Part1

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

   - Optical Network Model: org.onosproject.optical-model
   - OpenFlow Base Provider: org.onosproject.openflow-base
   - LLDP Link Provider: org.onosproject.lldpprovider
   - Host Location Provider: org.onosproject.hostprovider

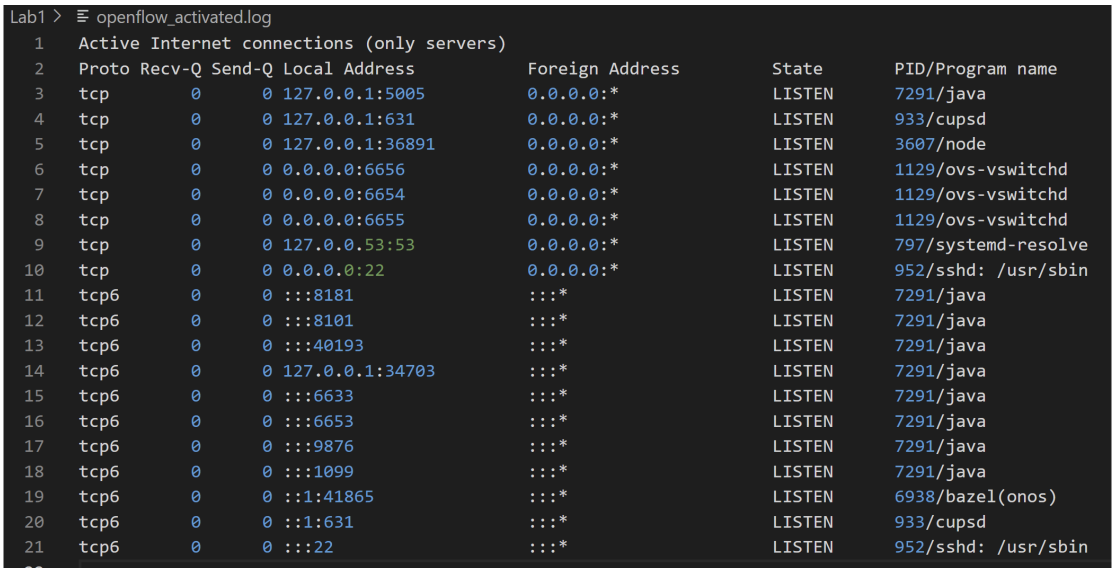
2. After activate ONOS and run P.14 command. Will H1 ping H2 successfully? Why or why not?

   No, because there are no flows inserted on the data-plane, so the data-plane doesn’t know how to forward the packet. After activating the Reactive Forwarding app, it will insert forwarding flows, and h1 can ping h2.

