SDN-NFV

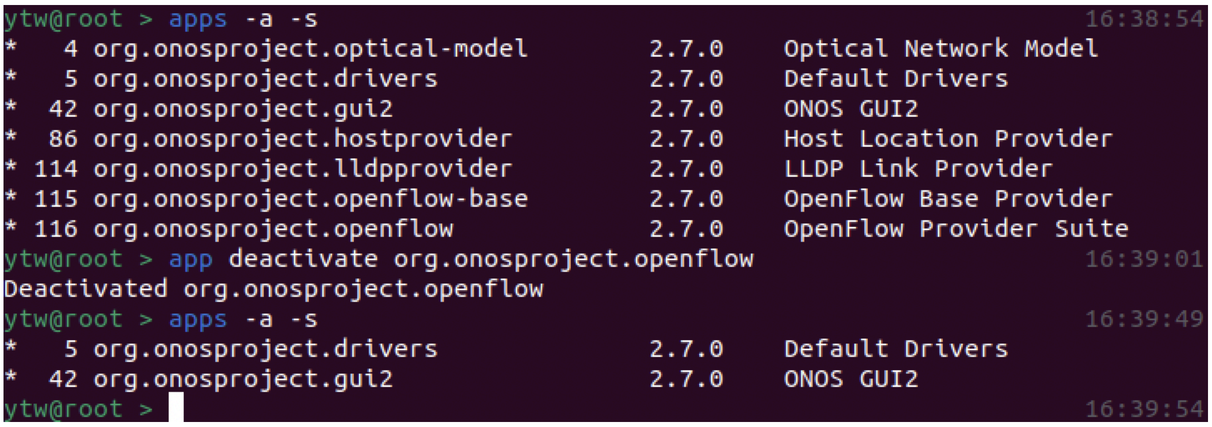
lab1-report

111550120 吳盈庭

Part1: Answer the Questions

1. When ONOS activates “org.onosproject.openflow,” what APPs does it activate?

   By the screenshot below, the APPS that are deactivated after deactivating “org.onosproject.openflow” are “**org.onosproject.optical-model**”, “**org.onosproject.hostprovider**”, “**org.onosproject.lldprovider**”, “**org.onosproject.openflow-bast**, “**org.onosproject.openflow**”.



