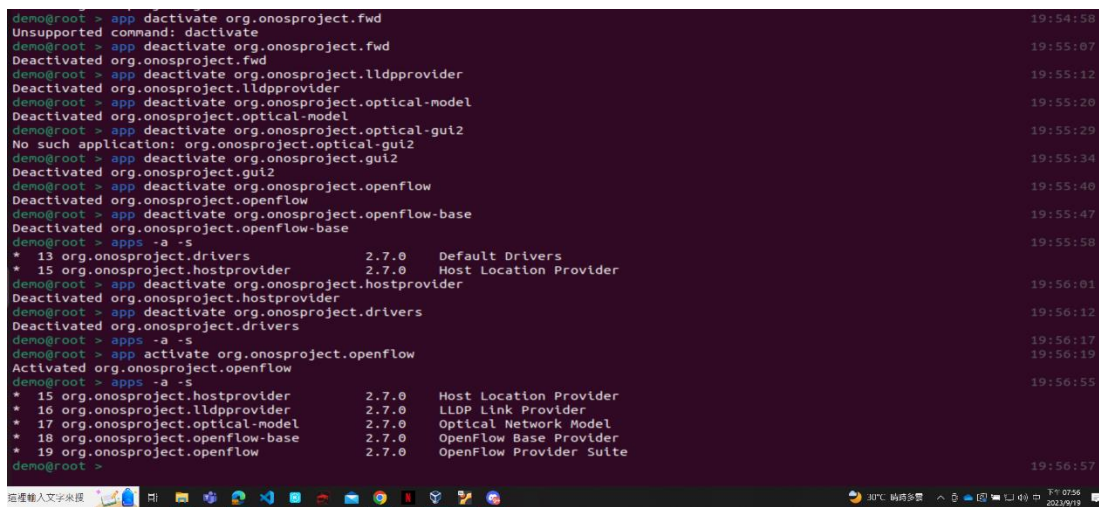


Part1:

Q1: When ONOS activates “org.onosproject.openflow,” what are the APPs which it also activates?

Ans: 如圖一所示，我先將所有 app deactivate，再 activate org.onosproject.openflow,後，以下這四個 app 會隨著 org.onosproject.openflow, 一起 activate:

1. org.onosproject.hostprovider
2. org.onosproject.lldpprovider
3. org.onosproject.optical-model
4. org.onosproject.openflow-base

