

2023 SDN-NFV Project1

312552017 董家鳴

Part1 : Answer Questions

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

First, I deactivate all apps. And I activate the *openflow* to get the result. According to the Figure 1, the apps activated are:

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



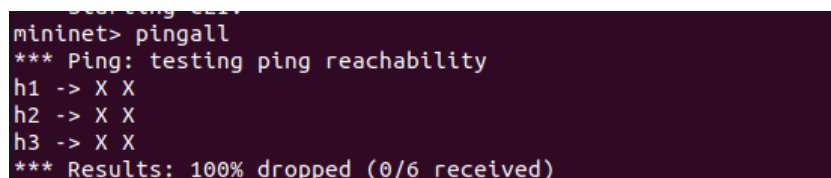
```
onosproject> app deactivate org.onosproject.optical-model org.onosproject.drivers org.onosproject.hostprovider org.onosproject.lldpprovider org.onosproject.openflow-base org.onosproject.openflow org.onosproject.guiz
deactivated org.onosproject.optical-model
deactivated org.onosproject.drivers
deactivated org.onosproject.hostprovider
deactivated org.onosproject.lldpprovider
deactivated org.onosproject.openflow-base
deactivated org.onosproject.openflow
deactivated org.onosproject.guiz
onosproject> app activate org.onosproject.openflow
activated org.onosproject.openflow
onosproject>
+ 4 org.onosproject.optical-model 2.7.0 Optical Network Model
+ 44 org.onosproject.hostprovider 2.7.0 Host Location Provider
+ 45 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider
+ 46 org.onosproject.openflow-base 2.7.0 Openflow Base Provider
+ 47 org.onosproject.openflow 2.7.0 Openflow Provider Suite
```

Figure 1: Apps activated by openflow

2. After activating ONOS and running the commands on P.17 and P.20. Will H1 ping H2 successfully? Why or why not? **Answer: No**

Since no flows installed on the data-plane, which forward the traffic appropriately. ONOS comes with a simple Reactive Forwarding app that installs forwarding flows on demand, but this application is not activated by default [1].

Therefore, under the default app, ping will fail, as shown in Figure 2. After enabling app org.onosproject.fwd, ping will succeed, as shown in Figure 3.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
```

Figure 2: failed ping

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

Figure 3: Successful ping

3. Which TCP port the controller listens for the OpenFlow connection request from the switch? **Answer: 6653**

Let us observe the packets through wireshark. From Figure 4 we can see the TCP packets of the openflow connection, and from Figure 5 we can see the ephemeral source port of the switch (from which the switch established its connection to the controller), so we can infer port 6653 TCP port for controller listens for.

8493 584.549988664 127.0.0.1	127.0.0.1	OpenFlow	124 Type: OFPT_MULTIPART_REQUEST
8494 584.541927077 127.0.0.1	127.0.0.1	TCP	68 57818 -> 6653 [ACK] Seq=26891 Ack=5965 Win=44032 Len=0 TSval=392795735 TSecr=392795735
8495 584.541313919 127.0.0.1	127.0.0.1	OpenFlow	468 Type: OFPT_MULTIPART_REPLY
8496 584.541340189 127.0.0.1	127.0.0.1	TCP	68 6653 -> 57818 [ACK] Seq=5965 Ack=27291 Win=44032 Len=0 TSval=392795736 TSecr=392795736
8497 584.541552187 127.0.0.1	127.0.0.1	OpenFlow	84 Type: OFPT_MULTIPART_REQUEST
8498 584.541672323 127.0.0.1	127.0.0.1	OpenFlow	6180 Type: OFPT_MULTIPART_REPLY
8499 584.584806987 127.0.0.1	127.0.0.1	TCP	68 6653 -> 57818 [ACK] Seq=5981 Ack=33403 Win=44032 Len=0 TSval=392795779 TSecr=392795736
8500 584.598255389 127.0.0.1	127.0.0.1	OpenFlow	92 Type: OFPT_MULTIPART_REQUEST
8501 584.598544892 127.0.0.1	127.0.0.1	OpenFlow	990 Type: OFPT_MULTIPART_REPLY (formed Packet)
8502 584.598617463 127.0.0.1	127.0.0.1	TCP	68 6653 -> 57818 [ACK] Seq=6085 Ack=34331 Win=44032 Len=0 TSval=392795785 TSecr=392795785
8503 584.598635656 127.0.0.1	127.0.0.1	OpenFlow	148 Type: OFPT_MULTIPART_REQUEST
8504 584.598655555 127.0.0.1	127.0.0.1	OpenFlow	68 57820 -> 6653 [ACK] Seq=39553 Ack=8487 Win=44032 Len=0 TSval=392795792 TSecr=392795792
8505 584.598629913 127.0.0.1	127.0.0.1	OpenFlow	468 Type: OFPT_MULTIPART_REPLY
8506 584.598472598 127.0.0.1	127.0.0.1	OpenFlow	6180 Type: OFPT_MULTIPART_REPLY
8507 584.598519978 127.0.0.1	127.0.0.1	TCP	68 6653 -> 57820 [ACK] Seq=8487 Ack=37865 Win=48448 Len=0 TSval=392795793 TSecr=392795793
8508 584.598533847 127.0.0.1	127.0.0.1	OpenFlow	148 Type: OFPT_MULTIPART_REQUEST
8509 584.598623884 127.0.0.1	127.0.0.1	OpenFlow	468 Type: OFPT_MULTIPART_REPLY
8510 584.598628556 127.0.0.1	127.0.0.1	TCP	68 6653 -> 57818 [ACK] Seq=5981 Ack=27291 Win=44032 Len=0 TSval=392795793 TSecr=392795793

Figure 4: Wireshark packets