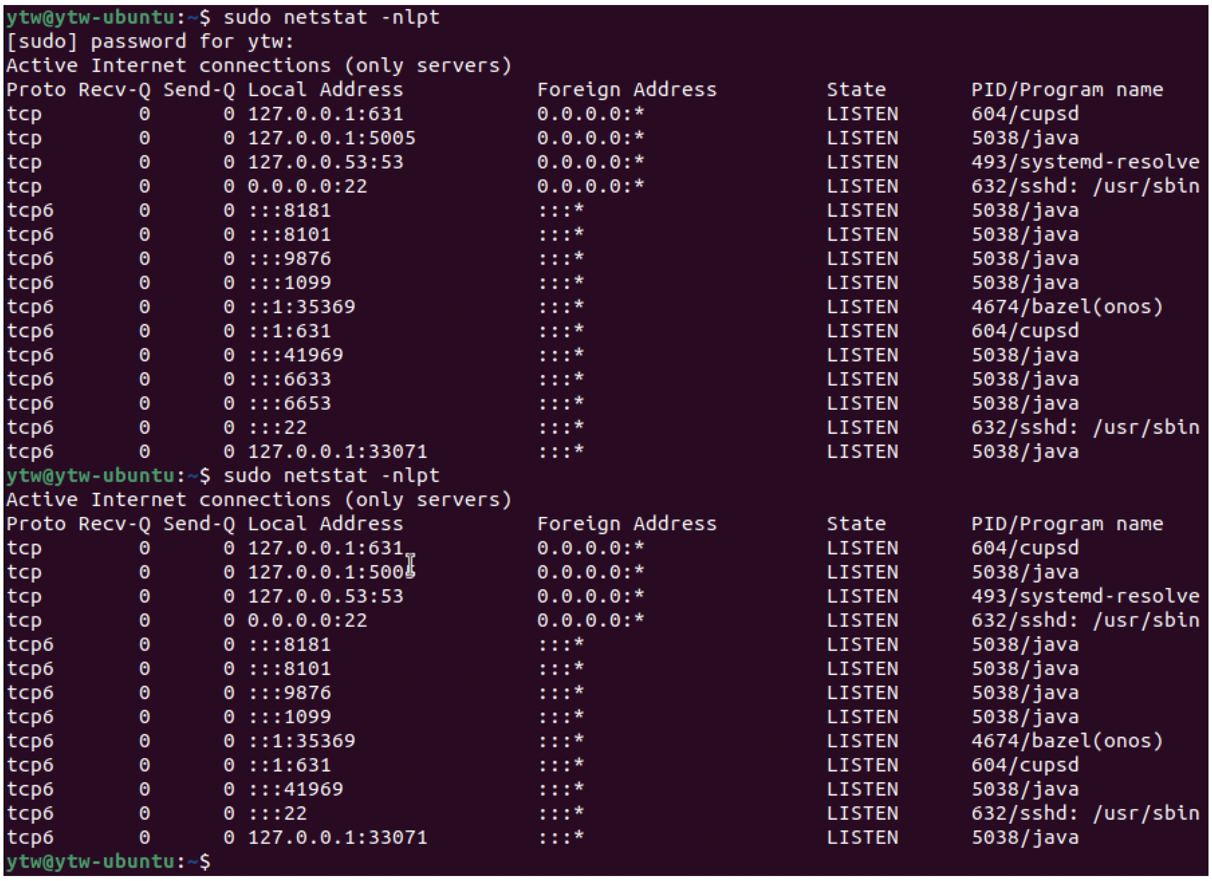
```
ytw@root > apps -a -s                                                       16:38:54
*   4 org.onosproject.optical-model       2.7.0    Optical Network Model
*   5 org.onosproject.drivers             2.7.0    Default Drivers
*  42 org.onosproject.gui2                2.7.0    ONOS GUI2
*  86 org.onosproject.hostprovider        2.7.0    Host Location Provider
* 114 org.onosproject.lldpprovider        2.7.0    LLDP Link Provider
* 115 org.onosproject.openflow-base       2.7.0    OpenFlow Base Provider
* 116 org.onosproject.openflow            2.7.0    OpenFlow Provider Suite
ytw@root > app deactivate org.onosproject.openflow                          16:39:01
Deactivated org.onosproject.openflow
ytw@root > apps -a -s                                                       16:39:49
*   5 org.onosproject.drivers             2.7.0    Default Drivers
*  42 org.onosproject.gui2                2.7.0    ONOS GUI2
ytw@root >                                                                  16:39:54
```

2. After we activate ONOS and run P.17 Mininet command, will H1 ping H2 successfully? Why or why not?

   No, because there are no flows on the data-plane. We can solve this by activating “org.onosproject.fwd”.

3. Which TCP port does the controller listen to the OpenFlow connection request from the switch? (Take a screenshot and explain your answer.)

   This is a screenshot before and after deactivating org.onosproject.openflow. We can see when openflow is activated (upper part of the screenshot), there are two more tcp6 port, **6653 and 6633**, which the controller listens for openflow connection requests.



```
ytw@ytw-ubuntu:~$ sudo netstat -nlpt
[sudo] password for ytw:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      604/cupsd
tcp        0      0 127.0.0.1:5005          0.0.0.0:*               LISTEN      5038/java
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      493/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      632/sshd: /usr/sbin
tcp6       0      0 :::8181                 :::*                    LISTEN      5038/java
tcp6       0      0 :::8101                 :::*                    LISTEN      5038/java
tcp6       0      0 :::9876                 :::*                    LISTEN      5038/java
tcp6       0      0 :::1099                 :::*                    LISTEN      5038/java
tcp6       0      0 ::1:35369               :::*                    LISTEN      4674/bazel(onos)
tcp6       0      0 ::1:631                 :::*                    LISTEN      604/cupsd
tcp6       0      0 :::41969                :::*                    LISTEN      5038/java
tcp6       0      0 :::6633                 :::*                    LISTEN      5038/java
tcp6       0      0 :::6653                 :::*                    LISTEN      5038/java
tcp6       0      0 :::22                   :::*                    LISTEN      632/sshd: /usr/sbin
tcp6       0      0 127.0.0.1:33071         :::*                    LISTEN      5038/java
ytw@ytw-ubuntu:~$ sudo netstat -nlpt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      604/cupsd
tcp        0      0 127.0.0.1:5005          0.0.0.0:*               LISTEN      5038/java
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      493/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      632/sshd: /usr/sbin
tcp6       0      0 :::8181                 :::*                    LISTEN      5038/java
tcp6       0      0 :::8101                 :::*                    LISTEN      5038/java
tcp6       0      0 :::9876                 :::*                    LISTEN      5038/java
tcp6       0      0 :::1099                 :::*                    LISTEN      5038/java
tcp6       0      0 ::1:35369               :::*                    LISTEN      4674/bazel(onos)
tcp6       0      0 ::1:631                 :::*                    LISTEN      604/cupsd
tcp6       0      0 :::41969                :::*                    LISTEN      5038/java
tcp6       0      0 :::22                   :::*                    LISTEN      632/sshd: /usr/sbin
tcp6       0      0 127.0.0.1:33071         :::*                    LISTEN      5038/java
ytw@ytw-ubuntu:~$
```