3. Which TCP port the controller listens for the OpenFlow connection request from the switch? Screenshot

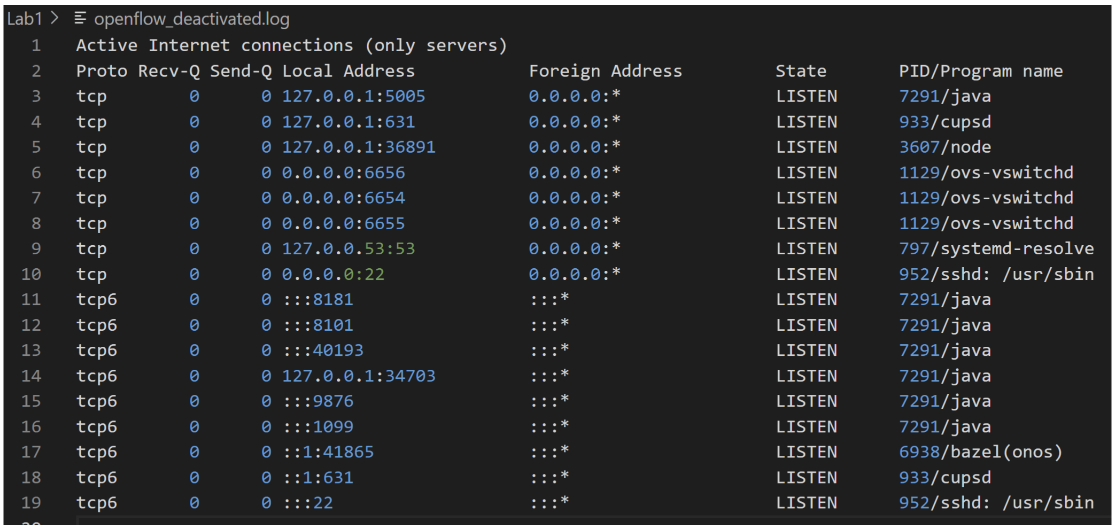
   6633, 6653

   OpenFlow Activate



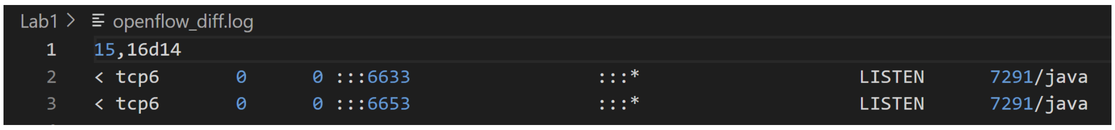
```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:5005          0.0.0.0:*               LISTEN      7291/java
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      933/cupsd
tcp        0      0 127.0.0.1:36891         0.0.0.0:*               LISTEN      3607/node
tcp        0      0 0.0.0.0:6656            0.0.0.0:*               LISTEN      1129/ovs-vswitchd
tcp        0      0 0.0.0.0:6654            0.0.0.0:*               LISTEN      1129/ovs-vswitchd
tcp        0      0 0.0.0.0:6655            0.0.0.0:*               LISTEN      1129/ovs-vswitchd
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      797/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      952/sshd: /usr/sbin
tcp6       0      0 :::8181                 :::*                    LISTEN      7291/java
tcp6       0      0 :::8101                 :::*                    LISTEN      7291/java
tcp6       0      0 :::40193                :::*                    LISTEN      7291/java
tcp6       0      0 127.0.0.1:34703         :::*                    LISTEN      7291/java
tcp6       0      0 :::6633                 :::*                    LISTEN      7291/java
tcp6       0      0 :::6653                 :::*                    LISTEN      7291/java
tcp6       0      0 :::9876                 :::*                    LISTEN      7291/java
tcp6       0      0 :::1099                 :::*                    LISTEN      7291/java
tcp6       0      0 ::1:41865               :::*                    LISTEN      6938/bazel(onos)
tcp6       0      0 ::1:631                 :::*                    LISTEN      933/cupsd
tcp6       0      0 :::22                   :::*                    LISTEN      952/sshd: /usr/sbin
```