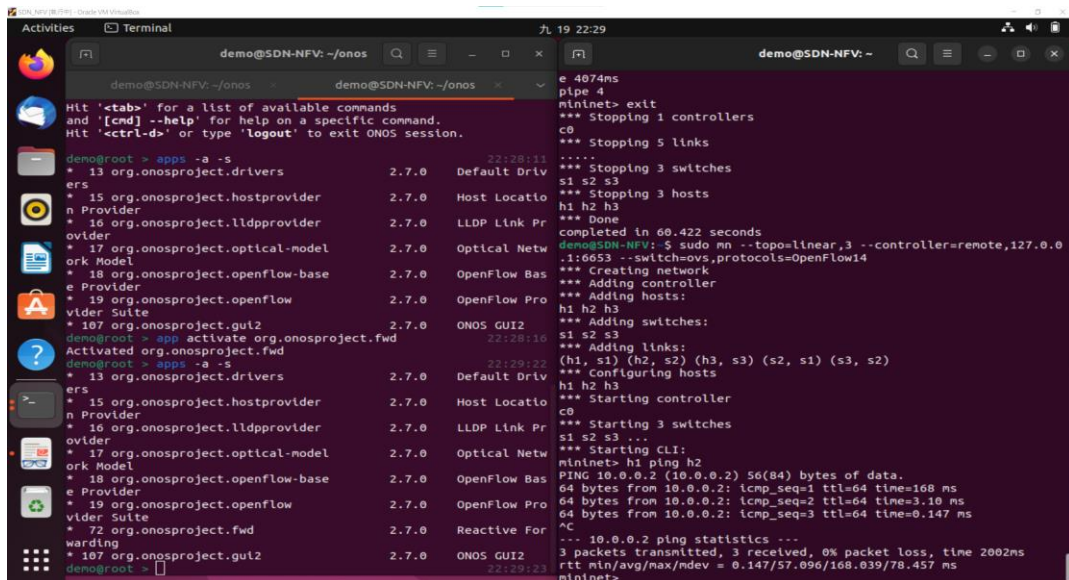
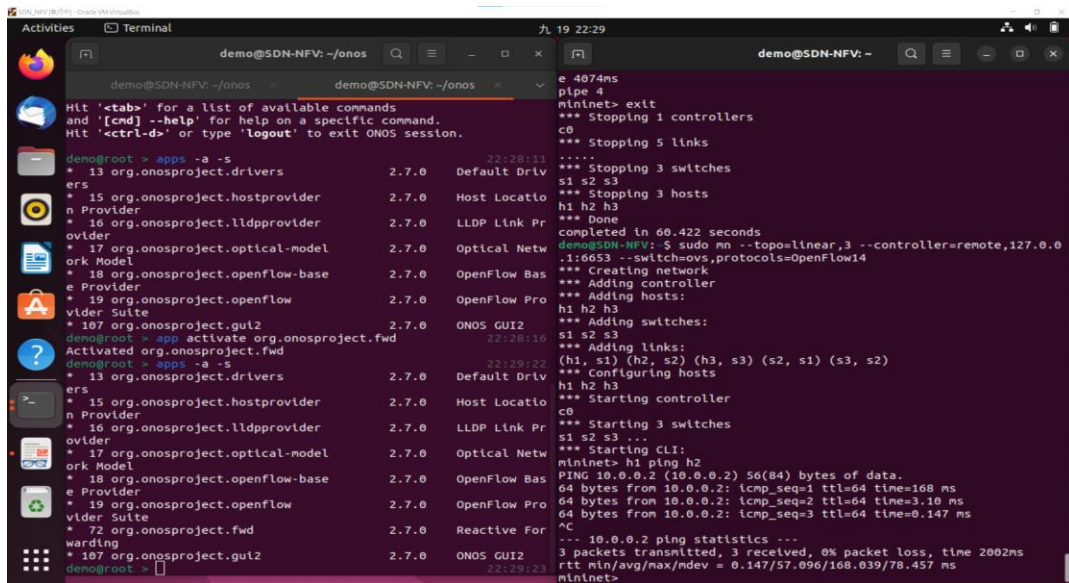


```
deno@root > app deactivate org.onosproject.fwd 19:54:58
Unsupported command: deactivate
deno@root > app deactivate org.onosproject.fwd 19:55:07
Deactivated org.onosproject.fwd
deno@root > app deactivate org.onosproject.lldpprovider 19:55:12
Deactivated org.onosproject.lldpprovider
deno@root > app deactivate org.onosproject.optical-model 19:55:20
Deactivated org.onosproject.optical-model
deno@root > app deactivate org.onosproject.optical-gui2 19:55:29
No such application: org.onosproject.optical-gui2
deno@root > app deactivate org.onosproject.gui2 19:55:34
Deactivated org.onosproject.gui2
deno@root > app deactivate org.onosproject.openflow 19:55:40
Deactivated org.onosproject.openflow
deno@root > app deactivate org.onosproject.openflow-base 19:55:47
Deactivated org.onosproject.openflow-base
deno@root > apps -a -s 19:55:58
* 13 org.onosproject.drivers 2.7.0 Default Drivers
* 15 org.onosproject.hostprovider 2.7.0 Host Location Provider
deno@root > app deactivate org.onosproject.hostprovider 19:56:01
Deactivated org.onosproject.hostprovider
deno@root > app deactivate org.onosproject.drivers 19:56:12
Deactivated org.onosproject.drivers
deno@root > apps -a -s 19:56:17
deno@root > app activate org.onosproject.openflow 19:56:19
Activated org.onosproject.openflow
deno@root > apps -a -s 19:56:53
* 15 org.onosproject.hostprovider 2.7.0 Host Location Provider
* 16 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider
* 17 org.onosproject.optical-model 2.7.0 Optical Network Model
* 18 org.onosproject.openflow-base 2.7.0 OpenFlow Base Provider
* 19 org.onosproject.openflow 2.7.0 OpenFlow Provider Suite
deno@root > 19:56:57
```

圖一

Q2: After activating ONOS and running the commands on P.17 and P.20. Will H1 ping H2 successfully? Why or why not?

