NC864 SDN - Assignment 1

By: **Shathin R Rao [MT2020067]**

Repository: github.com/Shathin/nc864-sdn

The Script

`script.py` creates a network with a randomized topology using the Python Mininet API.

The hosts are randomly connected to a switch and the switches are connected sequentially i.e., $S_1 \rightarrow S_2 \rightarrow ... \rightarrow S_n$.

Each host h_i is assigned the IP address $10.0.0.i/24$ and a MAC address of $00:00:00:00:00:i$.

OpenDayLight is used as the remote controller for the generated network.

```
sudo python3 script.py [options]
```

Options

- `--controller` → Defines the OpenDayLight controller's IP address. This is a mandatory option.

  Usage example: `--controller=192.168.122.61`

- `--hosts` → Defines the number of hosts to be created in the network. This option is optional, skipping this defaults the value of the number of hosts to 12.

  Usage example: `--hosts=12`

- `--switches` → Defines the number of switches to be created in the network. This option is optional, skipping this defaults the value of the number of switches to 4.

  Usage example: `--switches=4`

- `--bw` → Defines the bandwidth range to be used. Bandwidth is in $Mbps$. This option is optional, skipping this defaults the value to the range $0 \rightarrow 5$ $Mbps$.

  Usage example: `--bw=0,5`

- `--delay` → Defines the link delay range to be used. Delay is in ms. This option is optional, skipping this defaults the value to the range $2 \rightarrow 30$ ms.

  Usage example: `--delay=2,30`

📌 Be sure to clean up before executing the script for another time. Run `sudo mn -c` command to perform the cleanup.

Following is the screenshot which shows the execution of the which creates a network with 12 hosts and 4 switches and randomly assigns an bandwidth to each link in the range $(1, 5)$ $Mbps$ and a delay in the range of $(2, 30)$ ms -

```
ryuu@shathin-ubuntu:~/.../assignment-1$ sudo python3 script.py --controller=192.168.122.173 --hosts=12 --switches=4 --bw=1,5 --delay=2,30
Node mapping :
        s2
                <- ('h1', '3 Mbps', '25ms')
                <- ('h3', '2 Mbps', '26ms')
                <- ('h6', '1 Mbps', '6ms')
                <- ('h8', '5 Mbps', '5ms')
                <- ('h12', '5 Mbps', '5ms')
                <- ('s3', '5 Mbps', '4ms')
        s4
                <- ('h2', '3 Mbps', '9ms')
                <- ('h5', '1 Mbps', '16ms')
                <- ('h7', '4 Mbps', '28ms')
                <- ('h9', '5 Mbps', '15ms')
                <- ('h10', '5 Mbps', '2ms')
        s1
                <- ('h4', '4 Mbps', '28ms')
                <- ('s2', '5 Mbps', '4ms')
        s3
                <- ('h11', '3 Mbps', '14ms')
                <- ('s4', '2 Mbps', '10ms')
mininet>
```

Mininet Commands

- `net` → List network connections.

```
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s4-eth1
h3 h3-eth0:s2-eth2
h4 h4-eth0:s1-eth1
h5 h5-eth0:s4-eth2
h6 h6-eth0:s2-eth3
h7 h7-eth0:s4-eth3
h8 h8-eth0:s2-eth4
h9 h9-eth0:s4-eth4
h10 h10-eth0:s4-eth5
h11 h11-eth0:s3-eth1
h12 h12-eth0:s2-eth5
s1 lo:  s1-eth1:h4-eth0 s1-eth2:s2-eth6
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h3-eth0 s2-eth3:h6-eth0 s2-eth4:h8-eth0 s2-eth5:h12-eth0 s2-eth6:s1-eth2 s2-eth7:s3-eth2
s3 lo:  s3-eth1:h11-eth0 s3-eth2:s2-eth7 s3-eth3:s4-eth6
s4 lo:  s4-eth1:h2-eth0 s4-eth2:h5-eth0 s4-eth3:h7-eth0 s4-eth4:h9-eth0 s4-eth5:h10-eth0 s4-eth6:s3-eth3
c0
```

- `pingall` → Ping between all hosts.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
```

- `<host-a> ping <host-b>` → Host A pings Host B.

