SDN Fundamentals & Techniques

Demo 1: Mininet - as an SDN emulator

Task 1

- Create using the CLI interface and the OpenFlow Reference Controller:

  – A single-switch topology with 10 hosts. Note that a “single-switch topology” means one switch connected to all hosts.

```
ubuntu@docker:~$ sudo mn --topo single,10
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1) (h10, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Using command-line interface creating a single-switch Mininet network topology with single switch and 10 hosts.
"`sudo mn`" command is invoking mininet software and mininet required superuser privileges.
"`--topo`" flag of the command is indicating the specs of the topology of the network.
"`single`" refers to a single switch and "`10`" specifies the number of hosts.

– A linear topology of 5 switches and 5 hosts.

```
ubuntu@docker:~$ sudo mn --topo linear,5
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller

*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet>
```

Using command-line interface creating a linear Mininet network topology with 5 switches and 5 hosts.

As the previous one in this case, “`linear`” refers to mininet linear topology of the network and “`5`” specified the number of hosts.

– A tree topology depth 3 fanout 2.

```
ubuntu@docker:~$ sudo mn --topo tree,depth=3,fanout=2
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller

*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet>
```

Similarly, “`tree`” refers to mininet tree topology of the network. “`depth=3`” and “`fanout=2`” specified the network topology depth and fanout respectively.

- Similar tasks using OpenFlow Reference controller:

```
ubuntu@docker:~$ sudo mn --controller remote,ip=127.0.0.1 --topo single,10
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1) (h10, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

```
ubuntu@docker:~$ sudo mn --controller remote,ip=127.0.0.1 --topo linear,5
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet>
```

```
ubuntu@docker:~$ sudo mn --controller remote,ip=1287.0.0.1 --topo tree,depth=3,fanout=2
*** Creating network
*** Adding controller
Unable to contact the remote controller at 1287.0.0.1:6653
Unable to contact the remote controller at 1287.0.0.1:6633
Setting remote controller to 1287.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet>
```

As like before, but here `"--controller remote,ip=127.0.0.1"` option specifies that the controller for the network topology will be remote and will have the IP address 127.0.0.1

Task 2

```
ubuntu@docker:~$ docker pull onosproject/onos
Using default tag: latest
latest: Pulling from onosproject/onos
Digest: sha256:5f4b73dd3c24ae2251c274090d25c582b69639c4759c69db25226b4a503f4f77
Status: Image is up to date for onosproject/onos:latest
docker.io/onosproject/onos:latest
ubuntu@docker:~$ docker run -t -d -p 6653:6653 -p 8181:8181 -p 8101:8101 -p 5005:5005 -p 830:830 --env JAVA_DEBUG_PORT="0.0.0.0:5005" --n
ame onos onosproject/onos debug
174a390345da032caa2927d1129263dfddaf597c6a888fc9be4219274e6c3a7d
ubuntu@docker:~$
ubuntu@docker:~$ docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -q  --filter ancestor=onosproje
ct/onos)
172.17.0.2
ubuntu@docker:~$
```

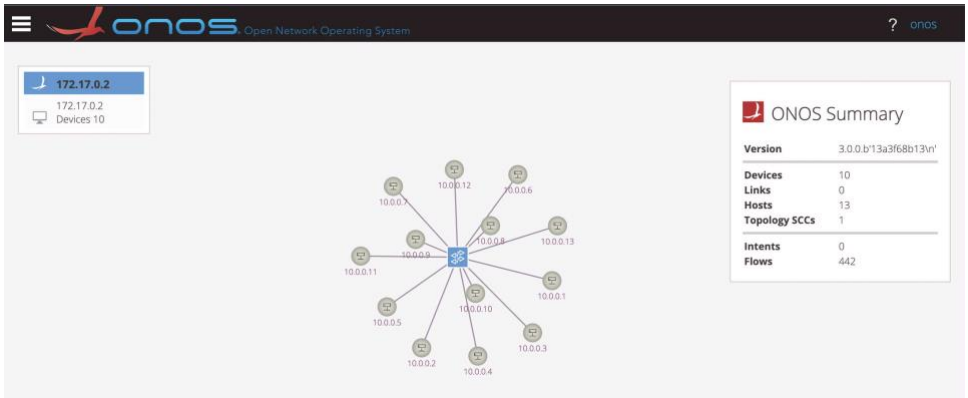
Here, I setup the environment for the Task 2 & 3 manually as instructed in the documentation.