Ans: 如圖二所示，在沒有 activate org.onosproject.fwd 前，h1 ping 不到 h2，原因是 there are no flows installed on the data-plane，which forward the traffic appropriately，在圖三 activate org.onosproject.fwd 後，h1 就可以 ping 到 h2 了



Q3: Which TCP port the controller listens for the OpenFlow connection request from the switch? Screenshot

Ans: 如圖四所示，Controller 會 listen 6653 port 或 6633 port 並請求建立 OpenFlow connection，6653 和 6633 都是 OpenFlow port，較新的 Openflow version 使用 6653，較舊的 version 則使用 6633

```

demo@SDN-NFV: /ano$ netstat -nlpt
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.0:53:53        0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:5005         0.0.0.0:*               LISTEN      2523/java
tcp6       0      0 :::6653                :::*                     LISTEN      2523/java
tcp6       0      0 :::6633                :::*                     LISTEN      2523/java
tcp6       0      0 :::39635               :::*                     LISTEN      2523/java
tcp6       0      0 :::135061              :::*                     LISTEN      2143/bazel(onos)
tcp6       0      0 :::8181                :::*                     LISTEN      2523/java
tcp6       0      0 :::8181                :::*                     LISTEN      2523/java
tcp6       0      0 :::22                  :::*                     LISTEN      -
tcp6       0      0 :::1:631               :::*                     LISTEN      -
tcp6       0      0 :::1099                :::*                     LISTEN      2523/java
tcp6       0      0 :::9876                :::*                     LISTEN      2523/java
tcp6       0      0 127.0.0.1:42511       :::*                     LISTEN      2523/java
demo@SDN-NFV: /ano$

```

圖四

Q4: In question 3, which APP enables the controller to listen on the TCP port?

Ans: openflow-base enable controller to listen on the TCP port，openflow-base 是 openflow 的 dependency app 之一，如 Q1 列出的 app 都是 openflow 的 dependency app，openflow-base 控制 controller listen TCP port，其餘無關。

Part2

如圖五，我按照題目設定 5 個 host 跟 5 個 switch，並按照題目的拓樸在他們之間建立 Link，建出來的 network 如圖六，pingall 成功，圖七是拓樸圖，如題目所要求的一樣。

```

1  from mininet.topo import Topo
2
3  class Project1_Topo_312581034( Topo ):
4      def __init__( self ):
5          Topo.__init__( self )
6
7          h1 = self.addHost( 'h1' )
8          h2 = self.addHost( 'h2' )
9          h3 = self.addHost( 'h3' )
10         h4 = self.addHost( 'h4' )
11         h5 = self.addHost( 'h5' )
12
13         s1 = self.addSwitch( 's1' )
14         s2 = self.addSwitch( 's2' )
15         s3 = self.addSwitch( 's3' )
16         s4 = self.addSwitch( 's4' )
17         s5 = self.addSwitch( 's5' )
18
19
20         self.addLink( h1, s1 )
21         self.addLink( h2, s2 )
22         self.addLink( h3, s3 )
23         self.addLink( h4, s4 )
24         self.addLink( h5, s5 )
25
26         self.addLink( s2, s1 )
27         self.addLink( s2, s3 )
28         self.addLink( s2, s4 )
29         self.addLink( s2, s5 )
30
31
32
33  topos = { 'topo_part2_312581034': Project1_Topo_312581034 }
34

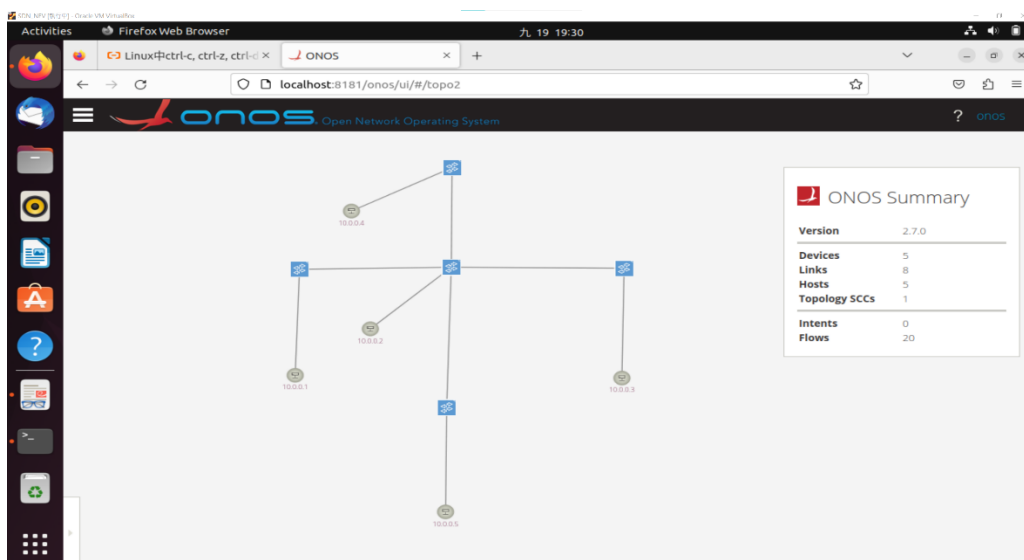
```

