|  |  |
| --- | --- |
| Logo  Description automatically generated | THỰC HÀNH SDN  Static routing |

Nhóm học phần: 20Nh10

1) Mã Sinh viên, Họ và Tên: 102200013, Nguyễn Công Cường

MỤC LỤC

[1. Introduction 1](#_Toc177629144)

[2. Run python mininet program (2 ways) 1](#_Toc177629145)

[2.1. Python program using RemoteController 1](#_Toc177629146)

[2.2. Python program without RemoteController 2](#_Toc177629147)

[3. Static routing 2](#_Toc177629148)

[3.1. File mystaticrouting01.py 2](#_Toc177629149)

[3.2. Run program 4](#_Toc177629150)

[4. Bài tập 6](#_Toc177629151)

[5. Tham khảo 6](#_Toc177629152)

*>> Yêu cầu chụp hình ảnh là kết quả thực hành của SV. Không sử dụng lại hình ảnh của bài lab.*

# Introduction

# Run python mininet program (2 ways)

## Python program using RemoteController

* File example01.py

from mininet.topo import Topo

from mininet.net import Mininet

from mininet.cli import CLI

from mininet.node import RemoteController

class CustomTopo (Topo):

def build(self):

S1 = self.addSwitch('s1')

H1 = self.addHost('h1')

H2 = self.addHost('h2')

self.addLink(S1, H1)

self.addLink(S1, H2)

topo = CustomTopo()

net = Mininet(topo, controller=lambda name: RemoteController(name, ip='127.0.0.1', protocol='tcp', port = 6633), autoSetMacs=True)

net.start()

CLI(net)

net.stop()

* Run program:

$ sudo python example01.py

## Python program without RemoteController

* File example02.py

from mininet.topo import Topo

from mininet.net import Mininet

from mininet.cli import CLI

class CustomTopo (Topo):

def build(self):

S1 = self.addSwitch('s1')

H1 = self.addHost('h1')

H2 = self.addHost('h2')

self.addLink(S1, H1)

self.addLink(S1, H2)

topo = CustomTopo()

topos = {'mytopo': CustomTopo}

* Run program:

$ sudo mn --custom example01.py --topo=mytopo --mac --controller=remote,ip=127.0.0.1,port=6633

# Static routing

## File mystaticrouting01.py

#!/usr/bin/python

from mininet.topo import Topo

from mininet.net import Mininet

from mininet.node import Node

from mininet.node import RemoteController

from mininet.log import setLogLevel, info

from mininet.cli import CLI

class LinuxRouter(Node):

def config(self, \*\*params):

super(LinuxRouter, self).config(\*\*params)

self.cmd('sysctl net.ipv4.ip\_forward=1')

def terminate(self):

self.cmd('sysctl net.ipv4.ip\_forward=0')

super(LinuxRouter, self).terminate()

class NetworkTopo(Topo):

def build(self, \*\*\_opts):

# Add 4 routers in four different subnets

r1 = self.addHost('r1', cls=LinuxRouter, ip='10.0.0.1/24')

r2 = self.addHost('r2', cls=LinuxRouter, ip='10.1.0.1/24')

r3 = self.addHost('r3', cls=LinuxRouter, ip='10.2.0.1/24')

r4 = self.addHost('r4', cls=LinuxRouter, ip='10.3.0.1/24')

# Add 2 switches

s1 = self.addSwitch('s1')

s2 = self.addSwitch('s2')

s3 = self.addSwitch('s3')

s4 = self.addSwitch('s4')

# Add host-switch links in the same subnet

self.addLink(s1,

r1,

intfName2='r1-eth1',

params2={'ip': '10.0.0.1/24'})

self.addLink(s2,

r2,

intfName2='r2-eth1',

params2={'ip': '10.1.0.1/24'})

self.addLink(s3,

r3,

intfName2='r3-eth1',

params2={'ip': '10.2.0.1/24'})

self.addLink(s4,

r4,

intfName2='r4-eth1',

params2={'ip': '10.3.0.1/24'})