```
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=96.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=96.3 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=96.2 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=96.2 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=96.2 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=96.2 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5006ms
rtt min/avg/max/mdev = 96.154/96.325/96.886/0.259 ms
```

- `dump` → Dump node info. Contains node type and name, node's IP for each interface and PID.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=35475>
<Host h2: h2-eth0:10.0.0.2 pid=35477>
<Host h3: h3-eth0:10.0.0.3 pid=35479>
<Host h4: h4-eth0:10.0.0.4 pid=35481>
<Host h5: h5-eth0:10.0.0.5 pid=35483>
<Host h6: h6-eth0:10.0.0.6 pid=35485>
<Host h7: h7-eth0:10.0.0.7 pid=35487>
<Host h8: h8-eth0:10.0.0.8 pid=35489>
<Host h9: h9-eth0:10.0.0.9 pid=35491>
<Host h10: h10-eth0:10.0.0.10 pid=35493>
<Host h11: h11-eth0:10.0.0.11 pid=35495>
<Host h12: h12-eth0:10.0.0.12 pid=35497>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=35502>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,s2-eth5:None,s2-eth6:None,s2-eth7:None pid=35505>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=35508>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-eth4:None,s4-eth5:None,s4-eth6:None pid=35511>
<RemoteController c0: 192.168.122.173:6653 pid=35469>
```

- `nodes` → List all nodes.

```
mininet> nodes
available nodes are:
c0 h1 h10 h11 h12 h2 h3 h4 h5 h6 h7 h8 h9 s1 s2 s3 s4
```

- `links` → Report on links.

```
mininet> links
h1-eth0<->s2-eth1 (OK OK)
h2-eth0<->s4-eth1 (OK OK)
h3-eth0<->s2-eth2 (OK OK)
h4-eth0<->s1-eth1 (OK OK)
h5-eth0<->s4-eth2 (OK OK)
h6-eth0<->s2-eth3 (OK OK)
h7-eth0<->s4-eth3 (OK OK)
h8-eth0<->s2-eth4 (OK OK)
h9-eth0<->s4-eth4 (OK OK)
h10-eth0<->s4-eth5 (OK OK)
h11-eth0<->s3-eth1 (OK OK)
h12-eth0<->s2-eth5 (OK OK)
s1-eth2<->s2-eth6 (OK OK)
s2-eth7<->s3-eth2 (OK OK)
s3-eth3<->s4-eth6 (OK OK)
```

- `<host> ifconfig` → Get network details of `host`.

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::200:ff:fe00:1  prefixlen 64  scopeid 0x20<link>
        ether 00:00:00:00:00:01  txqueuelen 1000  (Ethernet)
        RX packets 365  bytes 28927 (28.9 KB)
        RX errors 0  dropped 62  overruns 0  frame 0
        TX packets 67  bytes 4994 (4.9 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
```

- `<switch> ifconfig` → Get network details of `switch`.

```
mininet> s3 ifconfig
enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 30:e1:71:9a:51:e1  txqueuelen 1000  (Ethernet)
        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

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 13125  bytes 1786520 (1.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13125  bytes 1786520 (1.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::30ea:54ff:fe1f:aafe  prefixlen 64  scopeid 0x20<link>
        ether 32:ea:54:1f:aa:fe  txqueuelen 1000  (Ethernet)
        RX packets 58  bytes 4232 (4.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 370  bytes 29257 (29.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::d45f:2aff:fee9:5e5f  prefixlen 64  scopeid 0x20<link>
        ether d6:5f:2a:e9:5e:5f  txqueuelen 1000  (Ethernet)
        RX packets 356  bytes 27301 (27.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 156  bytes 14292 (14.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Small section of the entire result

- `py host.IP()` → Display `host`'s IP address.