圖五

```
demo@SDN-NFV: ~/Desktop
demo@SDN-NFV: ~/onos
demo@SDN-NFV: ~/Desktop

ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([_[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
kill -9 -f mininet:
*** Shutting down stale tunnels
kill -9 -f Tunnel=Ethernet
kill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
demo@SDN-NFV: ~/Desktop$ sudo mn --custom=project1_part2_312581034.py --topo=topo_part2_312581034 --controller=remote,ip=127.0.0.1,po
rt=6653 --switch=ovs,protocols=OpenFlow14
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s2, s3) (s2, s4) (s2, s5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting S switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet> S
```

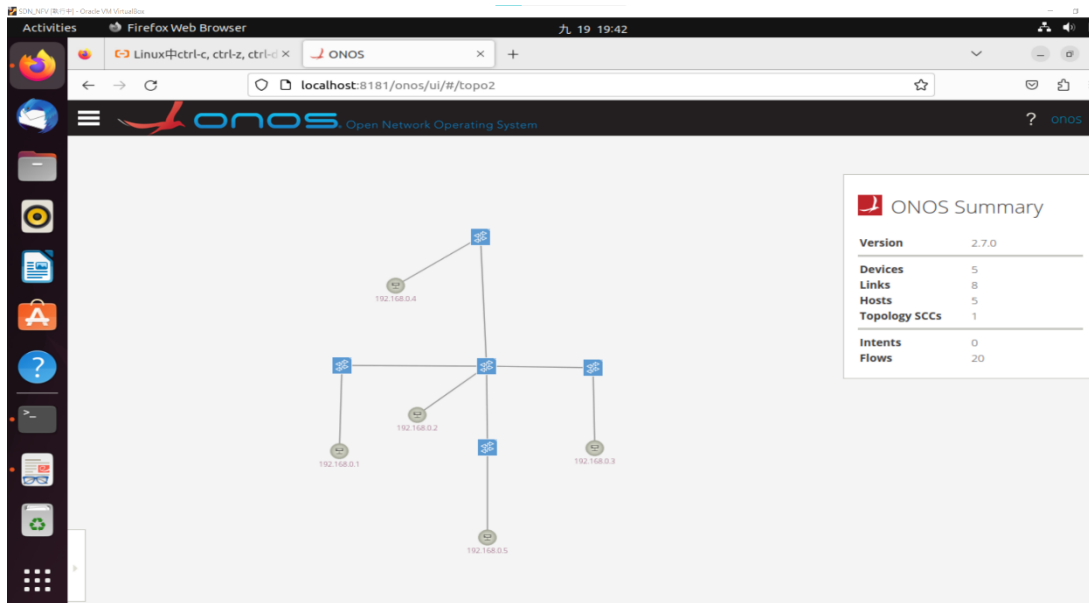
圖六



圖七

Part3

如圖八，程式碼與 part2 基本一樣，差別在於為每個 host 設定各自的 static ip，建立起的網路與拓樸圖如圖九與圖十，pingall 成功，拓樸圖上 host 顯示的是自己設定的 static ip，圖十一 ~ 圖十六是 dump 與各個 host ifconfig 的截圖，可以看到 ip 都有設定成功