- Create using the Python API interface and the ONOS SDN Controller:
  - A single-switch topology with 13 hosts. Note that a “single-switch topology” means one switch connected to all hosts.
  - Below is the code snipet for the task
  - Mininet CLI command & output which excutes the python script for the task.
  - ONOS screenshot is the result that shows the network topology

single-switch.py > SingleSwitchTopo > build

```python
#!/usr/bin/python
# -*- single switch topo, 13 hosts-*-
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.util import dumpNodeConnections

class SingleSwitchTopo(Topo):
    "Single switch connected to n hosts."
    def build(self, n=13):
        switch = self.addSwitch('s1', protocols='OpenFlow13')
        # Python's range(N) generates 0..N-1
        for h in range(n):
            host = self.addHost('h%s' % (h + 1))
            self.addLink(host, switch)

if __name__ == '__main__':
    setLogLevel('info')
    topo = SingleSwitchTopo(n=13)
    net = Mininet( topo=topo, controller=None)
    net.addController( 'c0', controller=RemoteController, ip='127.0.0.1', port=6653 )
    net.start()
    dumpNodeConnections(net.hosts)
    CLI(net)
    net.pingAll()
    net.stop()
```

```
ubuntu@docker:~/e-sdn$ sudo python3 single-switch.py
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1) (h10, s1) (h11, s1) (h12, s1) (h13, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
*** Starting controller
c0
*** Starting 1 switches
s1 ...
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
h5 h5-eth0:s1-eth5
h6 h6-eth0:s1-eth6
h7 h7-eth0:s1-eth7
h8 h8-eth0:s1-eth8
h9 h9-eth0:s1-eth9
h10 h10-eth0:s1-eth10
h11 h11-eth0:s1-eth11
h12 h12-eth0:s1-eth12
h13 h13-eth0:s1-eth13
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Results: 0% dropped (156/156 received)
mininet> c
```