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

“**org.onosproject.openflow-base**”, by deactivating the apps one by one.

Part2: Create a Custom Topology

code:

```
from mininet.topo import Topo

class Lab1_Topo_111550120( Topo ):
   def __init__( self ):
       Topo.__init__( self )

       # Add hosts
       h1 = self.addHost( 'h1' )
       h2 = self.addHost( 'h2' )
       h3 = self.addHost( 'h3' )
       h4 = self.addHost( 'h4' )
       h5 = self.addHost( 'h5' )

       # Add switches
       s1 = self.addSwitch( 's1' )
       s2 = self.addSwitch( 's2' )
       s3 = self.addSwitch( 's3' )
       s4 = self.addSwitch( 's4' )

       # Add links
       self.addLink( h1, s1 )
       self.addLink( h2, s2 )
       self.addLink( h3, s3 )
       self.addLink( h4, s4 )
       self.addLink( h5, s4 )
       self.addLink( s1, s2 )
       self.addLink( s2, s3 )
       self.addLink( s2, s4 )

topos = { 'topo_part2_111550120': Lab1_Topo_111550120 }
```

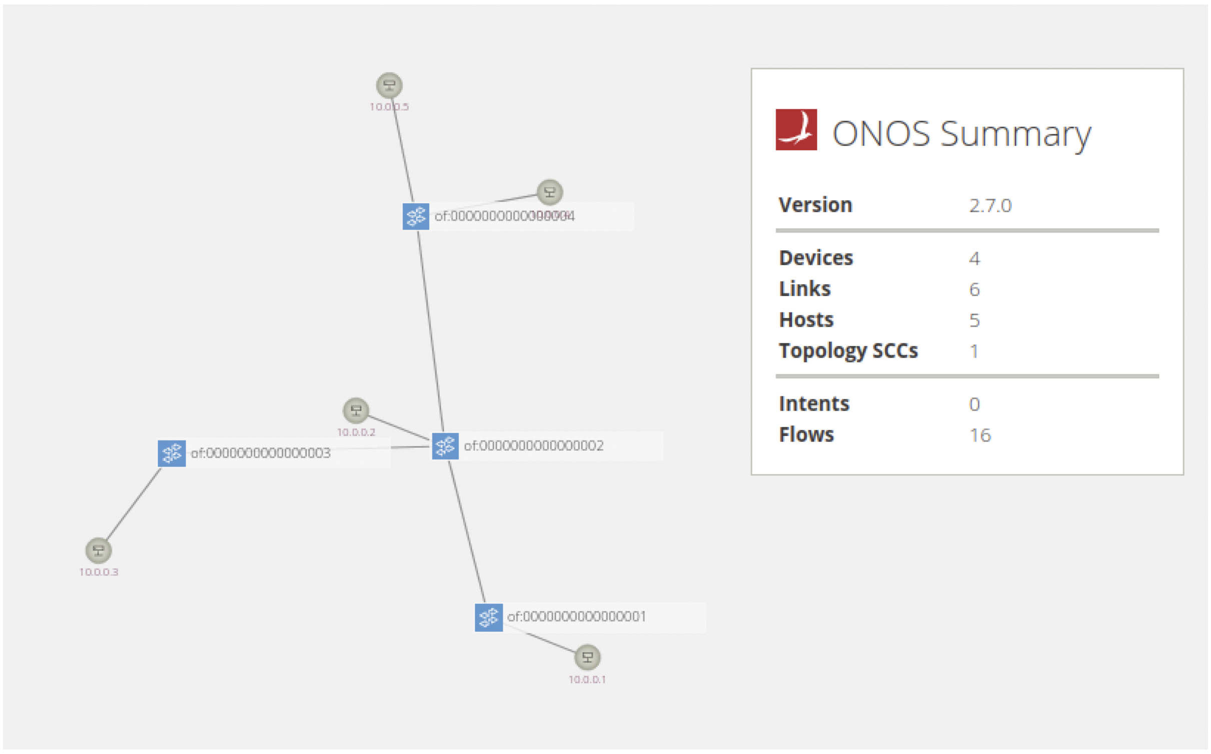
run the command:

```
sudo mn -custom=lab1_part2_111550120.py \
--topo=topo_part2_111550120 \
--controller=remote,ip=127.0.0.1:6653 \
--switch=ovs,protocols=OpenFlow14
```

