SDNFV Lab1

Part 1

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

   Apps enabled:

   - org.onosproject.hostprovider
   - org.onosproject.lldpprovider
   - org.onosproject.openflow-base
   - org.onosproject.openflow
   - org.onosproject.optical-model.

   Before enable

   Search | All Fields

	Title	App ID	Version	Category	Origin
✓	Default Drivers	org.onosproject.drivers	2.7.0	Drivers	ONOS Community
✓	ONOS GUI2	org.onosproject.gui2	2.7.0	Graphical User Interface	ONOS Community
■	Access Control Lists	org.onosproject.acl	2.7.0	Security	ONOS Community
■	Arista Drivers	org.onosproject.drivers.arista	2.7.0	Drivers	ONOS Community
■	Artemis	org.onosproject.artemis	2.7.0	Monitoring	ONOS Community
■	BGP Router	org.onosproject.bgprouter	2.7.0	Traffic Engineering	ONOS Community
■	BMv2 Drivers	org.onosproject.drivers.bmv2	2.7.0	Drivers	ONOS Community
■	Barefoot Drivers	org.onosproject.drivers.barefoot	2.7.0	Drivers	ONOS Community
■	Basic Optical Drivers	org.onosproject.drivers.optical	2.7.0	Drivers	ONOS Community
■	Basic Pipelines	org.onosproject.pipelines.basic	2.7.0	Pipeline	ONOS Community
■	CORD Support	org.onosproject.cord-support	2.7.0	Integration	ONOS Community

   After enable

   Applications (169 Total)

   Search | All Fields

	Title	App ID	Version	Category	Origin
✓	Default Drivers	org.onosproject.drivers	2.7.0	Drivers	ONOS Community
✓	Host Location Provider	org.onosproject.hostprovider	2.7.0	Provider	ONOS Community
✓	LLDP Link Provider	org.onosproject.lldpprovider	2.7.0	Provider	ONOS Community
✓	ONOS GUI2	org.onosproject.gui2	2.7.0	Graphical User Interface	ONOS Community
✓	OpenFlow Base Provider	org.onosproject.openflow-base	2.7.0	Provider	ONOS Community
✓	OpenFlow Provider Suite	org.onosproject.openflow	2.7.0	Provider	ONOS Community
✓	Optical Network Model	org.onosproject.optical-model	2.7.0	Optical	ONOS Community
■	Access Control Lists	org.onosproject.acl	2.7.0	Security	ONOS Community
■	Arista Drivers	org.onosproject.drivers.arista	2.7.0	Drivers	ONOS Community
■	Artemis	org.onosproject.artemis	2.7.0	Monitoring	ONOS Community

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

   H1 can ping H2 successfully (as shown in graph). The ping succeeded because I have the network connected and have the org.onosproject.fwd enabled.

```
stanley@SDN-NFV:~/onos$ sudo mn --topo=linear,3 --controller=remote,127.0.0.1:6653 --switch=ovs,protocols=OpenFlow14
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=9257>
<Host h2: h2-eth0:10.0.0.2 pid=9259>
<Host h3: h3-eth0:10.0.0.3 pid=9261>
<OVSSwitch{'protocols': 'OpenFlow14'} s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=9266>
<OVSSwitch{'protocols': 'OpenFlow14'} s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=9269>
<OVSSwitch{'protocols': 'OpenFlow14'} s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=9272>
<RemoteController{} c0: 127.0.0.1:6653 pid=9251>
mininet> h1 ping 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=32.2 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.182 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.048 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3090ms
rtt min/avg/max/mdev = 0.043/8.106/32.152/13.882 ms
mininet>
```

3. Which TCP port does the controller listen to the OpenFlow connection request from the switch?

   In the screenshot, you can see that the controller listens on port 6653 for the connection request from the switch.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=10656>
<Host h2: h2-eth0:10.0.0.2 pid=10658>
<Host h3: h3-eth0:10.0.0.3 pid=10660>
<Host h4: h4-eth0:10.0.0.4 pid=10662>
<OVSSwitch{'protocols': 'OpenFlow14'} s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=10667>
<OVSSwitch{'protocols': 'OpenFlow14'} s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=10670>
<OVSSwitch{'protocols': 'OpenFlow14'} s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=10673>
<OVSSwitch{'protocols': 'OpenFlow14'} s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None pid=10676>
<RemoteController{} c0: 127.0.0.1:6653 pid=10650>
```

   After a connection request is accepted, the controller and the switch will connect with different ports (6654, 6655, 6656…)

```
stanley@SDN-NFV:~/onos$ netstat -nltp
(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      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:5005          0.0.0.0:*               LISTEN      2925/java
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6657            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6656            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6655            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6654            0.0.0.0:*               LISTEN      -
```

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

   OpenFlow Base Provider (org.onosproject.openflow-base). After deleting the whole suite, I add the Optical Model first due to the requirement and add the OpenFlow Base Provider. After activating the base provider, the port 6653 is turned on.

