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: Shourove Sutradhar Dip

ID: IT-16008

4th year 2nd semester

Session: 2015-2016

Dept. of ICT, MBSTU

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment N0: 06

Name of Experiment: Switching an interface to move a host around a network using Mininet.

Objectives:

1. Learn how to detach and attach an interface.
2. Understand how hosts, links and switches work on a single machine.
3. Learn to use Mininet for interactive development, testing and demos.
4. Detach an interface from one switch and attaching to another as a basic way to move a host around a network.

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 info, output, warn, setLogLevel

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 ):
```

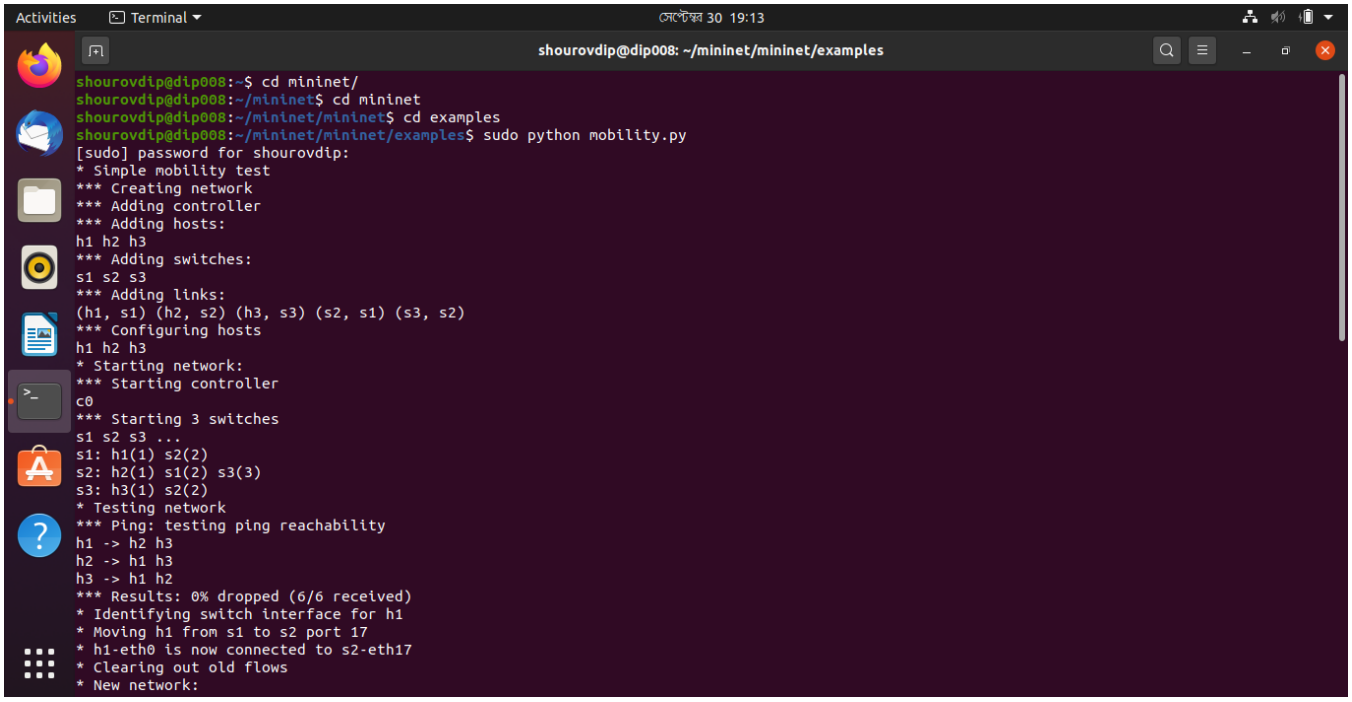
```
    "Attach an interface and set its port"
    port = self.ports[ intf ]
    if port:
        if self.isOldOVS():
            self.cmd( 'ovs-vsctl add-port', self, intf )
        else:
            self.cmd( 'ovs-vsctl add-port', self, intf,
                      '-- set Interface', intf,
       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