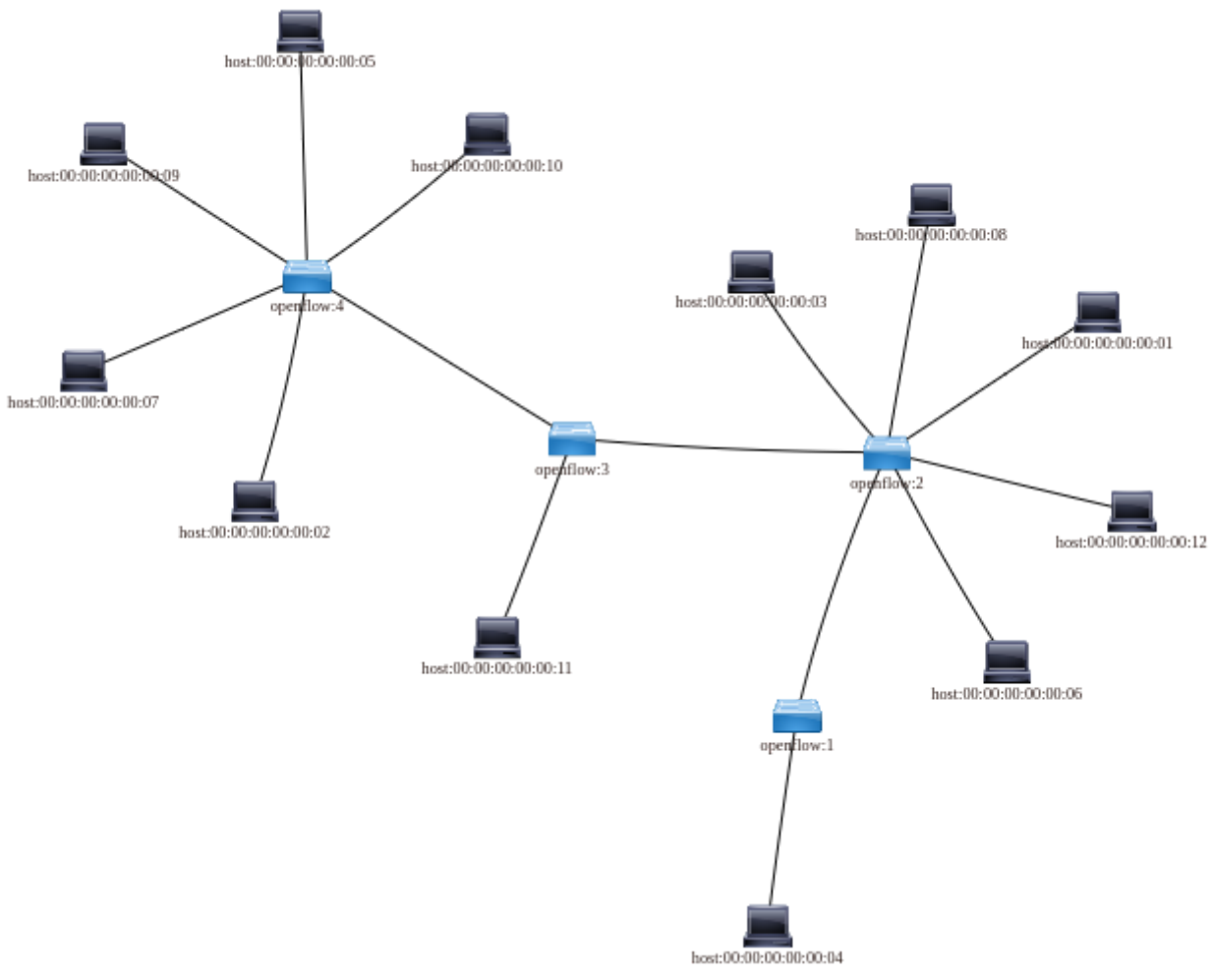
```
mininet> py h10.IP()
10.0.0.10
```

- `py host.MAC()` → Dispay `host`'s MAC address

```
mininet> py h10.MAC()
00:00:00:00:00:10
```

OpenDayLight

The topology of the network can be viewed on the DLUX UI which is accessible at - `http://<controllerIP>:8181/index.html#/topology`



OpenVSwitch

If the following commands are executed inside the Mininet prompt then prepend the below commands with `sh`. Also, add the `--protocols=OpenFlow13` options to all the command since we use OpenFlow13 protocol in the program.

