import os

from mininet.net import Mininet

from mininet.topo import Topo

from mininet.log import setLogLevel, info

from mininet.cli import CLI

from mininet.node import RemoteController

from p4\_mininet import P4Switch, P4Host

import argparse

from time import sleep

parser = argparse.ArgumentParser(description='Mininet demo')

parser.add\_argument('--behavioral-exe', help='Path to behavioral executable',

type=str, action="store", required=False, default='simple\_switch' )

parser.add\_argument('--thrift-port', help='Thrift server port for table updates',

type=int, action="store", default=9090)

parser.add\_argument('--num-hosts', help='Number of hosts to connect to switch',

type=int, action="store", default=2)

parser.add\_argument('--mode', choices=['l2', 'l3'], type=str, default='l3')

parser.add\_argument('--json', help='Path to JSON config file',

type=str, action="store", required=True)

parser.add\_argument('--pcap-dump', help='Dump packets on interfaces to pcap files',

type=str, action="store", required=False, default=False)

args = parser.parse\_args()

class SingleSwitchTopo(Topo):

def \_\_init\_\_(self, sw\_path, json\_path, thrift\_port, pcap\_dump, \*\*opts):

Topo.\_\_init\_\_(self, \*\*opts)

switch3 = self.addSwitch('s3', sw\_path = sw\_path, json\_path = json\_path, thrift\_port = thrift\_port,cls = P4Switch ,pcap\_dump = pcap\_dump)

host1 = self.addHost('h1', mac = '00:00:00:00:00:01', ip="192.168.10.1/24")

host2 = self.addHost('h2', mac = '00:00:00:00:00:02', ip="192.168.20.1/24")

host4 = self.addHost('s4')

host5 = self.addHost('s5')

self.addLink(host1, switch3, port1 = 0, port2 = 1)

self.addLink(host2, host5, port1 = 0, port2 = 0)

self.addLink(switch3, host4, port1 = 2, port2 = 0)

self.addLink(host4, host5, port1 = 1, port2 = 1)

def main():

topo = SingleSwitchTopo(args.behavioral\_exe, args.json, args.thrift\_port, args.pcap\_dump)

net = Mininet(topo = topo, host = P4Host, controller = None)

net.start()

h1,h2=net.get('h1','h2');

s4,s5=net.get('s4','s5');

s3=net.get('s3');

h1.cmd("ip route add default via 192.168.10.254")

h2.cmd("ip route add default via 192.168.20.254")

s3.cmd("ifconfig s3-eth1 down")

s3.cmd("ifconfig s3-eth1 hw ether 00:00:00:00:00:03")

s3.cmd("ifconfig s3-eth1 up")

s3.cmd("ifconfig s3-eth2 down")

s3.cmd("ifconfig s3-eth2 hw ether 00:00:00:00:00:06")

s3.cmd("ifconfig s3-eth2 up")

s5.cmd("ifconfig eth0 down")

s5.cmd("ifconfig eth0 hw ether 00:00:00:00:00:04")

s5.cmd("ifconfig eth0 up")

s4.cmd("ifconfig eth0 down")

s4.cmd("ifconfig eth0 hw ether 00:00:00:00:00:05")

s4.cmd("ifconfig eth0 up")

s4.cmd("ifconfig eth0 0")

s4.cmd("ifconfig h4-eth1 0")

s4.cmd("ip addr add 34.1.1.4/24 brd + dev eth0")

s4.cmd("ip addr add 45.1.1.4/24 brd + dev s4-eth1")

s4.cmd("echo 1 > /proc/sys/net/ipv4/ip\_forward")

s4.cmd("arp -s 34.1.1.3 00:00:00:00:00:06")

s5.cmd("ifconfig eth0 0")

s5.cmd("ifconfig h5-eth1 0")

s5.cmd("ip addr add 192.168.20.254/24 brd + dev eth0")

s5.cmd("ip addr add 45.1.1.5/24 brd + dev s5-eth1")

s5.cmd("echo 1 > /proc/sys/net/ipv4/ip\_forward")

h1.cmd("arp -s 192.168.10.254 00:00:00:00:00:03")

h2.cmd("arp -s 192.168.20.254 00:00:00:00:00:04")

s5.cmd("ip tunnel add neta mode gre remote 34.1.1.3 local 45.1.1.5 ttl 255")

s5.cmd("ip addr add 12.1.1.2/24 dev neta")

s5.cmd("ifconfig neta up")

s5.cmd("ip route add 192.168.10.0/24 via 12.1.1.2")

s5.cmd("ip route add default via 45.1.1.4")

sleep(1)

print('\033[0;32m'),

print "Gotcha!"

print('\033[0m')

CLI(net)

try:

net.stop()

except:

print('\033[0;31m'),

print('Stop error! Trying sudo mn -c')

print('\033[0m')

os.system('sudo mn -c')

print('\033[0;32m'),

print ('Stop successfully!')

print('\033[0m')

if \_\_name\_\_ == '\_\_main\_\_':

setLogLevel('info')

main()