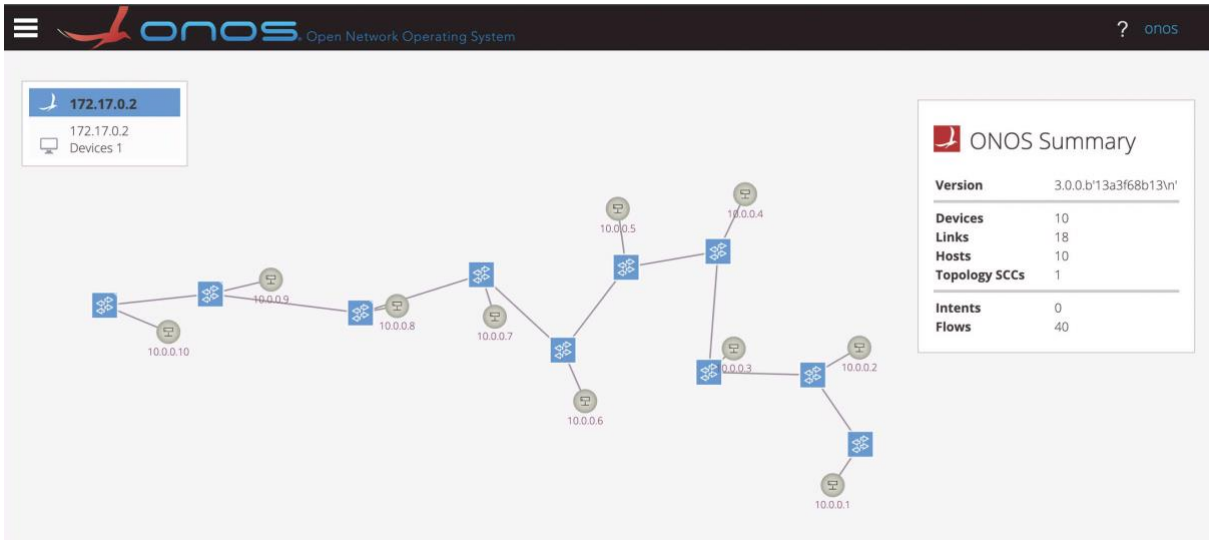


- A linear topology of 10 switches and 10 hosts.

```
#!/usr/bin/python
# -*- linear topo, 10 hosts, 10 switches -*-
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.util import dumpNodeConnections

class LinearTopo(Topo):
    "Linear topology with 10 hosts and 10 switches."
    def build(self):
        switches = []
        hosts = []

        # create 10 switches
        for i in range(1, 11):
            switch = self.addSwitch('s{}'.format(i), protocols='OpenFlow13')
            switches.append(switch)

        # create 10 hosts and connect each to a switch
        for i in range(1, 11):
            host = self.addHost('h{}'.format(i))
            hosts.append(host)
            self.addLink(host, switches[i-1])

        # connect switches to each other
```

```python
        for i in range(1, 10):
            self.addLink(switches[i-1], switches[i])


if __name__ == '__main__':
    setLogLevel('info')
    topo = LinearTopo()
    net = Mininet(topo=topo, controller=None)
    net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6653)
    net.start()
    dumpNodeConnections(net.hosts)
    CLI(net)
    net.pingAll()
    net.stop()
```

```
ubuntu@docker:~/e-sdn$ sudo python3 linear-topology.py
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (h7, s7) (h8, s8) (h9, s9) (h10, s10) (s1, s2) (s2, s3) (s3, s4) (s4, s5) (s5, s6) (s6, s7)
 (s7, s8) (s8, s9) (s9, s10)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c0
*** Starting 10 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 ...
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
h5 h5-eth0:s5-eth1
h6 h6-eth0:s6-eth1
h7 h7-eth0:s7-eth1
h8 h8-eth0:s8-eth1
h9 h9-eth0:s9-eth1
h10 h10-eth0:s10-eth1
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet>
```