圖十

```
mininet> dump
<Host h1: h1-eth0:192.168.0.1 pld=7321>
<Host h2: h2-eth0:192.168.0.2 pld=7321>
<Host h3: h3-eth0:192.168.0.3 pld=7325>
<Host h4: h4-eth0:192.168.0.4 pld=7327>
<Host h5: h5-eth0:192.168.0.5 pld=7329>
<OVSSwitch('protocols': 'OpenFlow14') s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pld=7334>
<OVSSwitch('protocols': 'OpenFlow14') s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,s2-eth5:None pld=7337>
<OVSSwitch('protocols': 'OpenFlow14') s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pld=7340>
<OVSSwitch('protocols': 'OpenFlow14') s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None pld=7343>
<OVSSwitch('protocols': 'OpenFlow14') s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None pld=7346>
<RemoteController('ip': '127.0.0.1', 'port': 6653) c0: 127.0.0.1:6653 pld=7315>
mininet>
```

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.1 netmask 255.255.255.224 broadcast 192.168.0.31
    inet6 fe80::6894:20ff:fe7c:75c prefixlen 64 scopeid 0x20<link>
    ether 6a:94:20:7c:07:5c txqueuelen 1000 (Ethernet)
    RX packets 270 bytes 34125 (34.1 KB)
    RX errors 0 dropped 198 overruns 0 frame 0
    TX packets 28 bytes 2056 (2.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet> s
```

```
mininet> h2 ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.2 netmask 255.255.255.224 broadcast 192.168.0.31
    inet6 fe80::5461:35ff:fe76:c82f prefixlen 64 scopeid 0x20<link>
    ether 56:61:35:76:c8:2f txqueuelen 1000 (Ethernet)
    RX packets 282 bytes 35793 (35.7 KB)
    RX errors 0 dropped 210 overruns 0 frame 0
    TX packets 28 bytes 2056 (2.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet> s
```

```
mininet> h3 ifconfig
h3-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.3 netmask 255.255.255.224 broadcast 192.168.0.31
    inet6 fe80::1442:afff:fe00:60a prefixlen 64 scopeid 0x20<link>
    ether 10:42:af:06:00:0a txqueuelen 1000 (Ethernet)
    RX packets 282 bytes 36233 (36.2 KB)
    RX errors 0 dropped 218 overruns 0 frame 0
    TX packets 28 bytes 2056 (2.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet>
```

```
mininet> h3 ifconfig
h3-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.3 netmask 255.255.255.224 broadcast 192.168.0.31
    inet6 fe80::1442:afff:fe00:60a prefixlen 64 scopeid 0x20<link>
    ether 10:42:af:06:00:0a txqueuelen 1000 (Ethernet)
    RX packets 282 bytes 36233 (36.2 KB)
    RX errors 0 dropped 218 overruns 0 frame 0
    TX packets 28 bytes 2056 (2.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet>
```

```
mininet> h4 ifconfig
h4-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.4 netmask 255.255.255.224 broadcast 192.168.0.31
    inet6 fe80::b876:71ff:fe75:ccd0 prefixlen 64 scopeid 0x20<link>
    ether ba:76:71:75:cc:d0 txqueuelen 1000 (Ethernet)
    RX packets 292 bytes 37623 (37.6 KB)
    RX errors 0 dropped 228 overruns 0 frame 0
    TX packets 28 bytes 2056 (2.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet>
```

圖十一 ~ 圖十六

What I learned or solved:

有學到關於 SDN 的技術，如何建立一個網路，以及查看他的拓樸圖，和如何客製自己的網路拓樸。