OpenFlow Deactivate


Lab1 > openflow_deactivated.log

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:5005          0.0.0.0:*               LISTEN      7291/java
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      933/cupsd
tcp        0      0 127.0.0.1:36891         0.0.0.0:*               LISTEN      3607/node
tcp        0      0 0.0.0.0:6656            0.0.0.0:*               LISTEN      1129/ovs-vswitchd
tcp        0      0 0.0.0.0:6654            0.0.0.0:*               LISTEN      1129/ovs-vswitchd
tcp        0      0 0.0.0.0:6655            0.0.0.0:*               LISTEN      1129/ovs-vswitchd
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      797/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      952/sshd: /usr/sbin
tcp6       0      0 :::8181                 :::*                    LISTEN      7291/java
tcp6       0      0 :::8101                 :::*                    LISTEN      7291/java
tcp6       0      0 :::40193                :::*                    LISTEN      7291/java
tcp6       0      0 127.0.0.1:34703         :::*                    LISTEN      7291/java
tcp6       0      0 :::9876                 :::*                    LISTEN      7291/java
tcp6       0      0 :::1099                 :::*                    LISTEN      7291/java
tcp6       0      0 ::1:41865               :::*                    LISTEN      6938/bazel(onos)
tcp6       0      0 ::1:631                 :::*                    LISTEN      933/cupsd
tcp6       0      0 :::22                   :::*                    LISTEN      952/sshd: /usr/sbin
```

Diff between OpenFlow activate and deactivate

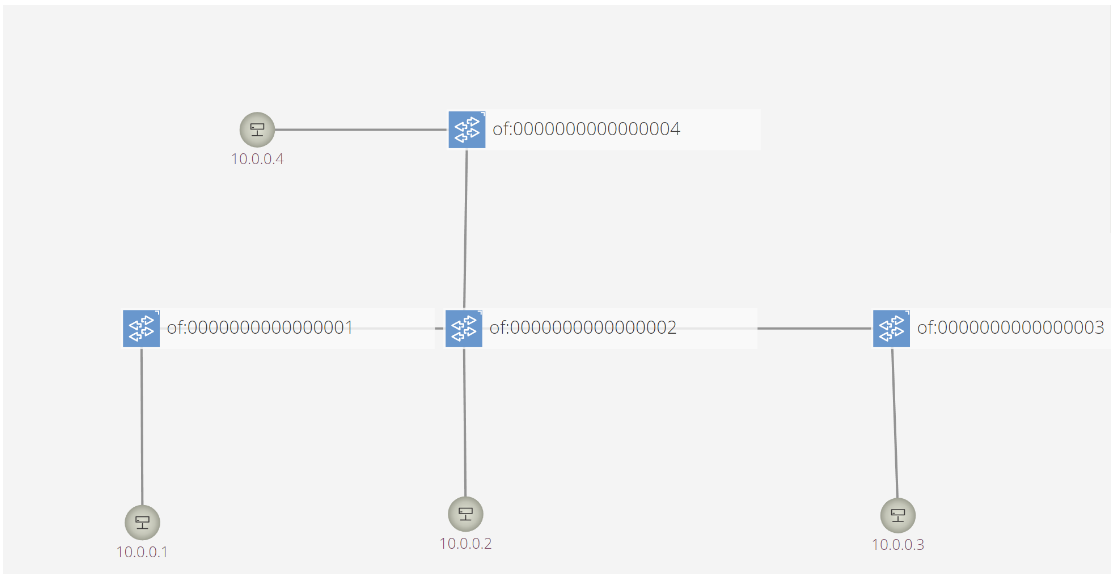

Lab1 > openflow_diff.log

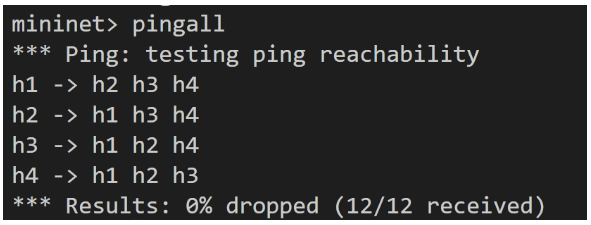
```
15,16d14
< tcp6       0      0 :::6633                 :::*                    LISTEN      7291/java
< tcp6       0      0 :::6653                 :::*                    LISTEN      7291/java
```

4. In question 3, which APP enables the controller to listen on the TCP port?

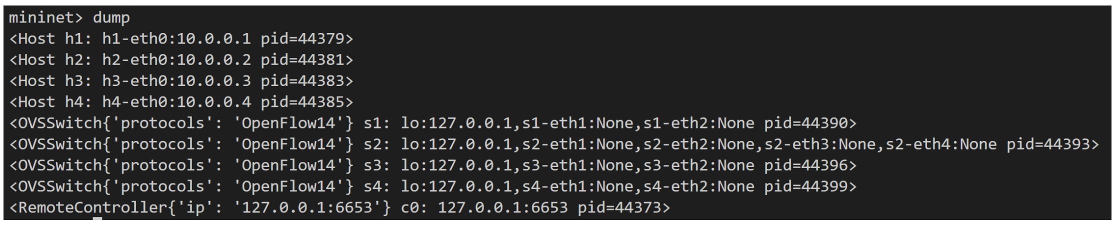
   OpenFlow Base Provider: org.onosproject.openflow-base

Part2

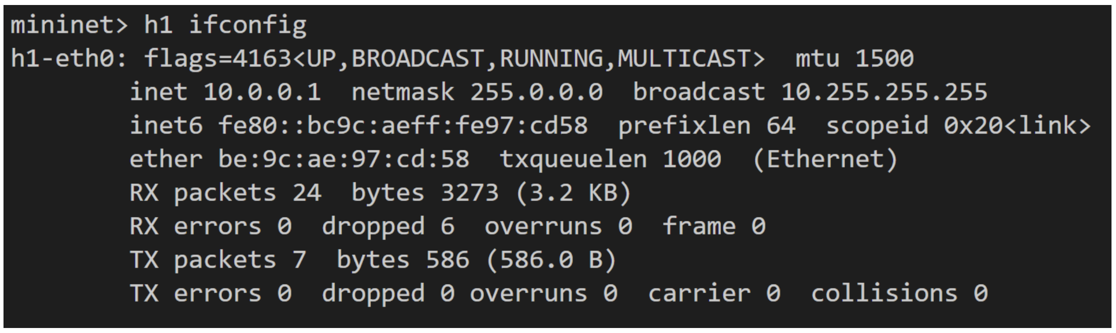




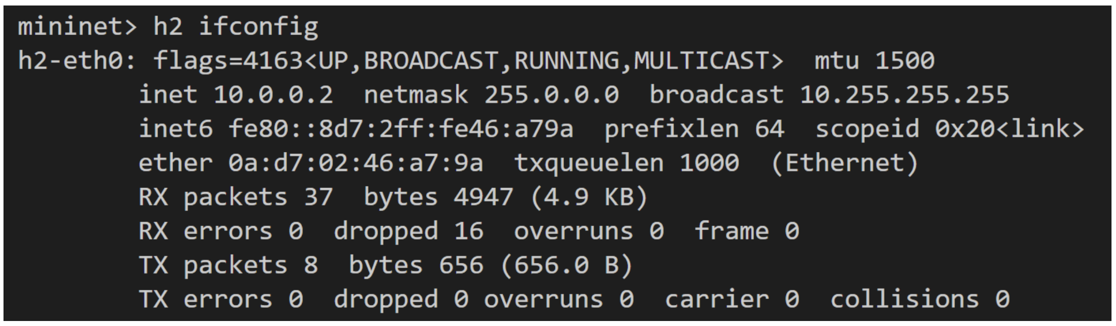
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```



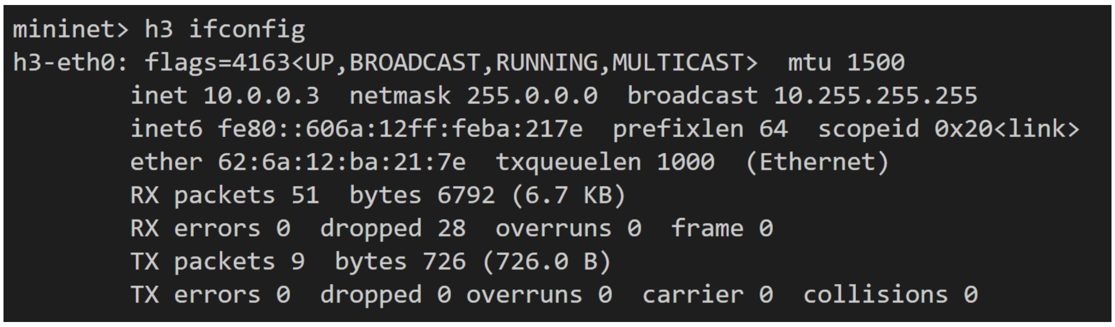
```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=44379>
<Host h2: h2-eth0:10.0.0.2 pid=44381>
<Host h3: h3-eth0:10.0.0.3 pid=44383>
<Host h4: h4-eth0:10.0.0.4 pid=44385>
<OVSSwitch{'protocols': 'OpenFlow14'} s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=44390>
<OVSSwitch{'protocols': 'OpenFlow14'} s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None pid=44393>
<OVSSwitch{'protocols': 'OpenFlow14'} s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=44396>