```
openflowroot> devices
id=0f:0000000000000001, available=true, local-status=connected 4n7s app, role=MASTER, type=SWITCH, nfr=Nicira, Inc., hw=Open vSwitch, sw=2.17.2, serial=None, chassis=1, driver=ovs, ch
annelId=127.0.0.1:57818, datapathDescriptions1, managementAddress=127.0.0.1, protocol=OF_14
id=0f:0000000000000002, available=true, local-status=connected 4n7s app, role=MASTER, type=SWITCH, nfr=Nicira, Inc., hw=Open vSwitch, sw=2.17.2, serial=None, chassis=2, driver=ovs, ch
annelId=127.0.0.1:57820, datapathDescriptions2, managementAddress=127.0.0.1, protocol=OF_14
id=0f:0000000000000003, available=true, local-status=connected 4n7s app, role=MASTER, type=SWITCH, nfr=Nicira, Inc., hw=Open vSwitch, sw=2.17.2, serial=None, chassis=3, driver=ovs, ch
annelId=127.0.0.1:57818, datapathDescriptions3, managementAddress=127.0.0.1, protocol=OF_14
```

Figure 5: Switch devices

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

Answer: org.onosproject.openflowbase

Because the openflow app opened four other apps at one time, I first turned off the openflow app and found that the 6653 port was still open, as shown in Figure 6. Next, I turned off the openflow-base app and found that the 6653 port was closed, as shown in Figure 7, and the wireshark packets could not be transmitted, as shown in Figure 8, so it was inferred that the openflow-base app enables the controller to listen on the TCP port. The closing process is shown in Figure 9.

```
sdn-nfv@sdnfv-VirtualBox:~$ 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 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 0.0.0.0:6656           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      -
tcp        0      0 127.0.0.1:5005         0.0.0.0:*               LISTEN      2313/java
tcp        0      0 127.0.0.1:631          0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                  :::*                     LISTEN      -
tcp6       0      0 :::9876                 :::*                     LISTEN      2313/java
tcp6       0      0 :::38783                :::*                     LISTEN      2313/java
tcp6       0      0 :::1:631                :::*                     LISTEN      -
tcp6       0      0 :::1099                  :::*                     LISTEN      2313/java
tcp6       0      0 :::6633                  :::*                     LISTEN      2313/java
tcp6       0      0 :::6653                  :::*                     LISTEN      2313/java
tcp6       0      0 :::8101                  :::*                     LISTEN      2313/java
tcp6       0      0 :::8181                  :::*                     LISTEN      2313/java
tcp6       0      0 :::1:43325               :::*                     LISTEN      1984/bazel(onos)
tcp6       0      0 127.0.0.1:36925        :::*                     LISTEN      2313/java
```

Figure 6: Wireshark packets

