Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 06

Course code: ICT-4202

Course title: Wireless and Mobile Communication Lab

Date of Performance: 25.09.2020

Date of Submission: 30.09.2020

Submitted by

Name: Raisa Jerin Sristy

ID: IT-16056

4th year 2nd semester

Session: 2015-2016

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

LAB NO.:06

Name of Experiment: **Switching a interface to move a host around a network using mininet.**

Objectives:

1. To manage the flows in a switch manually.
2. To manage the flow entries on a Open vSwitch (OVS).
3. To evaluate a network using a graphical interface.
4. To know the functionality of mininet.

Source code:

```
#!/usr/bin/python

"""
Simple example of Mobility with Mininet
(aka enough rope to hang yourself.)

We move a host from s1 to s2, s2 to s3, and
then back to s1.

Gotchas:

The reference controller doesn't support
mobility, so we need to
manually flush the switch flow tables!

Good luck!

to-do:

- think about wifi/hub behavior
- think about clearing last hop - why doesn't
that work?
"""

from mininet.net import Mininet
from mininet.node import OVSSwitch
from mininet.topo import LinearTopo
from mininet.log import output, warn
from random import randint
```

```
class MobilitySwitch( OVSSwitch ):
    "Switch that can reattach and rename interfaces"

    def delIntf( self, intf ):
        "Remove (and detach) an interface"
        port = self.ports[ intf ]
        del self.ports[ intf ]
        del self.intfs[ port ]
        del self.nameToIntf[ intf.name ]

    def addIntf( self, intf, rename=False, **kwargs ):
        "Add (and reparent) an interface"
        OVSSwitch.addIntf( self, intf, **kwargs )
        intf.node = self
        if rename:
            self.renameIntf( intf )

    def attach( self, intf ):
        "Attach an interface and set its port"
        port = self.ports[ intf ]
        if port:
            if self.isOldOVS():
                self.cmd( 'ovs-vsctl add-port', self, intf )
            else:
                self.cmd( 'ovs-vsctl add-port', self, intf,
                        '-- set Interface', intf,
                        'ofport_request=%s' % port )
            self.validatePort( intf )

    def validatePort( self, intf ):
        "Validate intf's OF port number"
        ofport = int( self.cmd( 'ovs-vsctl get Interface', intf,
                            'ofport' ) )
        if ofport != self.ports[ intf ]:
            warn( 'WARNING: ofport for', intf, 'is actually', ofport,
                '\n' )

    def renameIntf( self, intf, newname='' ):
        "Rename an interface (to its canonical name)"
        intf.ifconfig( 'down' )
        if not newname:
            newname = '%s-eth%d' % ( self.name, self.ports[ intf ] )
        intf.cmd( 'ip link set', intf, 'name', newname )
        del self.nameToIntf[ intf.name ]
        intf.name = newname
        self.nameToIntf[ intf.name ] = intf
        intf.ifconfig( 'up' )
```

```
    def moveIntf( self, intf, switch,
port=None, rename=True ):
        "Move one of our interfaces to another
switch"
        self.detach( intf )
        self.delIntf( intf )
        switch.addIntf( intf, port=port,
rename=rename )
        switch.attach( intf )



def printConnections( switches ):
    "Compactly print connected nodes to each
switch"
    for sw in switches:
        output( '%s: ' % sw )
        for intf in sw.intfList():
            link = intf.link
            if link:
                intf1, intf2 = link.intf1, link.intf2
                remote = intf1 if intf1.node != sw
else intf2
                output( '%s(%s) ' % ( remote.node,
sw.ports[ intf ] ) )
        output( '\n' )



def moveHost( host, oldSwitch, newSwitch,
newPort=None ):
    "Move a host from old switch to new
switch"
    hintf, sintf = host.connectionsTo(
oldSwitch )[ 0 ]
    oldSwitch.moveIntf( sintf, newSwitch,
port=newPort )
    return hintf, sintf



def mobilityTest():
    "A simple test of mobility"
    print '* Simple mobility test'
    net = Mininet( topo=LinearTopo( 3 ),
switch=MobilitySwitch )
    print '* Starting network:'
    net.start()
    printConnections( net.switches )
    print '* Testing network'
    net.pingAll()
    print '* Identifying switch interface for h1'
    h1, old = net.get( 'h1', 's1' )
    for s in 2, 3, 1:
        new = net[ 's%d' % s ]
        port = randint( 10, 20 )
        print '* Moving', h1, 'from', old, 'to',
new, 'port', port
        hintf, sintf = moveHost( h1, old, new,
newPort=port )
        print '*', hintf, 'is now connected to',
sintf
        print '* Clearing out old flows'
        for sw in net.switches:
```

```
    sw.dpctl( 'del-flows' )
    print '* New network:'
    printConnections( net.switches )
    print '* Testing connectivity:'
    net.pingAll()
    old = new
    net.stop()

if __name__ == '__main__':
    mobilityTest()
```

