SDN Fundamentals & Techniques

Demo: Mininet – as a SDN emulator

Jing Yan (yanj3, jing.yan@aalto.fi) Feb 2nd, 2020

Task 1: Create using the CLI interface and the OpenFlow Reference Controller

Q1: A single-switch topology with 10 hosts - Commands

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

```
root@mininet-vm:~# mn --topo single,10
*** 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: c0
*** Starting 1 switches: s1 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=2062>
...
<Host h10: h10-eth0:10.0.0.10 pid=2080>
<OVSSwitch s1:
lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-et
h6:None,s1-eth7:None,s1-eth8:None,s1-eth9:None,s1-eth10:None pid=2085>
<Controller c0: 127.0.0.1:6653 pid=2055>
mininet>
```

Q2: A linear topology of 5 switches and 5 hosts - Commands

```
root@mininet-vm:~# mn --topo linear,5
*** 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: c0
*** Starting 5 switches: s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=2690>
...
<Host h5: h5-eth0:10.0.0.5 pid=2698>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=2703>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=2706>
...
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=2712>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None pid=2715>
<Controller c0: 127.0.0.1:6653 pid=2683>
mininet>
```

Q3: A tree topology depth 3 fanout 2 - Commands

```
root@mininet-vm:~# mn --topo tree,depth=3,fanout=2
*** 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: c0
*** Starting 7 switches: s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3173>
...
<Host h8: h8-eth0:10.0.0.8 pid=3187>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=3192>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=3195>
...
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None pid=3210>
<Controller c0: 127.0.0.1:6653 pid=3166>
```

```
mininet>
```

Q4: Brief Explanation Part for Q1, Q2, Q3

First of all, OpenFlow Reference Controller is integrated with Mininet, so all the configuration just happens among the mininet VM. (In my environment, I deploy mininet through importing a given mininet VM image.)

- So for Q1, use “mn --topo single, 10” to create a single-switch topology with 10 hosts, and “mininet> dump” to display summary information about all devices. In this case, 10 hosts are connected to the switch.
- So for Q2, use “mn --topo linear,5” to create a linear topology of 5 switches and 5 hosts and “mininet> dump” to display summary information about all devices. In this case, each switch has one host, and all switches connect in a line.
- So for Q3, use “mn --topo tree,depth=3,fanout=2” to create a tree topology depth 3 fanout 2 and “mininet> dump” to display summary information about all devices. In this case, there are 7 switches and 8 hosts, and we can see the tree topology through the “Adding links” log part of “mn --topo tree,depth=3,fanout=2” command.

Q5: Repeat the same operations using the remote SDN controller installed before

i.e., ONOS

Q5.0 Environment description

I created ONOS VM and Mininet VM with the bridge network mode on my local physical machine, displaying ip info of ONOS VM, Mininet VM and my physical local machine as follows:

- Local Physical Machine:
```
yanjing@yanjingdeMacBook-Pro ~ % ifconfig en0 | grep inet
   inet 192.168.1.101
```
- Mininet VM based on VirtualBox:
```
root@mininet-vm:~# ifconfig -a | grep inet
       inet 192.168.1.109
```
- ONOS VM based on VirtualBox:
```
root@jingyan:/home/jingyan/onos# ifconfig -a | grep inet
       inet 192.168.1.112

In the 1st terminal:
root@jingyan:/home/jingyan/onos# bazel run onos-local clean debug

In the 2nd terminal:
root@jingyan:/home/jingyan/onos# ./tools/test/bin/onos localhost
```

Q5.1 A single-switch topology with 10 hosts - Commands

```
root@mininet-vm:~# mn --controller remote,ip=192.168.1.112,port=9876 --topo
single,10
*** 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: c0
*** Starting 1 switches: s1 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5254>
...
<Host h10: h10-eth0:10.0.0.10 pid=5272>
<OVSSwitch s1:
lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-et
h6:None,s1-eth7:None,s1-eth8:None,s1-eth9:None,s1-eth10:None pid=5277>
<RemoteController{'ip': '192.168.1.112', 'port': 9876} c0: 192.168.1.112:9876
pid=5248>
mininet>
```

Q5.2 A linear topology of 5 switches and 5 hosts - Commands

```
root@mininet-vm:~# mn --controller remote,ip=192.168.1.112,port=9876 --topo
linear,5
*** 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: c0
*** Starting 5 switches: s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5636>
...
<Host h5: h5-eth0:10.0.0.5 pid=5644>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=5649>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=5652>
...
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=5658>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None pid=5661>
<RemoteController{'ip': '192.168.1.112', 'port': 9876} c0: 192.168.1.112:9876
pid=5630>
mininet>
```

Q5.3 A tree topology depth 3 fanout 2 - Commands

```
root@mininet-vm:~# mn --controller remote,ip=192.168.1.112,port=9876 --topo
tree,depth=3,fanout=2
*** 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: c0
*** Starting 7 switches: s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=6016>
...
<Host h8: h8-eth0:10.0.0.8 pid=6030>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=6035>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=6038>
...
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None pid=6053>
<RemoteController{'ip': '192.168.1.112', 'port': 9876} c0: 192.168.1.112:9876
pid=6010>
mininet>
```

Q5.4 Brief Explanation Part for Q5.1, Q5.2, Q5.3

Compared to Q4: Brief Explanation Part for Q1, Q2, Q3 part, I added "--controller remote,ip=192.168.1.112,port=9876" based on my environment setting for the solutions of Q5.1, Q5.2 and Q5.3.