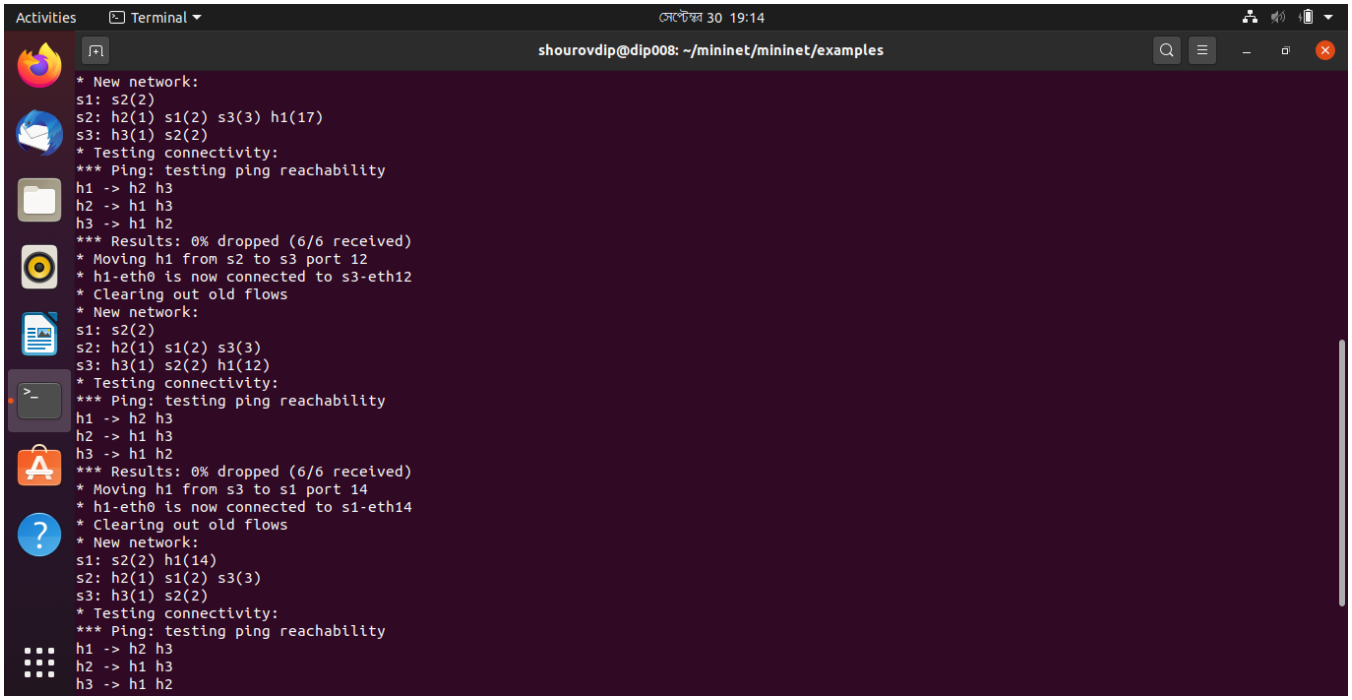
```
        intf.name = newname
        self.nameToIntf[ intf.name ] = intf
        intf.ifconfig( 'up' )

    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 ):
```