```
stanley@SDN-NFV:~/onos$ netstat -nltp | grep 665
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:6657            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6656            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6655            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6654            0.0.0.0:*               LISTEN      -
tcp6       0      0 :::6653                 :::*                    LISTEN      15077/java
```

Applications (169 Total)

Search | All Fields

		Title	App ID	Version	Category
✔		Default Drivers	org.onosproject.drivers	2.7.0	Drivers
✔		ONOS GUI2	org.onosproject.gui2	2.7.0	Graphical User Interface
✔		OpenFlow Base Provider	org.onosproject.openflow-base	2.7.0	Provider
✔		Optical Network Model	org.onosproject.optical-model	2.7.0	Optical

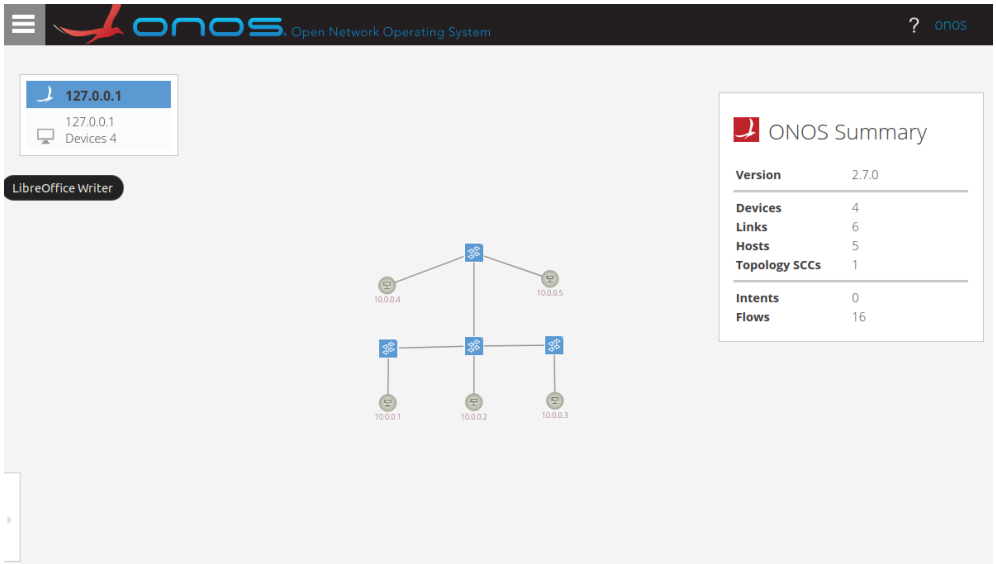
Part 2

Steps:

1. Write the python script to create the topology

```python
from mininet.topo import Topo

class Lab1_Topo_110705013( 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( s1, s2 )
        self.addLink( s2, s3 )
        self.addLink( h4, s4 )
        self.addLink( h5, s4 )
        self.addLink( s2, s4 )


topos = { 'topo_part2_110705013': Lab1_Topo_110705013 }
```

2. Run sudo mn command
3. Run pingall to have the controller find all the hosts
4. Goto 127.0.0.1:8181/onos/ui and login
5. Goto menu bar -> Topology
6. Take a screenshot (Result as following)



Part 3

Modify the code in part 2. Specify the host with static IP like “ h1 = self.addHost('h1', ip='192.168.0.1/27') ”

```
mininet> dump
<Host h1: h1-eth0:192.168.0.1 pid=4579>
<Host h2: h2-eth0:192.168.0.2 pid=4581>
<Host h3: h3-eth0:192.168.0.3 pid=4583>
<Host h4: h4-eth0:192.168.0.4 pid=4585>
<Host h5: h5-eth0:192.168.0.5 pid=4587>
<OVSSwitch{'protocols': 'OpenFlow14'} s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=4592>
<OVSSwitch{'protocols': 'OpenFlow14'} s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None pid=4595>
<OVSSwitch{'protocols': 'OpenFlow14'} s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=4598>
<OVSSwitch{'protocols': 'OpenFlow14'} s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=4601>
<RemoteController{'ip': '127.0.0.1:6653'} c0: 127.0.0.1:6653 pid=4573>
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::5c0e:f4ff:fe36:164c  prefixlen 64  scopeid 0x20<link>
        ether 5e:0e:f4:36:16:4c  txqueuelen 1000  (Ethernet)
        RX packets 59  bytes 7814 (7.8 KB)
        RX errors 0  dropped 36  overruns 0  frame 0
        TX packets 9  bytes 726 (726.0 B)
        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::b424:f1ff:fe6a:6857  prefixlen 64  scopeid 0x20<link>
        ether b6:24:f1:6a:68:57  txqueuelen 1000  (Ethernet)
        RX packets 69  bytes 9103 (9.1 KB)
        RX errors 0  dropped 44  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        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::bca5:a8ff:fefa:9d  prefixlen 64  scopeid 0x20<link>
        ether be:a5:a8:fa:00:9d  txqueuelen 1000  (Ethernet)
        RX packets 73  bytes 9659 (9.6 KB)
        RX errors 0  dropped 48  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        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::882b:2ff:fe4d:c2ab  prefixlen 64  scopeid 0x20<link>
        ether 8a:2b:02:4d:c2:ab  txqueuelen 1000  (Ethernet)
        RX packets 77  bytes 10215 (10.2 KB)
        RX errors 0  dropped 52  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        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::580e:4ff:fe36:a3ab  prefixlen 64  scopeid 0x20<link>
        ether 5a:0e:04:36:a3:ab  txqueuelen 1000  (Ethernet)
        RX packets 83  bytes 11049 (11.0 KB)
        RX errors 0  dropped 58  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        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
```

What I've learned or solved

In this lab

1. I installed virtual box, built an Ubuntu VM, install ONOS, Bazelisk, Mininet, and OvS with the environment setup file provided by TAs.
2. I learned how to activate control plane apps via CLI and GUI, gained more detailed knowledge on the applications after answering the questions in part 1.
3. I learned how to create a network with a specific topology using mininet, either by default topology or custom topology using python script.
4. I learned how to assign static IPs to the hosts.