<OVSSwitch{'protocols': 'OpenFlow14'} s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None pid=44399>
<RemoteController{'ip': '127.0.0.1:6653'} c0: 127.0.0.1:6653 pid=44373>
```



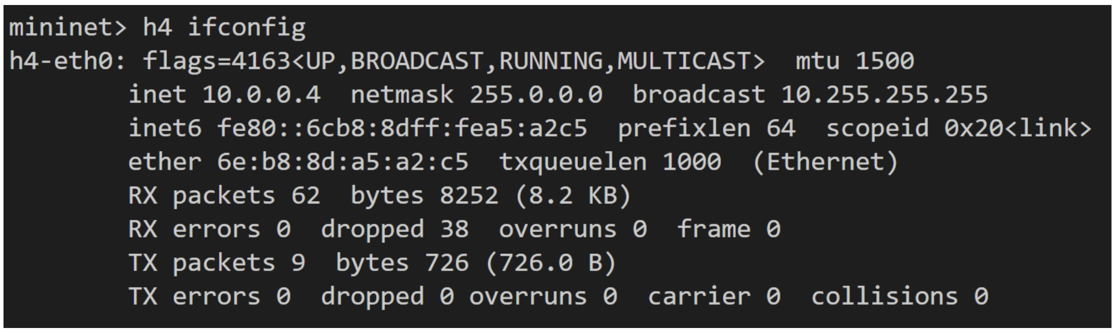
```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::bc9c:aeff:fe97:cd58  prefixlen 64  scopeid 0x20<link>
        ether be:9c:ae:97:cd:58  txqueuelen 1000  (Ethernet)
        RX packets 24  bytes 3273 (3.2 KB)
        RX errors 0  dropped 6  overruns 0  frame 0
        TX packets 7  bytes 586 (586.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
mininet> h2 ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::8d7:2ff:fe46:a79a  prefixlen 64  scopeid 0x20<link>
        ether 0a:d7:02:46:a7:9a  txqueuelen 1000  (Ethernet)
        RX packets 37  bytes 4947 (4.9 KB)
        RX errors 0  dropped 16  overruns 0  frame 0
        TX packets 8  bytes 656 (656.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
mininet> h3 ifconfig
h3-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.3  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::606a:12ff:feba:217e  prefixlen 64  scopeid 0x20<link>
        ether 62:6a:12:ba:21:7e  txqueuelen 1000  (Ethernet)
        RX packets 51  bytes 6792 (6.7 KB)
        RX errors 0  dropped 28  overruns 0  frame 0
        TX packets 9  bytes 726 (726.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