```
sdn-nfv@sdnfv-VirtualBox:~$ 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 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 0.0.0.0:6656           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      -
tcp        0      0 127.0.0.1:5005         0.0.0.0:*               LISTEN      2313/java
tcp        0      0 127.0.0.1:631          0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                  :::*                     LISTEN      -
tcp6       0      0 :::9876                 :::*                     LISTEN      2313/java
tcp6       0      0 :::38783                :::*                     LISTEN      2313/java
tcp6       0      0 :::1:631                :::*                     LISTEN      -
tcp6       0      0 :::1099                  :::*                     LISTEN      2313/java
tcp6       0      0 :::8101                  :::*                     LISTEN      2313/java
tcp6       0      0 :::8181                  :::*                     LISTEN      2313/java
tcp6       0      0 :::1:43325               :::*                     LISTEN      1984/bazel(onos)
tcp6       0      0 127.0.0.1:36925        :::*                     LISTEN      2313/java
```

Figure 7: Switch devices

```
6678 386.90621814 127.0.0.1 127.0.0.1 TCP 56 6653 - 56648 [EST] Seq=1 Ack=1 Win=0 Len=0
6679 386.90645316 127.0.0.1 127.0.0.1 TCP 56 6653 - 56652 [SYN] Seq=0 Win=0 Len=0 SACK_PERM=1 Tsvail=02597181 TSecr=0 WS=512
6680 386.906484539 127.0.0.1 127.0.0.1 TCP 56 6653 - 56658 [EST] Seq=1 Ack=1 Win=0 Len=0
6681 386.906497629 127.0.0.1 127.0.0.1 TCP 56 6653 - 56658 [SYN] Seq=0 Win=0 Len=0 SACK_PERM=1 Tsvail=02597181 TSecr=0 WS=512
6682 386.906497783 127.0.0.1 127.0.0.1 TCP 56 6653 - 56684 [EST] Seq=1 Ack=1 Win=0 Len=0
6683 386.907037429 127.0.0.1 127.0.0.1 TCP 56 6653 - 56658 [SYN] Seq=0 Win=0 Len=0 SACK_PERM=1 Tsvail=02598182 TSecr=0 WS=512
6684 386.907263784 127.0.0.1 127.0.0.1 TCP 56 6653 - 44000 [EST] Seq=1 Ack=1 Win=0 Len=0
6685 386.908292729 127.0.0.1 127.0.0.1 TCP 56 6653 - 44000 [EST] Seq=1 Ack=1 Win=0 Len=0
6686 386.908296789 127.0.0.1 127.0.0.1 TCP 56 6653 - 44004 [EST] Seq=1 Ack=1 Win=0 Len=0
6687 386.908302729 127.0.0.1 127.0.0.1 TCP 56 6653 - 44004 [SYN] Seq=0 Win=0 Len=0 SACK_PERM=1 Tsvail=02600182 TSecr=0 WS=512
6688 386.908142684 127.0.0.1 127.0.0.1 TCP 56 6653 - 44000 [EST] Seq=1 Ack=1 Win=0 Len=0
6689 386.908291259 127.0.0.1 127.0.0.1 TCP 56 6653 - 44000 [SYN] Seq=0 Win=0 Len=0 SACK_PERM=1 Tsvail=02600184 TSecr=0 WS=512
6690 387.009931523 127.0.0.1 127.0.0.1 TCP 56 6653 - 44070 [EST] Seq=1 Ack=1 Win=0 Len=0
6691 317.410016538 127.0.0.1 127.0.0.1 TCP 76 44042 - 44053 [SYN] Seq=0 Win=4368 Len=0 MSS=65495 SACK_PERM=1 Tsvail=02600184 TSecr=0 WS=512
6692 317.410044440 127.0.0.1 127.0.0.1 TCP 56 6653 - 44052 [EST] Seq=1 Ack=1 Win=0 Len=0
6693 317.410120968 127.0.0.1 127.0.0.1 TCP 76 44058 - 6653 [SYN] Seq=0 Win=4368 Len=0 MSS=65495 SACK_PERM=1 Tsvail=02600184 TSecr=0 WS=512
6694 317.410150518 127.0.0.1 127.0.0.1 TCP 76 44058 - 6653 [SYN] Seq=0 Win=4368 Len=0 MSS=65495 SACK_PERM=1 Tsvail=02600184 TSecr=0 WS=512
6695 318.425601438 127.0.0.1 127.0.0.1 TCP 76 44108 - 6653 [SYN] Seq=0 Win=4368 Len=0 MSS=65495 SACK_PERM=1 Tsvail=02600184 TSecr=0 WS=512
```

Figure 8: Switch devices