after activating “org.onosproject.fwd”, run:

```
pingall
```

result GUI:



ONOS Summary

Version	2.7.0
Devices	4
Links	6
Hosts	5
Topology SCCs	1
Intents	0
Flows	16

Part3: Statically assign Hosts IP Address in Mininet

Steps are mostly the same as part2, except that we need extra parameters when adding hosts to assign IP addresses.

code:

```python
from mininet.topo import Topo

class Lab1_Topo_111550120( Topo ):
   def __init__( self ):
       Topo.__init__( self )

       # Add hosts
       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' )
       h4 = self.addHost( 'h4', ip = '192.168.0.4/27' )
       h5 = self.addHost( 'h5', ip = '192.168.0.5/27' )

       # Add switches
       s1 = self.addSwitch( 's1' )
       s2 = self.addSwitch( 's2' )
       s3 = self.addSwitch( 's3' )
       s4 = self.addSwitch( 's4' )

       # Add links
       self.addLink( h1, s1 )
       self.addLink( h2, s2 )
       self.addLink( h3, s3 )
       self.addLink( h4, s4 )
       self.addLink( h5, s4 )
       self.addLink( s1, s2 )
       self.addLink( s2, s3 )
       self.addLink( s2, s4 )


topos = { 'topo_part3_111550120': Lab1_Topo_111550120 }
```

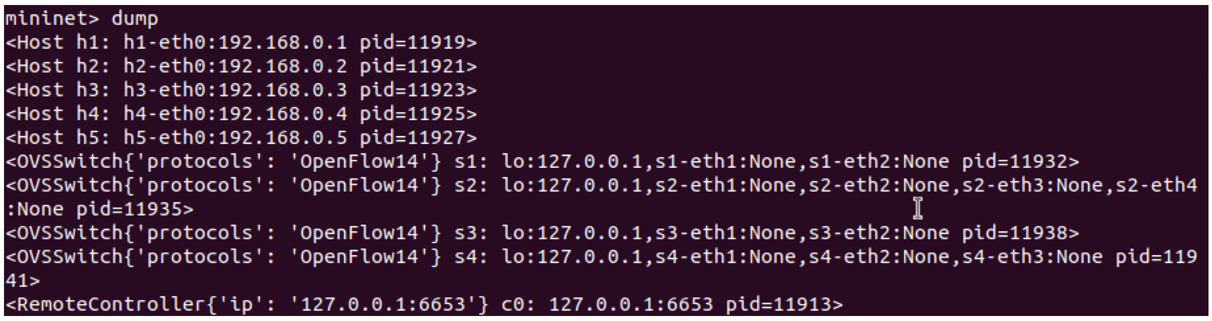
run the command:

```
sudo mn –custom=lab1_part2_111550120.py \
--topo=topo_part2_111550120 \
--controller=remote,ip=127.0.0.1:6653 \
--switch=ovs,protocols=OpenFlow14
```