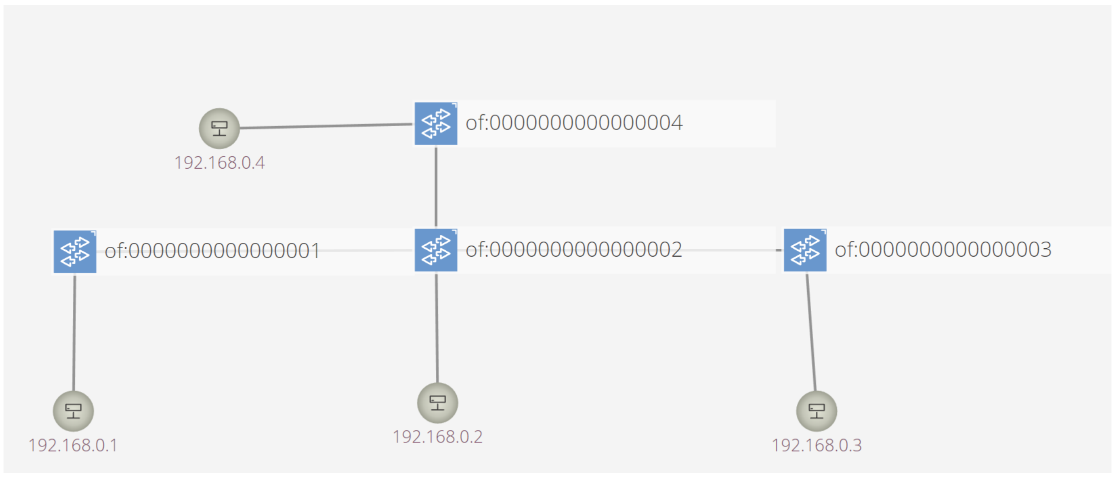


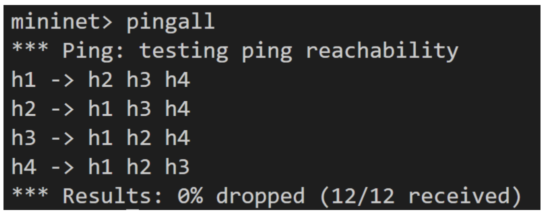
```
mininet> h4 ifconfig
h4-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.4  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::6cb8:8dff:fea5:a2c5  prefixlen 64  scopeid 0x20<link>
        ether 6e:b8:8d:a5:a2:c5  txqueuelen 1000  (Ethernet)
        RX packets 62  bytes 8252 (8.2 KB)
        RX errors 0  dropped 38  overruns 0  frame 0
        TX packets 9  bytes 726 (726.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

What I’ve done:

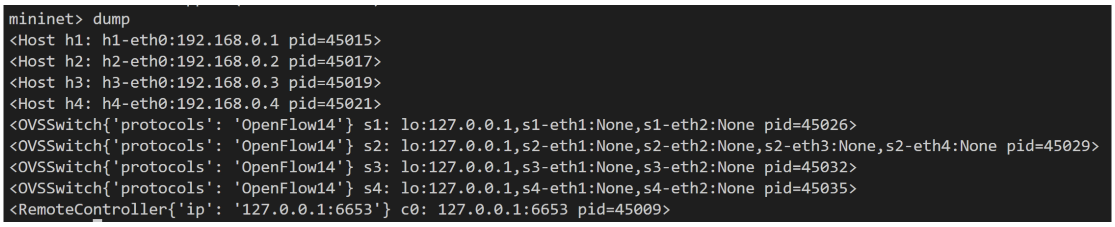
Modified sample.py, add s4 and h4, then use addLink() to build the topology.

Part3

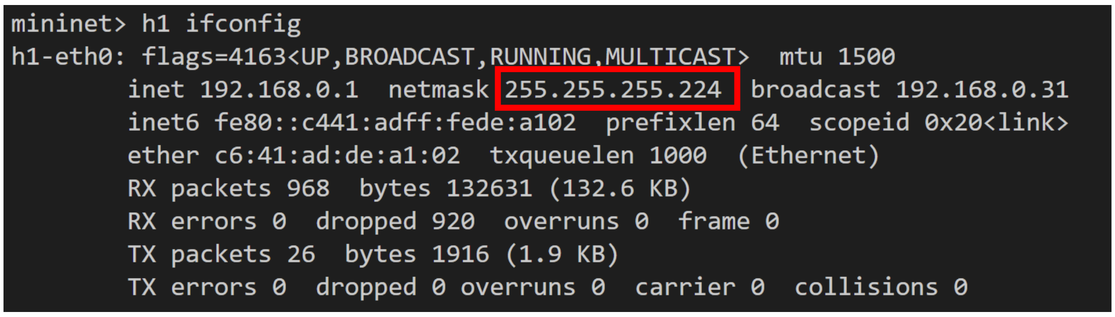




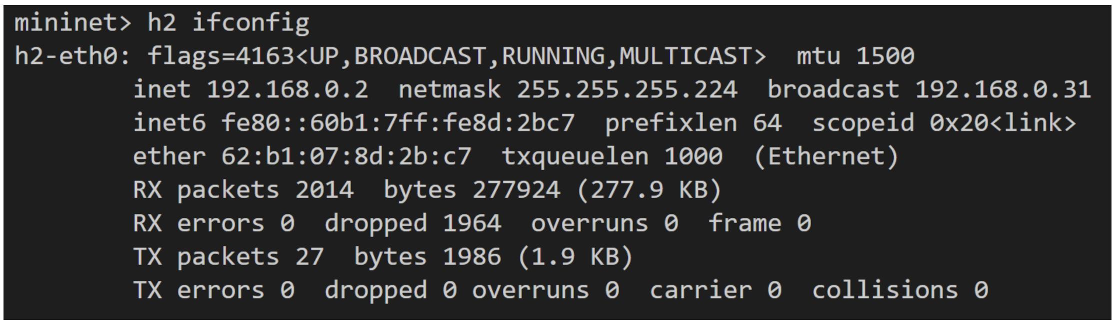
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```



