ELEC-E7311 - SDN Fundamentals & Techniques

Demo 1: Mininet - as an SDN emulator

Md Shawkot Hossain

Submission period: Spring, 2024

Task 1:

Default controller:

Single topology:

```
sudo mn --topo single,10
```

Linear topology:

```
sudo mn –topo=linear,5
```

Tree topology:

```
sudo mn --topo tree,3,2 # creating a topology with depth 3 and every switch has two child nodes.
```

ONOS controller:

Single topology:

```
sudo mn --topo single,10 –controller=remote,ip=127.0.0.1,port=6653 --switch ovs,protocols=OpenFlow13
```

Linear topology:

```
sudo mn --topo=linear,5 --controller=remote,ip=127.0.0.1,port=6653 --switch ovs,protocols=OpenFlow13
```

Tree topology:

```
sudo mn --topo tree,3,2 --controller=remote,ip=127.0.0.1,port=6653 --switch ovs,protocols=OpenFlow13
```

Task 2:

Single switch:

To creae this topology, I used mininet module, RemoteController to use ONOS controller. I also specified the OpenFlow 1.3 version when creating the switch, for example, “s1 = net.addSwitch('s1', protocols='OpenFlow13')” To simplify the code, i.e. creating 13 hosts, I used list comprehension and short notation of ‘for’ loop. Apart from that, I also used CLI module for command line, setLogLevel, info to make the network creation verbose.

see the next page.

```
#!/usr/bin/python3

from mininet.net import Mininet

from mininet.node import OVSKernelSwitch, RemoteController

from mininet.cli import CLI

from mininet.log import setLogLevel, info

def create_single_switch_topology():

    net = Mininet(controller=RemoteController, switch=OVSKernelSwitch)

    # Add a single ONOS controller

    c1 = net.addController('c1', controller=RemoteController, ip="127.0.0.1", port=6653)

    # Create a single switch

    s1 = net.addSwitch('s1', protocols='OpenFlow13')

    # Create 13 hosts

    num_hosts = 13

    hosts = [net.addHost(f'h{i+1}') for i in range(num_hosts)]

    # Connect hosts to the switch

    for host in hosts:

        net.addLink(host, s1)

    # Start controller and switch

    net.build()

    c1.start()

    s1.start([c1])

    # Start network

    net.start()

    # Open Mininet CLI

    CLI(net)

    # Stop network
```

Linear Topology

For this topology, I used almost the same module as before. To connect every host to switch, I used nested for loop. And then used another for loop to connect the switches in a linear fashion.

Please see the next page for the code.

```
#!/usr/bin/python3
from mininet.net import Mininet
from mininet.node import OVSKernelSwitch, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel, info
def create_linear_topology():
    net = Mininet(controller=RemoteController, switch=OVSKernelSwitch)
    # Add a single ONOS controller
    c1 = net.addController('c1', controller=RemoteController, ip="127.0.0.1", port=6653)
    # Create switches
    num_switches = 10
    switches = [net.addSwitch(f's{s}', protocols='OpenFlow13') for s in range(1,
num_switches + 1)]
    # Create hosts
    num_hosts_per_switch = 1
    hosts = [net.addHost(f'h{h}') for h in range(1, num_switches * num_hosts_per_switch +
1)]
    # Connect hosts to switches
    for i, switch in enumerate(switches):
        for j in range(num_hosts_per_switch):
            net.addLink(hosts[i * num_hosts_per_switch + j], switch)
    # Connect switches in a linear topology
    for i in range(num_switches - 1):
        net.addLink(switches[i], switches[i + 1])
    # Start controller and switches
    net.build()
    c1.start()
    for sw in switches:
        sw.start([c1])
    # Start network
```

Tree Topology:

A topology of depth 3 and fanout 2 means a network where the level is 3 and each switch has two childs except the leaf. The following code achieves the mentioned network topology and uses the same module as before.

Please see the next page for the code.

```
#!/usr/bin/python3
from mininet.net import Mininet
from mininet.node import OVSKernelSwitch, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel, info
def create_tree_topology():
    net = Mininet(controller=RemoteController, switch=OVSKernelSwitch)
    # Add a single ONOS controller
    c1 = net.addController('c1', controller=RemoteController, ip="127.0.0.1", port=6653)
    # Create the tree topology
    depth = 3
    fanout = 2
    create_tree(net, depth, fanout)
    # Start controller and switches
    net.build()
    c1.start()
    for switch in net.switches:
        switch.start([c1])
    # Start network
    net.start()
    # Open Mininet CLI
    CLI(net)
    # Stop network
    net.stop()
def create_tree(net, depth, fanout):
    if depth == 0:
        return
    switch = net.addSwitch(f's{depth}', protocols='OpenFlow13')
```