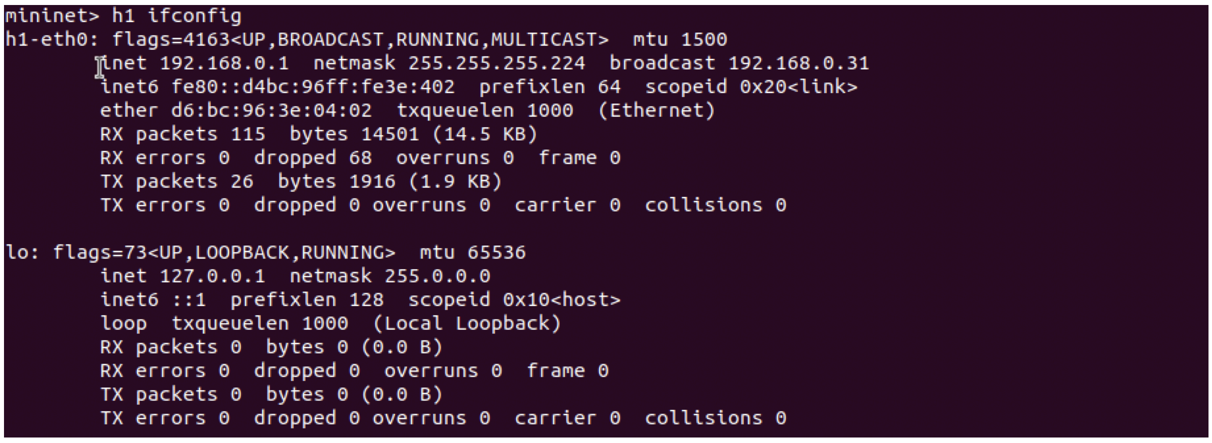
after activating “org.onosproject.fwd”, run:

```
pingall
```

result:

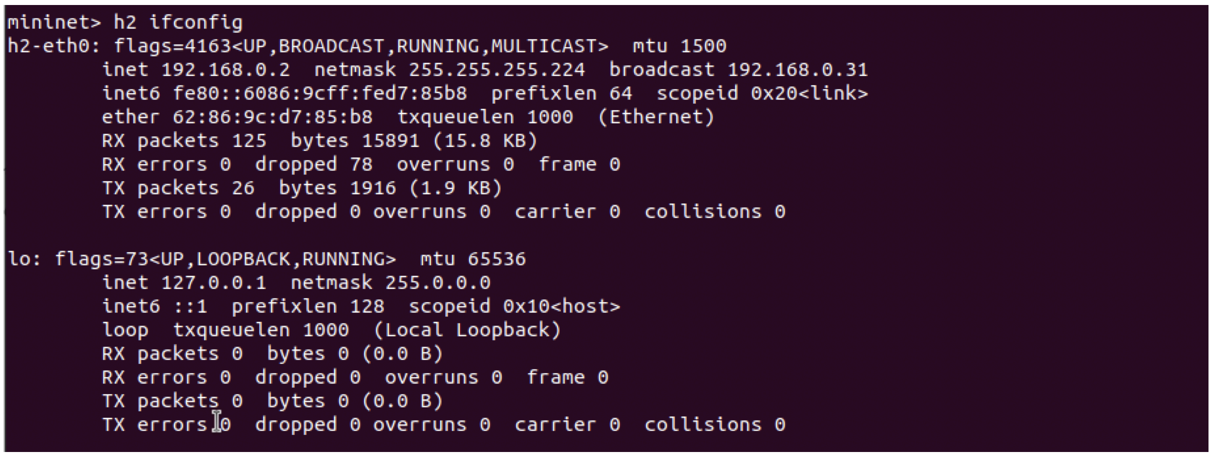


```
mininet> dump
<Host h1: h1-eth0:192.168.0.1 pid=11919>
<Host h2: h2-eth0:192.168.0.2 pid=11921>
<Host h3: h3-eth0:192.168.0.3 pid=11923>
<Host h4: h4-eth0:192.168.0.4 pid=11925>
<Host h5: h5-eth0:192.168.0.5 pid=11927>
<OVSSwitch{'protocols': 'OpenFlow14'} s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=11932>
<OVSSwitch{'protocols': 'OpenFlow14'} s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4
:None pid=11935>
<OVSSwitch{'protocols': 'OpenFlow14'} s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=11938>
<OVSSwitch{'protocols': 'OpenFlow14'} s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=119
41>
<RemoteController{'ip': '127.0.0.1:6653'} c0: 127.0.0.1:6653 pid=11913>
```