Output:

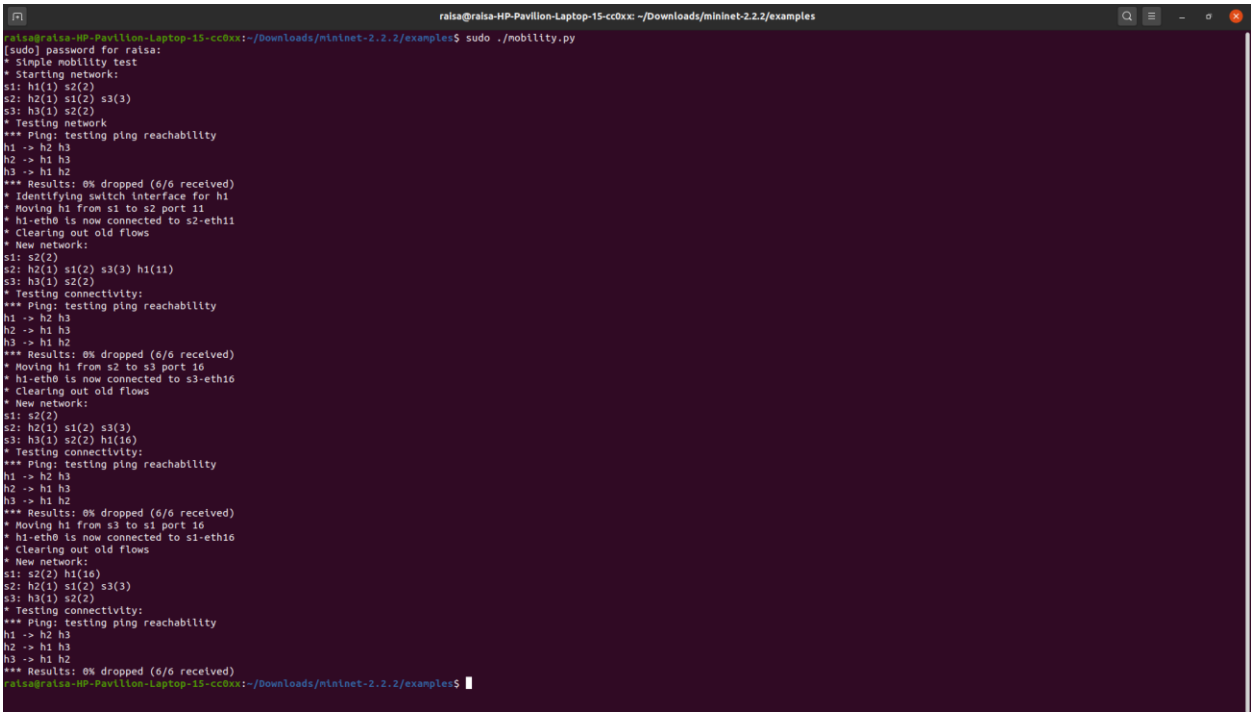

Figure: Output Screenshot of mobility.py

Conclusion:

Here in this experiment, using mininet , switching an interface to move a host around a network using mininet was successfully done and the achieved output displays how an interface is switch around a network on applying mininet.