Task 2: Create using the Python API interface and the ONOS SDN Controller

Q1: A single-switch topology with 13 hosts

Note that a "single-switch topology" means one switch connected to all hosts.

Code

```
#!/usr/bin/python
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.node import Host, RemoteController
if __name__ == '__main__':
   net=Mininet(topo=None, build=False)
   c0=net.addController(name='c0', controller=RemoteController, ip='192.168.1.112',
protocol='tcp', port=9876)
   s1=net.addSwitch('s1')
   for h in range(1,14):
       host=net.addHost("h%s" % h)
       net.addLink(host, s1)
   net.build()
   c0.start()
   net.get('s1').start([c0])
   CLI(net)
   net.stop()
```

Result

```
root@mininet-vm:~# python3 test1.py
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=9575>
...
<Host h13: h13-eth0:10.0.0.13 pid=9727>
<OVSSwitch s1:
lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-et
h6:None,s1-eth7:None,s1-eth8:None,s1-eth9:None,s1-eth10:None,s1-eth11:None,s1-eth12
:None,s1-eth13:None pid=9570>
<RemoteController c0: 192.168.1.112:9876 pid=9563>
mininet> links
h1-eth0<->s1-eth1 (OK OK)
...
h13-eth0<->s1-eth13 (OK OK)
mininet>
```

Q2: A linear topology of 10 switches and 10 hosts

Code

```
#!/usr/bin/python
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.node import Host, RemoteController
if __name__ == '__main__':
   net = Mininet(topo=None, build=False)
   c0=net.addController(name='c0', controller=RemoteController, ip='192.168.1.112', 
protocol='tcp', port=9876)
   for index in range(1,11):
       switch=net.addSwitch('s%s' % index)
       host=net.addHost("h%s" % index)
       net.addLink(switch, host)
   for index in range(1,10):
       net.addLink('s%s' % (index + 1), 's%s' % index)
   net.build()
   c0.start()
   for index in range(1,11):
       net.get('s%s'% index).start([c0])
   CLI(net)
   net.stop()
```

Result

```
root@mininet-vm:~# python3 test1.py
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=11243>
...
<Host h10: h10-eth0:10.0.0.10 pid=11387>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=11238>
...
<OVSSwitch s10: lo:127.0.0.1,s10-eth1:None,s10-eth2:None pid=11380>
<RemoteController c0: 192.168.1.112:9876 pid=11231>
mininet> links
s1-eth1<->h1-eth0 (OK OK)
s2-eth1<->h2-eth0 (OK OK)
...
s9-eth1<->h9-eth0 (OK OK)
s10-eth1<->h10-eth0 (OK OK)
s2-eth2<->s1-eth2 (OK OK)
s3-eth2<->s2-eth3 (OK OK)
...
s9-eth2<->s8-eth3 (OK OK)
s10-eth2<->s9-eth3 (OK OK)
mininet>
```

Q3: A tree topology depth 3 fanout 2

Code

```
#!/usr/bin/python
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.node import OVSSwitch
from mininet.topolib import TreeNet
from mininet.examples.treeping64 import HostV4

if __name__ == '__main__':
   network = TreeNet( depth=3, fanout=2, host=HostV4, switch=OVSSwitch, 
waitConnected=True)
   network.run( CLI, network )
```

Result

```
root@mininet-vm:~# python3 test1.py
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=12114>
...
<Host h8: h8-eth0:10.0.0.8 pid=12128>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=12091>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=12094>
...
```

```
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None pid=12109>
<RemoteController c0: 192.168.1.112:9876 pid=12084>
mininet> links
s1-eth1<->s2-eth1 (OK OK)
s1-eth2<->s5-eth1 (OK OK)
s2-eth2<->s3-eth1 (OK OK)
s2-eth3<->s4-eth1 (OK OK)
s5-eth2<->s6-eth1 (OK OK)
s5-eth3<->s7-eth1 (OK OK)
s3-eth2<->h1-eth0 (OK OK)
s3-eth3<->h2-eth0 (OK OK)
s4-eth2<->h3-eth0 (OK OK)
s4-eth3<->h4-eth0 (OK OK)
s6-eth2<->h5-eth0 (OK OK)
s6-eth3<->h6-eth0 (OK OK)
s7-eth2<->h7-eth0 (OK OK)
s7-eth3<->h8-eth0 (OK OK)
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

Code