- `ovs-ofctl dump-flows sw` → List out all the flow rules for the switch `sw`.

```
mininet> sh ovs-ofctl --protocols=OpenFlow13 dump-flows s3
 cookie=0x2b00000000000033, duration=653.356s, table=0, n_packets=260, n_bytes=22100, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b000000000000c9, duration=649.480s, table=0, n_packets=18, n_bytes=952, priority=2,in_port="s3-eth1" actions=output:"s3-eth2",output:"s3-eth3",CONTROLLER:65535
 cookie=0x2b000000000000ca, duration=649.480s, table=0, n_packets=140, n_bytes=10251, priority=2,in_port="s3-eth2" actions=output:"s3-eth1",output:"s3-eth3",CONTROLLER:65535
 cookie=0x2b000000000000cb, duration=649.480s, table=0, n_packets=108, n_bytes=7278, priority=2,in_port="s3-eth3" actions=output:"s3-eth1",output:"s3-eth2",CONTROLLER:65535
 cookie=0x2b00000000000033, duration=653.356s, table=0, n_packets=29, n_bytes=4427, priority=0 actions=drop
```

Flow rule for Switch 3 (`s3`)

- `ovs-ofctl del-flows sw` → Delete all the flows rules of the switch `sw`
- `ovs-ofctl add-flow sw priority=p,ip,nw_src=src,nw_dst=dst,actions=act` →Add a flow rule to switch `sw` based on the source IP `src` and destination IP `dst` addresses. To drop the packet specify the action as `drop`

```
mininet> h4 ping -c1 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=111 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 111.350/111.350/111.350/0.000 ms
```

Testing connectivity between Host `h4` (connected to Switch `s1`) and Host `h2` (connected to Switch `s4`)

The following adds a flow rule to Switch `s4` to drop the packets sent from Host `h4` [10.0.0.4] to Host `h2` [10.0.0.2].

```
sh ovs-ofctl --protocols=OpenFlow13 add-flow s4 priority=6969,ip,nw_src=10.0.0.4,nw_dst=10.0.0.2,actions=drop
```

```
mininet> h4 ping -c1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

mininet> h2 ping -c1 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.

--- 10.0.0.4 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Testing connectivity between Host `h4` (connected to Switch `s1`) and Host `h2` (connected to Switch `s4`) after adding the new flow rule

```
mininet> h10 ping -c1 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=97.5 ms

--- 10.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 97.484/97.484/97.484/0.000 ms
```

Testing connectivity between Host `h4` (connected to Switch `s1`) and Host `h10` (connected to Switch `s4`) after adding the new flow rule

- `ovs-ofctl add-flow sw priority=p,dl_src=smac,dl_dst=dmac,actions=act` → Add a flow rule to switch `sw` based on the source MAC `smac` and destination MAC `dmac` addresses. To drop the packet specify the action as `drop`.

```
mininet> h1 ping -c1 h11
PING 10.0.0.11 (10.0.0.11) 56(84) bytes of data.
64 bytes from 10.0.0.11: icmp_seq=1 ttl=64 time=86.6 ms

--- 10.0.0.11 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 86.609/86.609/86.609/0.000 ms
```

Testing connectivity between Host `h1` (connected to Switch `s2`) and Host `h11` (connected to Switch `s3`)

Adding flow rule to Switch `s3` to **drop** the packets sent from Host `h1` [00 : 00 : 00 : 00 : 00 : 01] to Host `h11` [00 : 00 : 00 : 00 : 00 : 11].

```
sh ovs-ofctl --protocols=OpenFlow13 add-flow s3 priority=6969,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:11,actions=drop
```

```
mininet> h1 ping -c1 h11
PING 10.0.0.11 (10.0.0.11) 56(84) bytes of data.

--- 10.0.0.11 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Testing connectivity between Host `h1` (connected to Switch `s2`) and Host `h11` (connected to Switch `s3`) after adding flow rule

```
mininet> h2 ping -c1 h11
PING 10.0.0.11 (10.0.0.11) 56(84) bytes of data.
64 bytes from 10.0.0.11: icmp_seq=1 ttl=64 time=66.8 ms

--- 10.0.0.11 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 66.758/66.758/66.758/0.000 ms
```

Testing connectivity between Host `h2` (connected to Switch `s2`) and Host `h11` (connected to Switch `s3`) after adding flow rule