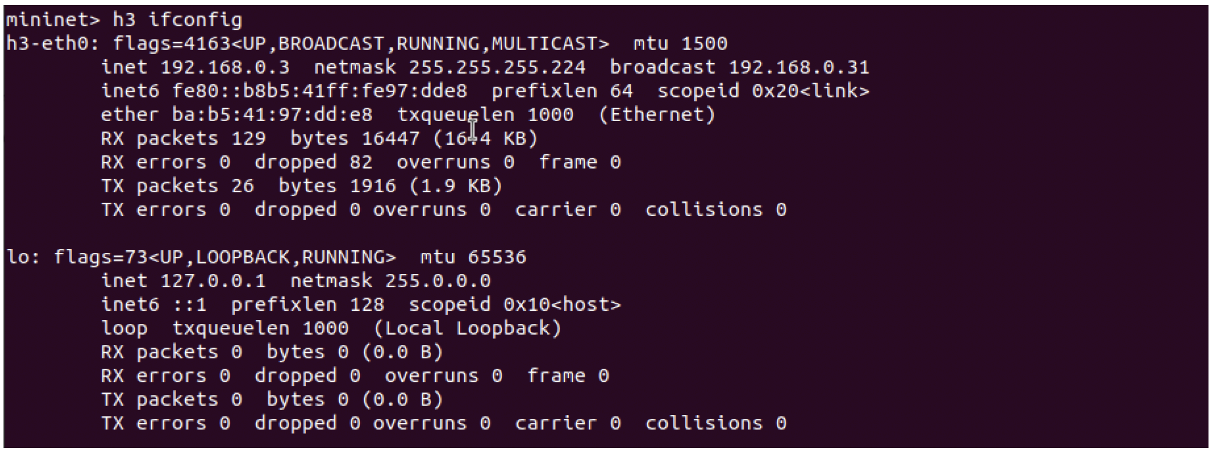
```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.1  netmask 255.255.255.224  broadcast 192.168.0.31
        inet6 fe80::d4bc:96ff:fe3e:402  prefixlen 64  scopeid 0x20<link>
        ether d6:bc:96:3e:04:02  txqueuelen 1000  (Ethernet)
        RX packets 115  bytes 14501 (14.5 KB)
        RX errors 0  dropped 68  overruns 0  frame 0
        TX packets 26  bytes 1916 (1.9 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
```



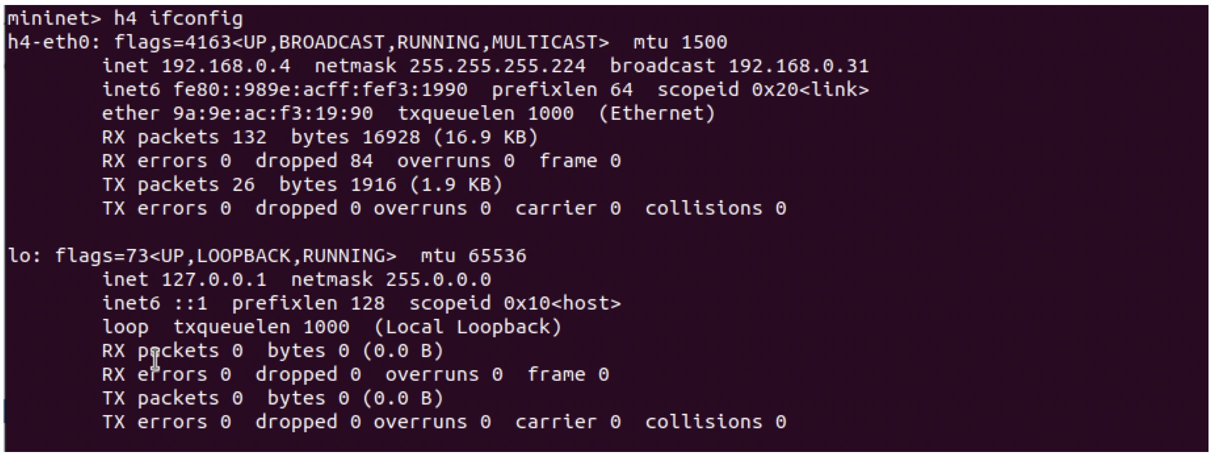
```
mininet> h2 ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.2  netmask 255.255.255.224  broadcast 192.168.0.31
        inet6 fe80::6086:9cff:fed7:85b8  prefixlen 64  scopeid 0x20<link>
        ether 62:86:9c:d7:85:b8  txqueuelen 1000  (Ethernet)
        RX packets 125  bytes 15891 (15.8 KB)
        RX errors 0  dropped 78  overruns 0  frame 0
        TX packets 26  bytes 1916 (1.9 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
```