```
#!/usr/bin/python
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.node import Host, RemoteController, OVSSwitch
from mininet.topolib import TreeNet
from mininet.examples.treeping64 import HostV4

net=Mininet(topo=None, build=False)
c0=net.addController(name='c0',   controller=RemoteController,   ip='192.168.1.112',
protocol='tcp', port=9876)

def single_switch(single_switch_host_num):
  global net
  global c0
  s1=net.addSwitch('s1')
  for index in range(0,single_switch_host_num):
      host=net.addHost("h%s" % (index+1))
      net.addLink(host, s1)
  net.build()
  c0.start()
  net.get('s1').start([c0])
  CLI(net)
  net.stop()

def linear(linear_host_num):
   global net
   global c0
   for index in range(0,linear_host_num):
       switch=net.addSwitch('s%s' % (index+1))
       host=net.addHost("h%s" % (index+1))
       net.addLink(switch, host)
   for index in range(0,linear_host_num-1):
      net.addLink('s%s' % (index+2), 's%s' % (index+1))
   net.build()
   c0.start()
   for index in range(0,linear_host_num):
       net.get('s%s'% (index+1)).start([c0])
   CLI(net)
   net.stop()
```

```
def tree(tree_depth, tree_fanout):
        network  =  TreeNet(  depth=tree_depth,   fanout=tree_fanout,   host=HostV4,
switch=OVSSwitch, waitConnected=True)
   network.run( CLI, network )
   net.stop()

if __name__ == '__main__':
   topo_type = input("Pls input Topo type (single-switch or linear or tree): ")
   if topo_type == 'single-switch':
        single_switch_host_num = input("Pls input host num for single-switch topo:
")
       single_switch(int(single_switch_host_num))
   elif topo_type == 'linear':
       linear_host_num = input("Pls input switch/host num for linear topo: ")
       linear(int(linear_host_num))
   elif topo_type == 'tree':
       tree_depth = input("Pls input depth for tree topo: ")
       tree_fanout = input("Pls input fanout for tree topo: ")
       tree(int(tree_depth), int(tree_fanout))
   else:
       print("Wrong topo_type")
```

Result - Test single-switch Topology

```
root@mininet-vm:~# python3 test1.py
Pls input Topo type (single-switch or linear or tree): single-switch
Pls input host num for single-switch topo: 5
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
h3-eth0<->s1-eth3 (OK OK)
h4-eth0<->s1-eth4 (OK OK)
h5-eth0<->s1-eth5 (OK OK)
mininet>
```

Result - Test linear Topology

```
root@mininet-vm:~# python3 test1.py
Pls input Topo type (single-switch or linear or tree): linear
Pls input switch/host num for linear topo: 5
mininet> links
s1-eth1<->h1-eth0 (OK OK)
s2-eth1<->h2-eth0 (OK OK)
s3-eth1<->h3-eth0 (OK OK)
s4-eth1<->h4-eth0 (OK OK)
s5-eth1<->h5-eth0 (OK OK)
s2-eth2<->s1-eth2 (OK OK)
s3-eth2<->s2-eth3 (OK OK)
s4-eth2<->s3-eth3 (OK OK)
s5-eth2<->s4-eth3 (OK OK)
mininet>
```

Result - Test tree Topology

```
root@mininet-vm:~# python3 test1.py
Pls input Topo type (single-switch or linear or tree): tree
Pls input depth for tree topo: 2
Pls input fanout for tree topo: 4
mininet> links
s1-eth1<->s2-eth5 (OK OK)
s1-eth2<->s3-eth5 (OK OK)
s1-eth3<->s4-eth5 (OK OK)
s1-eth4<->s5-eth5 (OK OK)
s2-eth1<->h1-eth0 (OK OK)
s2-eth2<->h2-eth0 (OK OK)
s2-eth3<->h3-eth0 (OK OK)
```

```
s2-eth4<->h4-eth0 (OK OK)
s3-eth1<->h5-eth0 (OK OK)
s3-eth2<->h6-eth0 (OK OK)
s3-eth3<->h7-eth0 (OK OK)
s3-eth4<->h8-eth0 (OK OK)
s4-eth1<->h9-eth0 (OK OK)
s4-eth2<->h10-eth0 (OK OK)
s4-eth3<->h11-eth0 (OK OK)
s4-eth4<->h12-eth0 (OK OK)
s5-eth1<->h13-eth0 (OK OK)
s5-eth2<->h14-eth0 (OK OK)
s5-eth3<->h15-eth0 (OK OK)
s5-eth4<->h16-eth0 (OK OK)
mininet>
```