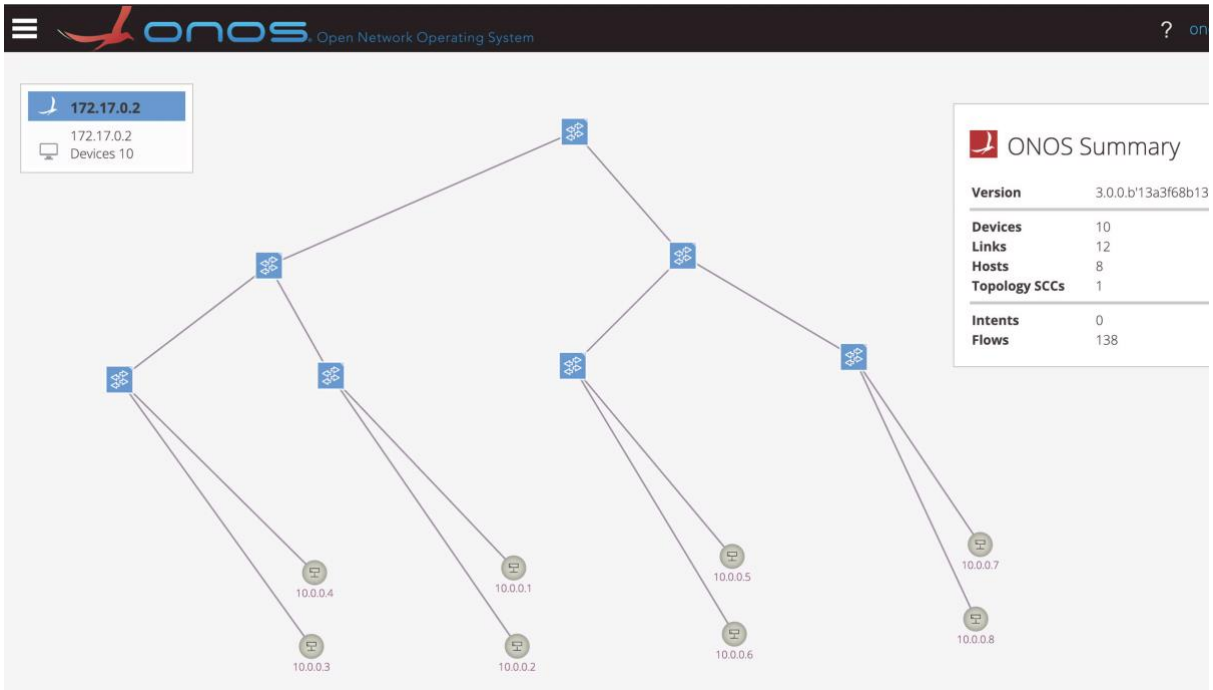


- A tree topology depth 3 fanout 2.

```
#!/usr/bin/python
# -*- tree topo, depth 3, fanout 2 -*-
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.util import dumpNodeConnections

class TreeTopo( Topo ):
    "Topology for a tree network with a given depth and fanout."

    def build( self, depth=1, fanout=2 ):
        # Numbering:  h1..N, s1..M
        self.hostNum = 1
        self.switchNum = 1
        # Build topology
        self.addTree( depth, fanout )

    def addTree( self, depth, fanout ):
        """Add a subtree starting with node n.
           returns: last node added"""
        isSwitch = depth > 0
        if isSwitch:
```

```
        node = self.addSwitch( 's%s' % self.switchNum )
        self.switchNum += 1
        for _ in range( fanout ):
            child = self.addTree( depth - 1, fanout )
            self.addLink( node, child )
    else:
        node = self.addHost( 'h%s' % self.hostNum )
        self.hostNum += 1
    return node

if __name__ == '__main__':
    setLogLevel('info')
    topo = TreeTopo()
    net = Mininet(topo=topo, controller=None)
    net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6653)
    net.start()
    dumpNodeConnections(net.hosts)
    CLI(net)
    net.pingAll()
    net.stop()
```

```
ubuntu@docker:~/e-sdn$ sudo python3 tree-topology.py
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, s4) (h2, s4) (h3, s5) (h4, s5) (h5, s6) (h6, s6) (h7, s7) (h8, s7) (s2, s1) (s3, s1) (s4, s2) (s5, s2) (s6, s3) (s7, s3)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
h1 h1-eth0:s4-eth2
h2 h2-eth0:s4-eth3
h3 h3-eth0:s5-eth2
h4 h4-eth0:s5-eth3
h5 h5-eth0:s6-eth2
h6 h6-eth0:s6-eth3
h7 h7-eth0:s7-eth2
h8 h8-eth0:s7-eth3
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>
```



Task 3

Try to automate the precedent task, i.e., Task 2, to be able to create a topology given the type, i.e., tree or linear, and for each type its specifications.