```
sdn-nfv@root> apps -a -s
* 4 org.onosproject.optical-model 2.7.0 Optical Network Model
* 5 org.onosproject.drivers 2.7.0 Default Drivers
* 44 org.onosproject.hostprovider 2.7.0 Host Location Provider
* 45 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider
* 46 org.onosproject.openFlow-base 2.7.0 OpenFlow Base Provider
* 47 org.onosproject.openFlow 2.7.0 OpenFlow Provider Suite

sdn-nfv@root> app deactivate org.onosproject.openflow
Deactivated org.onosproject.openflow
sdn-nfv@root> app deactivate org.onosproject.openflow-base
Deactivated org.onosproject.openflow-base
sdn-nfv@root>

sdn-nfv@root> apps -a -s
* 5 org.onosproject.drivers 2.7.0 Default Drivers
* 44 org.onosproject.hostprovider 2.7.0 Host Location Provider
* 45 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider

sdn-nfv@root> app activate org.onosproject.optical-model
Activated org.onosproject.optical-model
sdn-nfv@root> apps -a -s
* 4 org.onosproject.optical-model 2.7.0 Optical Network Model
* 5 org.onosproject.drivers 2.7.0 Default Drivers
* 44 org.onosproject.hostprovider 2.7.0 Host Location Provider
* 45 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider
```

Figure 9: Switch devices

Part2 : Create a custom Topology

Write a Python script to build the following topology

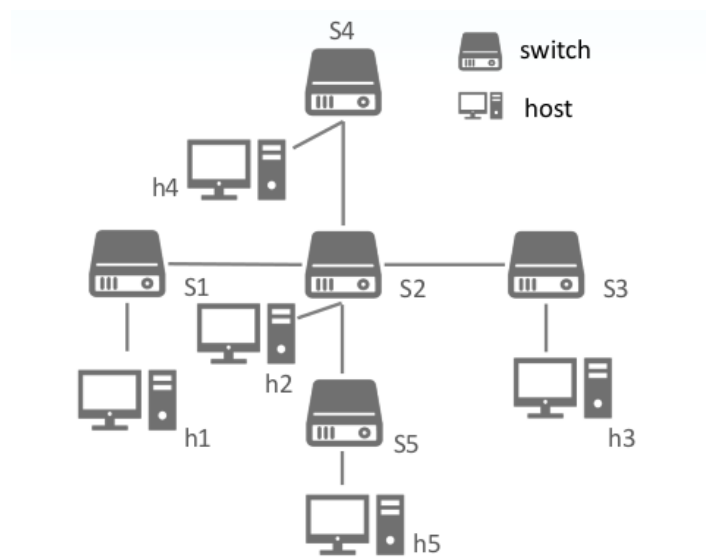


Figure 10: Target topology

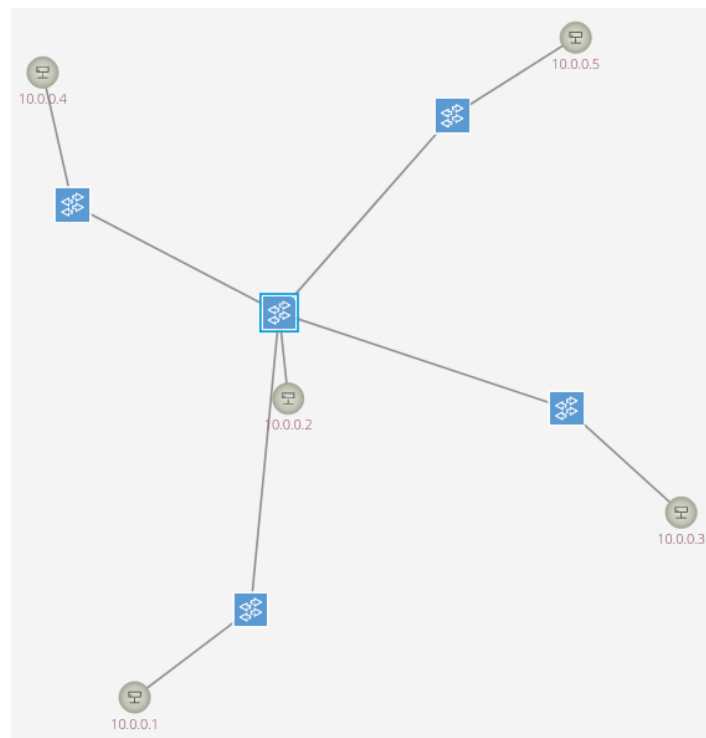


Figure 11: My topology on GUI