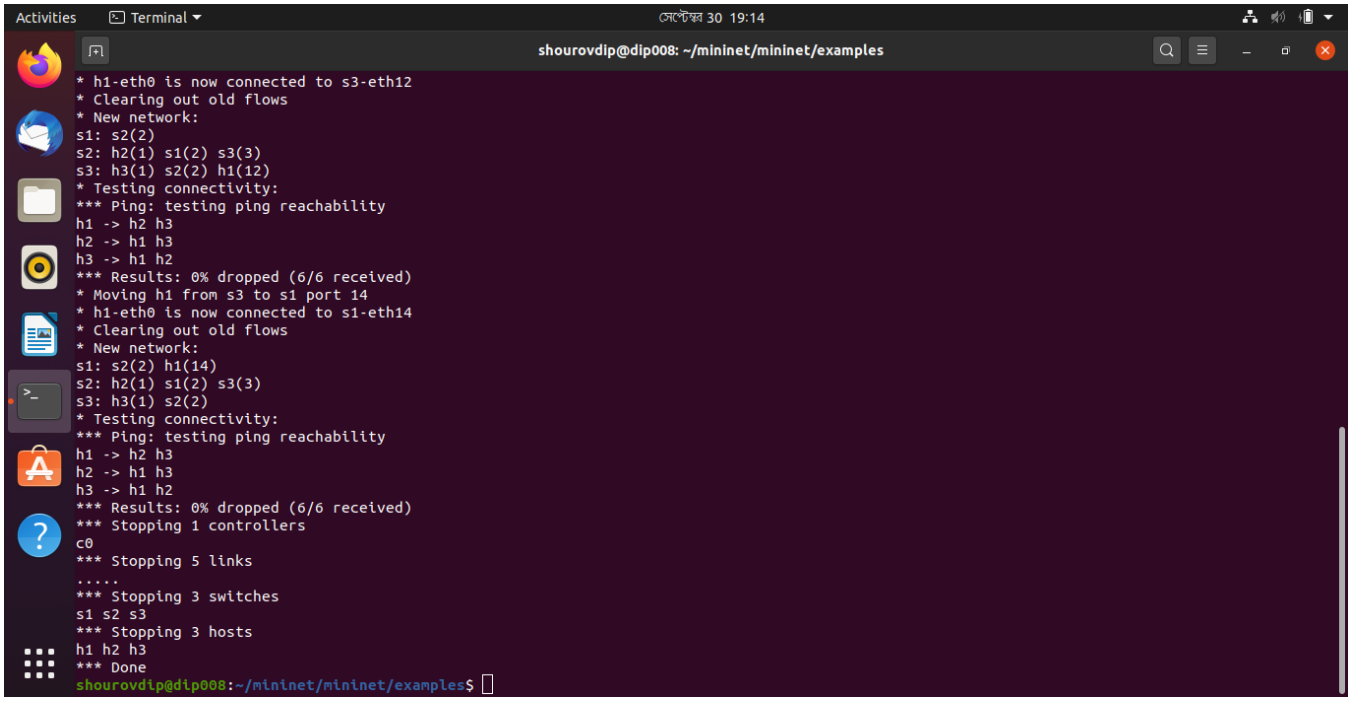
```
    "Move a host from old switch to new switch"
    hintf, sintf = host.connectionsTo( oldSwitch )[ 0 ]
    oldSwitch.moveIntf( sintf, newSwitch, port=newPort )
    return hintf, sintf


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

```
            info( '* New network:\n' )
            printConnections( net.switches )
            info( '* Testing connectivity:\n' )
            net.pingAll()
            old = new
        net.stop()

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

Output:


Activities
Terminal
সেপ্টেম্বর 30 19:13
shourovdip@dip008: ~/mininet/mininet/examples
shourovdip@dip008:~$ cd mininet/
shourovdip@dip008:~/mininet$ cd mininet
shourovdip@dip008:~/mininet/mininet$ cd examples
shourovdip@dip008:~/mininet/mininet/examples$ sudo python mobility.py
[sudo] password for shourovdip:
* Simple mobility test
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
* Starting network:
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
s1: h1(1) s2(2)
s2: h2(1) s1(2) s3(3)
s3: h3(1) s2(2)
* Testing network
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
* Identifying switch interface for h1
* Moving h1 from s1 to s2 port 17
* h1-eth0 is now connected to s2-eth17
* Clearing out old flows
* New network:


Activities
Terminal
সেপ্টেম্বর 30 19:14
shourovdip@dip008: ~/mininet/mininet/examples
* New network:
s1: s2(2)
s2: h2(1) s1(2) s3(3) h1(17)
s3: h3(1) s2(2)
* Testing connectivity:
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
* Moving h1 from s2 to s3 port 12
* h1-eth0 is now connected to s3-eth12
* Clearing out old flows
* New network:
s1: s2(2)
s2: h2(1) s1(2) s3(3)
s3: h3(1) s2(2) h1(12)
* Testing connectivity:
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
* Moving h1 from s3 to s1 port 14
* h1-eth0 is now connected to s1-eth14
* Clearing out old flows
* New network:
s1: s2(2) h1(14)
s2: h2(1) s1(2) s3(3)
s3: h3(1) s2(2)
* Testing connectivity:
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2



Conclusion:

From this lab, we've learnt how to detach an interface from one switch and attach it to another one. It is a basic way to move a host around a network. We also learnt to use Mininet which creates virtual networks using process-based virtualization and network namespaces. In Mininet, hosts are emulated as bash processes running in a network namespace, so any code that would normally run on a Linux server (like a web server or client program) should run just fine within a Mininet "Host".