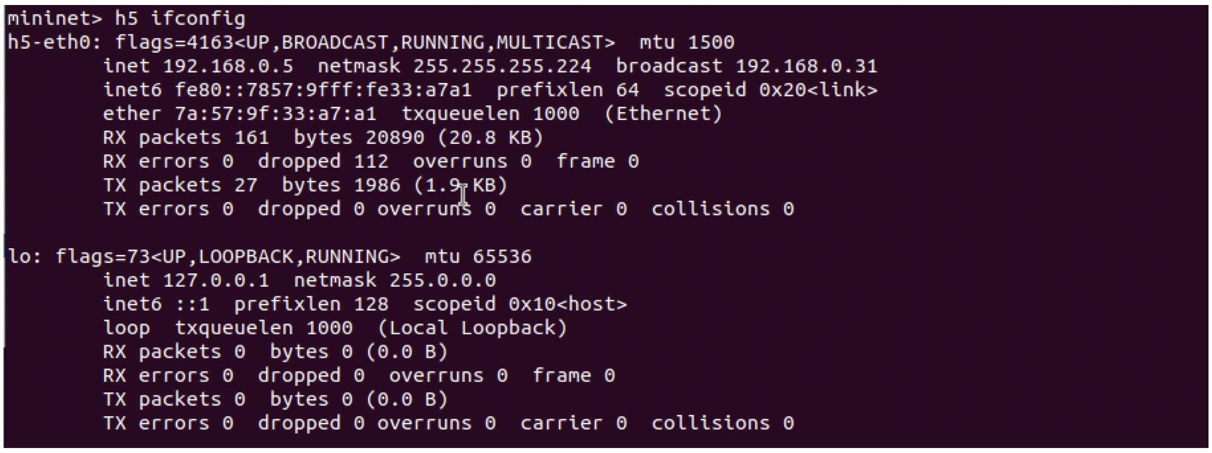
```
mininet> h3 ifconfig
h3-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.3  netmask 255.255.255.224  broadcast 192.168.0.31
        inet6 fe80::b8b5:41ff:fe97:dde8  prefixlen 64  scopeid 0x20<link>
        ether ba:b5:41:97:dd:e8  txqueuelen 1000  (Ethernet)
        RX packets 129  bytes 16447 (16.4 KB)
        RX errors 0  dropped 82  overruns 0  frame 0
        TX packets 26  bytes 1916 (1.9 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> 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::989e:acff:fef3:1990  prefixlen 64  scopeid 0x20<link>
        ether 9a:9e:ac:f3:19:90  txqueuelen 1000  (Ethernet)
        RX packets 132  bytes 16928 (16.9 KB)
        RX errors 0  dropped 84  overruns 0  frame 0
        TX packets 26  bytes 1916 (1.9 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> h5 ifconfig
h5-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.5  netmask 255.255.255.224  broadcast 192.168.0.31
        inet6 fe80::7857:9fff:fe33:a7a1  prefixlen 64  scopeid 0x20<link>
        ether 7a:57:9f:33:a7:a1  txqueuelen 1000  (Ethernet)
        RX packets 161  bytes 20890 (20.8 KB)
        RX errors 0  dropped 112  overruns 0  frame 0
        TX packets 27  bytes 1986 (1.9 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
```

Part4: What I Learned or Solved

I spent most of my time setting up the environment. I tried emulating a AMD64 Ubuntu in UTM first, but it is just awfully, painfully, unacceptably slow. I then tried using Docker, but something went wrong when building the display desktop and I couldn't solve it. Finally I succeeded by using qemu directly, even though it consumes power like a 10^n W light bulb.

After this lab, I am now more familiar with some basic commands in ONOS and mininet. I learned to build a topology and can assign IP addresses to hosts.