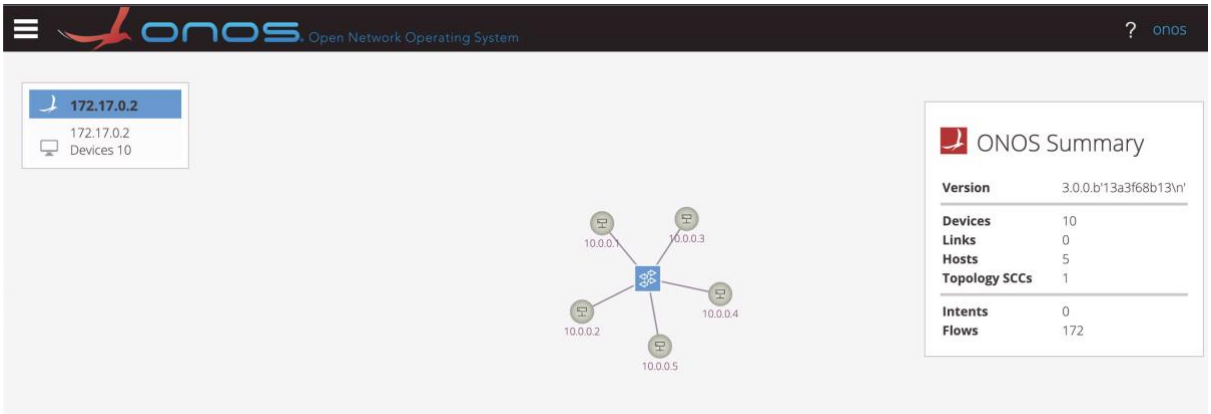
```
#!/usr/bin/python
# -*- automate single, linear, tree topology using user input-*-
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.util import dumpNodeConnections

class SingleSwitchTopo(Topo):
    "Single switch connected to hosts_input."
    def build(self, hosts_input):
        switch = self.addSwitch('s1', protocols='OpenFlow13')
        # Python's range(N) generates 0..N-1
        for h in range(hosts_input):
```