```
mininet> dump
<Host h1: h1-eth0:192.168.0.1 pid=45015>
<Host h2: h2-eth0:192.168.0.2 pid=45017>
<Host h3: h3-eth0:192.168.0.3 pid=45019>
<Host h4: h4-eth0:192.168.0.4 pid=45021>
<OVSSwitch{'protocols': 'OpenFlow14'} s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=45026>
<OVSSwitch{'protocols': 'OpenFlow14'} s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None pid=45029>
<OVSSwitch{'protocols': 'OpenFlow14'} s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=45032>
<OVSSwitch{'protocols': 'OpenFlow14'} s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None pid=45035>
<RemoteController{'ip': '127.0.0.1:6653'} c0: 127.0.0.1:6653 pid=45009>
```



```
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::c441:adff:fede:a102  prefixlen 64  scopeid 0x20<link>
        ether c6:41:ad:de:a1:02  txqueuelen 1000  (Ethernet)
        RX packets 968  bytes 132631 (132.6 KB)
        RX errors 0  dropped 920  overruns 0  frame 0
        TX packets 26  bytes 1916 (1.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
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::60b1:7ff:fe8d:2bc7  prefixlen 64  scopeid 0x20<link>
        ether 62:b1:07:8d:2b:c7  txqueuelen 1000  (Ethernet)
        RX packets 2014  bytes 277924 (277.9 KB)
        RX errors 0  dropped 1964  overruns 0  frame 0
        TX packets 27  bytes 1986 (1.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
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::28a4:36ff:fe87:ea3f  prefixlen 64  scopeid 0x20<link>
        ether 2a:a4:36:87:ea:3f  txqueuelen 1000  (Ethernet)
        RX packets 992  bytes 135967 (135.9 KB)
        RX errors 0  dropped 944  overruns 0  frame 0
        TX packets 26  bytes 1916 (1.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
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::4071:f1ff:fefb:2256  prefixlen 64  scopeid 0x20<link>
        ether 42:71:f1:fb:22:56  txqueuelen 1000  (Ethernet)
        RX packets 1002  bytes 137357 (137.3 KB)
        RX errors 0  dropped 954  overruns 0  frame 0
        TX packets 26  bytes 1916 (1.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

What I've done:

Modified part2, and use “ip” argument in addHost() to assign ip and netmask to a host. (ip=”ip/mask”)

EX:

```
h1 = self.addHost( 'h1', ip="192.168.0.1/27" )
```

What I’ve learned or solved

由這次實驗我學會了如何使用 python 建出不同的 mininet 網路拓譜，也學會了 ONOS 的基礎 CLI 與 GUI 操作。過程中我有遇到一個問題，就是我原本是使用 ssh 連入虛擬機進行所有操作，但是 mininet 的 switch 一直無法連上 onos，後來我將 onos 移到虛擬機上的 terminal 執行後 mininet switch 就成功連上了，我目前還在尋找問題原因跟其他解決方法。