# Add router-router links in new subnets for the router-router connections

self.addLink(r1,

r2,

intfName1='r1-eth2',

intfName2='r2-eth2',

params1={'ip': '10.100.0.1/24'},

params2={'ip': '10.100.0.2/24'})

self.addLink(r1,

r3,

intfName1='r1-eth3',

intfName2='r3-eth2',

params1={'ip': '10.101.0.1/24'},

params2={'ip': '10.101.0.2/24'})

self.addLink(r3,

r4,

intfName1='r3-eth3',

intfName2='r4-eth2',

params1={'ip': '10.102.0.1/24'},

params2={'ip': '10.102.0.2/24'})

self.addLink(r2,

r4,

intfName1='r2-eth3',

intfName2='r4-eth3',

params1={'ip': '10.103.0.1/24'},

params2={'ip': '10.103.0.2/24'})

# Adding hosts specifying the default route

h1 = self.addHost(name='h1',

ip='10.0.0.10/24',

defaultRoute='via 10.0.0.1')

h2 = self.addHost(name='h2',

ip='10.1.0.10/24',

defaultRoute='via 10.1.0.1')

h3 = self.addHost(name='h3',

ip='10.2.0.10/24',

defaultRoute='via 10.2.0.1')

h4 = self.addHost(name='h4',

ip='10.3.0.10/24',

defaultRoute='via 10.3.0.1')

# Add host-switch links

self.addLink(h1, s1)

self.addLink(h2, s2)

self.addLink(h3, s3)

self.addLink(h4, s4)

def run():

topo = NetworkTopo()

net = Mininet(topo=topo, controller=RemoteController)

# Add routing for reaching networks that aren't directly connected

info(net['r1'].cmd("ip route add 10.1.0.0/24 via 10.100.0.2 dev r1-eth2"))

info(net['r1'].cmd("ip route add 10.2.0.0/24 via 10.101.0.2 dev r1-eth3"))

info(net['r1'].cmd("ip route add 10.3.0.0/24 via 10.101.0.2 dev r1-eth3"))

info(net['r1'].cmd("ip route add 10.3.0.0/24 via 10.100.0.2 dev r1-eth2"))

info(net['r2'].cmd("ip route add 10.0.0.0/24 via 10.100.0.1 dev r2-eth2"))

info(net['r2'].cmd("ip route add 10.3.0.0/24 via 10.103.0.2 dev r2-eth3"))

info(net['r2'].cmd("ip route add 10.2.0.0/24 via 10.103.0.2 dev r2-eth3"))

info(net['r2'].cmd("ip route add 10.2.0.0/24 via 10.100.0.1 dev r2-eth2"))

info(net['r3'].cmd("ip route add 10.0.0.0/24 via 10.101.0.1 dev r3-eth2"))

info(net['r3'].cmd("ip route add 10.3.0.0/24 via 10.102.0.2 dev r3-eth3"))

info(net['r3'].cmd("ip route add 10.1.0.0/24 via 10.102.0.2 dev r3-eth3"))

info(net['r3'].cmd("ip route add 10.1.0.0/24 via 10.101.0.1 dev r3-eth2"))

info(net['r4'].cmd("ip route add 10.1.0.0/24 via 10.103.0.1 dev r4-eth3"))

*info(net['r4'].cmd("ip route add 10.2.0.0/24 via 10.102.0.1 dev r4-eth2"))*

info(net['r4'].cmd("ip route add 10.0.0.0/24 via 10.102.0.1 dev r4-eth2"))

info(net['r4'].cmd("ip route add 10.0.0.0/24 via 10.103.0.1 dev r4-eth3"))

net.start()

CLI(net)

net.stop()

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

setLogLevel('info')

run()

## Run program

$ sudo python mystaticrouting.py

mininet> pingall

\*\*\* Ping: testing ping reachability

h1 -> h2 h3 h4 r1 r2 r3 r4

h2 -> h1 h3 h4 r1 r2 r3 r4

h3 -> h1 h2 h4 r1 r2 r3 r4

h4 -> h1 h2 h3 r1 r2 r3 r4

**r1 -> h1 h2 h3 X r2 r3 X**

r2 -> h1 h2 X h4 r1 X r4

r3 -> h1 X h3 h4 r1 X r4

r4 -> X h2 h3 h4 X r2 r3

\*\*\* Results: 14% dropped (48/56 received)

* r1 has the following route table:

mininet> r1 ip route

10.0.0.0/24 dev r1-eth1 proto kernel scope link src 10.0.0.1

10.1.0.0/24 via 10.100.0.2 dev r1-eth2

10.2.0.0/24 via 10.101.0.2 dev r1-eth3

10.3.0.0/24 via 10.101.0.2 dev r1-eth3 metric 100

10.3.0.0/24 via 10.100.0.2 dev r1-eth2 metric 200

10.100.0.0/24 dev r1-eth2 proto kernel scope link src 10.100.0.1

10.101.0.0/24 dev r1-eth3 proto kernel scope link src 10.101.0.1

Requests to h4 at 10.3.0.10 are going to route via 10.101.0.2 (r3), with a source address of 10.101.0.1.