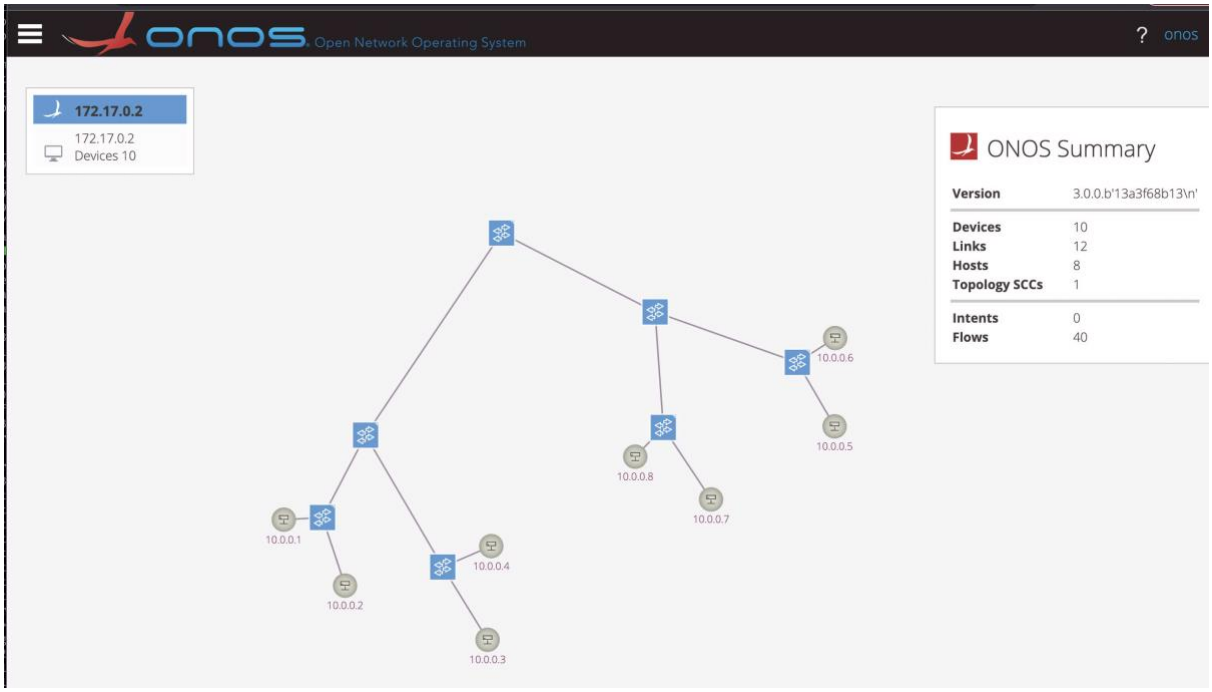
```
        host = self.addHost('h%s' % (h + 1))
        self.addLink(host, switch)

class LinearTopo(Topo):
    "Linear topology with hosts_input and switches_input."

    def build(self, hosts_input,switches_input):
        switches = []
        hosts = []

        # create switches
        for i in range(1, switches_input+1):
            switch = self.addSwitch('s{}'.format(i), protocols='OpenFlow13')
            switches.append(switch)

        # create hosts and connect each to a switch
        for i in range(1, hosts_input+1):
            host = self.addHost('h{}'.format(i))
            hosts.append(host)
            self.addLink(host, switches[i-1])

        # connect switches to each other
        for i in range(1, switches_input):
            self.addLink(switches[i-1], switches[i])

class TreeTopo( Topo ):
    "Topology for a tree network with a given depth and fanout."

    def build( self, depth=1, fanout=2 ):
        # Numbering:  h1..N, s1..M
        self.hostNum = 1
        self.switchNum = 1
        # Build topology
        self.addTree( depth, fanout )

    def addTree( self, depth, fanout ):
        """Add a subtree starting with node n.
           returns: last node added"""
        isSwitch = depth > 0
        if isSwitch:
```

```
        node = self.addSwitch( 's%s' % self.switchNum, protocols='OpenFlow13' )
        self.switchNum += 1
        for _ in range( fanout ):
            child = self.addTree( depth - 1, fanout )
            self.addLink( node, child )
    else:
        node = self.addHost( 'h%s' % self.hostNum )
        self.hostNum += 1
    return node

if __name__ == '__main__':
    setLogLevel('info')
    topo = input("Enter topology type (single, linear, tree): ")
    if topo == "single":
        hosts_input = int(input("Enter number of hosts: "))
        topo = SingleSwitchTopo(hosts_input=hosts_input)
    elif topo == "linear":
        hosts_input = int(input("Enter number of hosts: "))
        switches_input = int(input("Enter number of switches: "))
        topo = LinearTopo(hosts_input=hosts_input, switches_input=switches_input)
    elif topo == "tree":
        depth_input = int(input("Enter depth of tree: "))
        fanout_input = int(input("Enter fanout of tree: "))
        topo = TreeTopo(depth=depth_input, fanout=fanout_input)
    net = Mininet(topo=topo, controller=None)
    net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6653)
    net.start()
    dumpNodeConnections(net.hosts)
    CLI(net)
    net.pingAll()
    net.stop()
```

```
ubuntu@docker:~/e-sdn$ sudo python3 automate-topology.py
Enter topology type (single, linear, tree): linear
Enter number of hosts: 5
Enter number of switches: 5
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s1, s2) (s2, s3) (s3, s4) (s4, s5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
h5 h5-eth0:s5-eth1
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 → h2 h3 h4 h5
h2 → h1 h3 h4 h5
h3 → h1 h2 h4 h5
h4 → h1 h2 h3 h5
h5 → h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet>
```

```
ubuntu@docker:~/e-sdn$ sudo python3 automate-topology.py
Enter topology type (single, linear, tree): single
Enter number of hosts: 5
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 1 switches
s1 ...
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
h5 h5-eth0:s1-eth5
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 → h2 h3 h4 h5
h2 → h1 h3 h4 h5
h3 → h1 h2 h4 h5
h4 → h1 h2 h3 h5
h5 → h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet>
```






```
ubuntu@docker:~/e-sdn$ sudo python3 automate-topology.py
Enter topology type (single, linear, tree): tree
Enter depth of tree: 3
Enter fanout of tree: 2
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 → h2 h3 h4 h5 h6 h7 h8
h2 → h1 h3 h4 h5 h6 h7 h8
h3 → h1 h2 h4 h5 h6 h7 h8
h4 → h1 h2 h3 h5 h6 h7 h8
h5 → h1 h2 h3 h4 h6 h7 h8
h6 → h1 h2 h3 h4 h5 h7 h8
h7 → h1 h2 h3 h4 h5 h6 h8
h8 → h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>
```