```

from mininet.topo import Topo

class Project1_Topo_312552017( 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' )
        s5 = self.addSwitch( 's5' )

        # Add links, host to switch
        self.addLink( h1, s1 )
        self.addLink( h2, s2 )
        self.addLink( h3, s3 )
        self.addLink( h4, s4 )
        self.addLink( h5, s5 )

        # Add links, switch to switch
        self.addLink( s1, s2 )
        self.addLink( s3, s2 )
        self.addLink( s4, s2 )
        self.addLink( s5, s2 )

topos = { 'topo_part2_312552017': Project1_Topo_312552017 }

```

Figure 12: Python script for part 2

Part3 : Statically assign Hosts IP Address in Mininet

The format for manual assignment of host IP address is in Figure 13 shown. All the components are shown in Figure 14 by using *dump*. And the *ifconfig* of all hosts are shown in Figure 15, 16, 17.

- Format for manual assignment of host IP address:

- 192.168.0.0/27
- netmask 255.255.255.224

Host	IP Address
h1	192.168.0.1
h2	192.168.0.2
	...

Figure 13: IP format

```

mininet> dump
<Host h1: h1-eth0:192.168.0.1 pId=12736>
<Host h2: h2-eth0:192.168.0.2 pId=12738>
<Host h3: h3-eth0:192.168.0.3 pId=12740>
<Host h4: h4-eth0:192.168.0.4 pId=12742>
<Host h5: h5-eth0:192.168.0.5 pId=12744>
<OVSSwitch["protocols": "OpenFlow14"] s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pId=12749>
<OVSSwitch["protocols": "OpenFlow14"] s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,s2-eth5:None pId=12752>
<OVSSwitch["protocols": "OpenFlow14"] s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pId=12755>
<OVSSwitch["protocols": "OpenFlow14"] s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None pId=12758>
<OVSSwitch["protocols": "OpenFlow14"] s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None pId=12761>
<RemoteController["ip": "127.0.0.1", "port": 6653] c0: 127.0.0.1:6653 pId=12730>

```

Figure 14: mininet dump

```

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::647b:60ff:fe2d:6e5b prefixlen 64 scopeid 0x20<link>
    ether 66:7b:60:2d:6e:5b txqueuelen 1000 (Ethernet)
    RX packets 137 bytes 17740 (17.7 KB)
    RX errors 0 dropped 88 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

