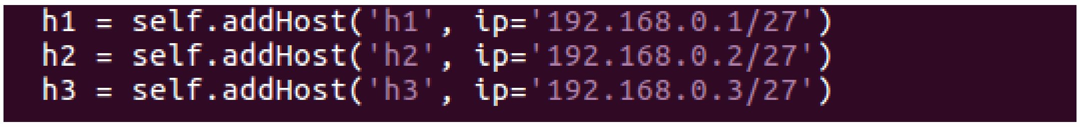
```
mininet> dump
<Host h1: h1-eth0:192.168.0.1 pid=22893>
<Host h2: h2-eth0:192.168.0.2 pid=22895>
<Host h3: h3-eth0:192.168.0.3 pid=22897>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=22902>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None pid=22905>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=22908>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=22911>
<RemoteController{'ip': '127.0.0.1:6653'} c0: 127.0.0.1:6653 pid=22886>
```



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```



```
mininet> h1 ifconfig
h1-eth0   Link encap:Ethernet  HWaddr aa:52:9c:f4:3e:e8
          inet addr:192.168.0.1  Bcast:192.168.0.31  Mask:255.255.255.224
          inet6 addr: fe80::a852:9cff:fef4:3ee8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:168 errors:0 dropped:132 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22882 (22.8 KB)  TX bytes:1426 (1.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```



```
  h1 = self.addHost('h1', ip='192.168.0.1/27')
  h2 = self.addHost('h2', ip='192.168.0.2/27')
  h3 = self.addHost('h3', ip='192.168.0.3/27')
```

⇨Part2 的部分，再加上處理 ip 和 mask 的參數，mask 的部分，由於 224 = 128+64+32，所以等於前面有 24+3 = 27 個 1，因此使用 ‘/27’

Part 4

1. Onos 和 mininet 的基本操作
2. 藉由觀察連線中的 information，來判斷使用的 port 或是其他更多有用的資訊
3. 虛擬機的進階操作，和 linux 的進階語法