* r3 has the following route table:

mininet> r3 ip route

10.0.0.0/24 via 10.101.0.1 dev r3-eth2

10.1.0.0/24 via 10.102.0.2 dev r3-eth3 metric 100

10.1.0.0/24 via 10.101.0.1 dev r3-eth2 metric 200

10.2.0.0/24 dev r3-eth1 proto kernel scope link src 10.2.0.1

10.3.0.0/24 via 10.102.0.2 dev r3-eth3

10.101.0.0/24 dev r3-eth2 proto kernel scope link src 10.101.0.2

10.102.0.0/24 dev r3-eth3 proto kernel scope link src 10.102.0.1

So r3 is going to route to 10.3.0.10 via 10.102.0.2. That's r4, which has this route table:

mininet> r4 ip route

10.0.0.0/24 via 10.102.0.1 dev r4-eth2 metric 100

10.0.0.0/24 via 10.103.0.1 dev r4-eth3 metric 200

10.1.0.0/24 via 10.103.0.1 dev r4-eth3

10.2.0.0/24 via 10.102.0.1 dev r4-eth2

10.3.0.0/24 dev r4-eth1 proto kernel scope link src 10.3.0.1

10.102.0.0/24 dev r4-eth2 proto kernel scope link src 10.102.0.2

10.103.0.0/24 dev r4-eth3 proto kernel scope link src 10.103.0.2

r4 has no route back to 10.101.0.0/24, nor does it have a default gateway. This causes two problems:

1) r4 will be unable to reply to those ICMP echo request packets. Attempts to contact any system on the 10.101.0.0/24 network will fail:

mininet> r4 ping 10.101.0.1

ping: connect: Network is unreachable

Because of the rp\_filter settings on r4...

mininet> r4 sysctl -a -r rp\_filter

net.ipv4.conf.all.arp\_filter = 0

net.ipv4.conf.all.rp\_filter = 2

net.ipv4.conf.default.arp\_filter = 0

net.ipv4.conf.default.rp\_filter = 2

net.ipv4.conf.lo.arp\_filter = 0

net.ipv4.conf.lo.rp\_filter = 2

net.ipv4.conf.r4-eth1.arp\_filter = 0

net.ipv4.conf.r4-eth1.rp\_filter = 2

net.ipv4.conf.r4-eth2.arp\_filter = 0

net.ipv4.conf.r4-eth2.rp\_filter = 2

net.ipv4.conf.r4-eth3.arp\_filter = 0

net.ipv4.conf.r4-eth3.rp\_filter = 2

2) incoming packets from a network that is not reachable will simply be dropped, so requests stop at r4 and never get passed on to h4.

The solution here is to give r4 a route back to 10.101.0.0/24 hosts. The obvious choices seems to be via r3.

Return to file \*.py. Add the following command to program:

info(net['r4'].cmd("ip route add 10.101.0.0/24 via 10.102.0.1 dev r4-eth2"))

* Rerun the program

mininet> r1 ping -c3 h4

PING 10.3.0.10 (10.3.0.10) 56(84) bytes of data.

64 bytes from 10.3.0.10: icmp\_seq=1 ttl=62 time=1.15 ms

64 bytes from 10.3.0.10: icmp\_seq=2 ttl=62 time=0.160 ms

64 bytes from 10.3.0.10: icmp\_seq=3 ttl=62 time=0.048 ms

--- 10.3.0.10 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2030ms

rtt min/avg/max/mdev = 0.048/0.451/1.147/0.493 ms

# Bài tập

continue to fix the following cases

r2 -> h1 h2 X h4 r1 X r4

r3 -> h1 X h3 h4 r1 X r4

r4 -> X h2 h3 h4 X r2 r3

# Tham khảo

1. https://stackoverflow.com/questions/72605106/simulate-a-network-with-routers-using-mininet-pox

(Tài liệu lưu hành nội bộ)

-----------------------------------------------

DANH MỤC HÌNH

**No table of figures entries found.**