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::c429:deff:febe:8735 prefixlen 64 scopeid 0x20<link>
    ether c6:29:de:be:87:35 txqueuelen 1000 (Ethernet)
    RX packets 141 bytes 18296 (18.2 KB)
    RX errors 0 dropped 92 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

```

Figure 15: h1 and h2 interface config

```

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::b402:2ff:fe4f:5b00 prefixlen 64 scopeid 0x20<link>
    ether b6:02:02:4f:5b:00 txqueuelen 1000 (Ethernet)
    RX packets 145 bytes 18852 (18.8 KB)
    RX errors 0 dropped 96 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

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::40de:2dff:fe33:1750 prefixlen 64 scopeid 0x20<link>
    ether 42:de:2d:33:17:50 txqueuelen 1000 (Ethernet)
    RX packets 149 bytes 19408 (19.4 KB)
    RX errors 0 dropped 100 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

```

Figure 16: h3 and h4 interface config

```

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::5898:53ff:fe45:cba8 prefixlen 64 scopeid 0x20<link>
ether 5a:98:53:45:cb:a8 txqueuelen 1000 (Ethernet)
RX packets 150 bytes 19498 (19.4 KB)
RX errors 0 dropped 100 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

```

Figure 17: h5 interface config

```

from mininet.topo import Topo

class Project1_Topo_312552017( Topo ):
    def __init__( self ):
        Topo.__init__( self )

        # IP Format
        base_ip = '192.168.0.{}'
        subnet = '/27'

        # Add hosts
        h1 = self.addHost( 'h1', ip=base_ip.format(1) + subnet)
        h2 = self.addHost( 'h2', ip=base_ip.format(2) + subnet)
        h3 = self.addHost( 'h3', ip=base_ip.format(3) + subnet)
        h4 = self.addHost( 'h4', ip=base_ip.format(4) + subnet)
        h5 = self.addHost( 'h5', ip=base_ip.format(5) + subnet)

        # Add switches
        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )
        s3 = self.addSwitch( 's3' )
        s4 = self.addSwitch( 's4' )
        s5 = self.addSwitch( 's5' )

        # Add links, host to switch
        self.addLink( h1, s1 )
        self.addLink( h2, s2 )
        self.addLink( h3, s3 )
        self.addLink( h4, s4 )
        self.addLink( h5, s5 )

        # Add links, switch to switch
        self.addLink( s1, s2 )
        self.addLink( s3, s2 )
        self.addLink( s4, s2 )
        self.addLink( s5, s2 )

topos = { 'topo_part3_312552017': Project1_Topo_312552017 }

```

Figure 18: Python script for part 3

What you' ve learned or solved

1. **An app may require other apps.** For example, when we want to activate the *org.onosproject.openflow* app, it automatically activates other apps.
2. **Reactive Forwarding.** Forwarding application is not activated by default. Every time

we open ONOS, we need to activate *org.onosproject.fwd* app and use ping (ARP request and ICMP Echo request) to identify the MAC address of the remote computer and confirm whether the target computer is accessible [2]. In addition, we can also see how forward works through open source code [3].

3. Port that the controller and switches listens for. The managementAddress of all switches is the same in the Figure 5. This means that any communication between ONOS and the switch will take place at the IP address of ONOS and managementAddress, besides, the individual switches are distinguished by the port number in the channelId field [4].

References

- [1] ONOS Wiki. *Basic ONOS Tutorial*. 2019. URL: <https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial>.
- [2] Sandip Roy. *Protocols Used for PING*. URL: <https://www.baeldung.com/cs/protocols-ping>.
- [3] ONOS Project. *onos*. URL: <https://github.com/opennetworkinglab/onos/tree/master/apps/fwd/src/main/java/org/onosproject/fwd>.
- [4] mirantis. *Software Defined Networking*. URL: <http://trainer.edu.mirantis.com/SDN50/sdn.html>.