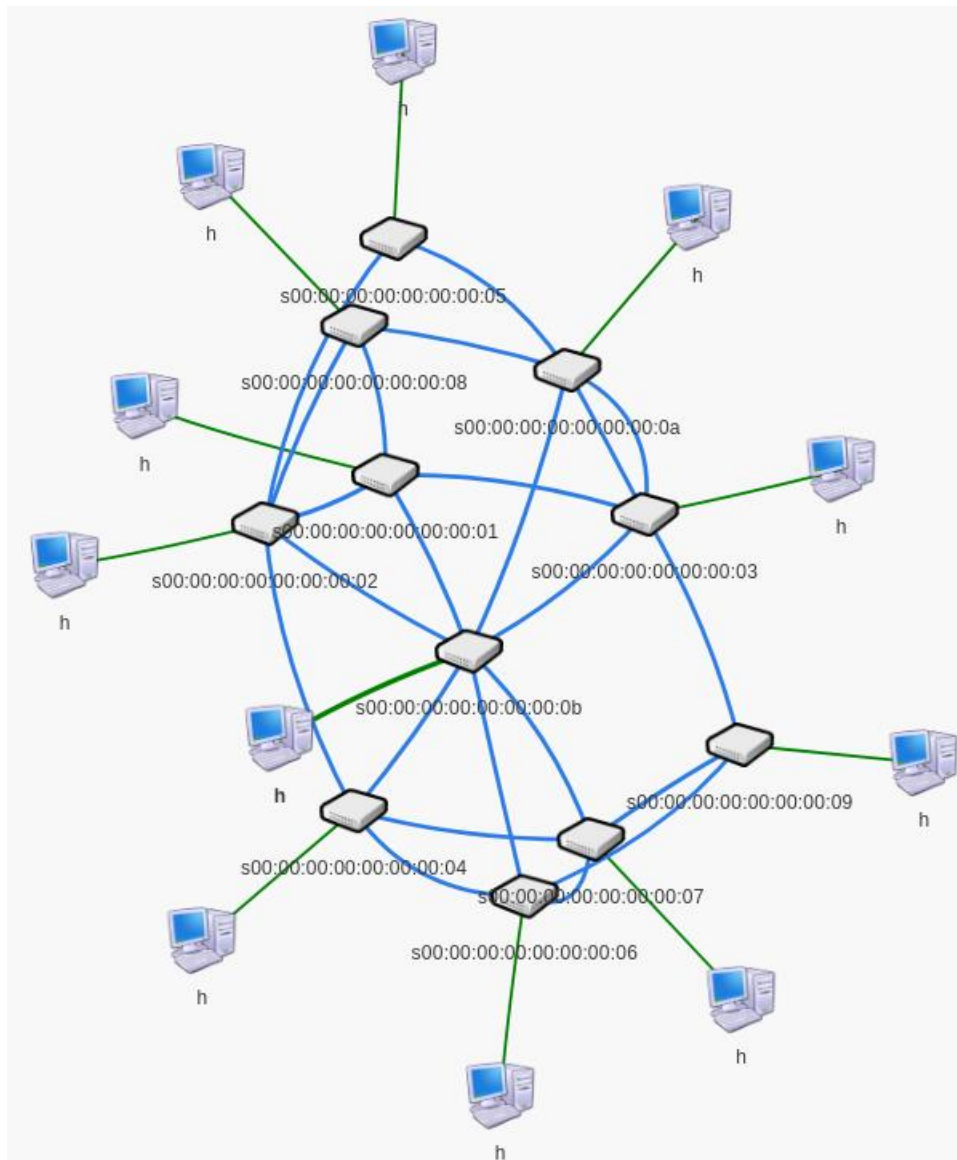


به نام خدا
امیررضا حسینی - ۹۸۲۰۳۶۳
شبکه ۲ - پروژه سوم

سوال یک) سناریویی که با ۱۱ سوئیچ و ۱۱ هاست ایجاد شده در تصویر زیر قابل مشاهده می باشد:



اجرای کنترلر floodlight با استفاده از دستور زیر:

```
java -jar target/floodlight.jar
```

MiniNet

```
anir@anlr-VirtualBox:~$ cd Desktop/
anir@anlr-VirtualBox:~/Desktop$ sudo mn --custom Topology.py --topo mytopo --controller=remote,ip=127.0.0.1,port=6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11
*** Adding links:
(h1, s2) (h2, s4) (h3, s1) (h4, s5) (h5, s10) (h6, s7) (h7, s11) (h8, s9) (h9, s8) (h10, s6) (h11, s3) (s1, s2) (s1, s11) (s2, s4) (s2, s5) (s3, s1) (s3, s9) (s4, s7) (s6, s4) (s6, s7) (s6, s9) (s7, s9) (s8, s1) (s8, s2) (s10, s3) (s10, s3) (s10, s5) (s10, s8) (s11, s2) (s11, s3) (s11, s4) (s11, s6) (s11, s7) (s11, s10)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Starting controller
c0
*** Starting 11 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 ...
*** Starting CLI:
mininet>
```

```
enabled ports
2022-12-31 22:26:40.998 INFO [n.f.t.TopologyManager] Recomputing topology due to: link-discovery
-updates
2022-12-31 22:26:56.123 INFO [n.f.l.l.LinkDiscoveryManager] Sending LLDP packets out of all the
enabled ports
2022-12-31 22:26:56.625 INFO [n.f.t.TopologyManager] Recomputing topology due to: link-discovery
-updates
2022-12-31 22:27:11.197 INFO [n.f.l.l.LinkDiscoveryManager] Sending LLDP packets out of all the
enabled ports
2022-12-31 22:27:11.266 INFO [n.f.t.TopologyManager] Recomputing topology due to: link-discovery
-updates
2022-12-31 22:27:26.291 INFO [n.f.l.l.LinkDiscoveryManager] Sending LLDP packets out of all the
enabled ports
2022-12-31 22:27:41.317 INFO [n.f.l.l.LinkDiscoveryManager] Sending LLDP packets out of all the
enabled ports
2022-12-31 22:27:41.353 INFO [n.f.t.TopologyManager] Recomputing topology due to: link-discovery
-updates
2022-12-31 22:27:56.469 INFO [n.f.l.l.LinkDiscoveryManager] Sending LLDP packets out of all the
enabled ports
2022-12-31 22:27:56.980 INFO [n.f.t.TopologyManager] Recomputing topology due to: link-discovery
-updates
```

همچنین کد مربوط به ساخت این سناریو به این صورت پیاده شده است (با نام `Topology.py` در پوشه پاسخنامه ضمیمه شده است):

```
#Amirreza Hosseini 9820363
#CN2 Project
#Floodlight Controller

from mininet.topo import Topo
class MyTopo( Topo ):
    def __init__( self ):
        Topo.__init__(self)
        #create 11 hosts
        Host1 = self.addHost('h1')
        Host2 = self.addHost('h2')
        Host3 = self.addHost('h3')
        Host4 = self.addHost('h4')
        Host5 = self.addHost('h5')
        Host6 = self.addHost('h6')
        Host7 = self.addHost('h7')
        Host8 = self.addHost('h8')
        Host9 = self.addHost('h9')
        Host10 = self.addHost('h10')
        Host11 = self.addHost('h11')

        #create 11 switches
        Switch1 = self.addSwitch('s1')
        Switch2 = self.addSwitch('s2')
        Switch3 = self.addSwitch('s3')
        Switch4 = self.addSwitch('s4')
        Switch5 = self.addSwitch('s5')
        Switch6 = self.addSwitch('s6')
        Switch7 = self.addSwitch('s7')
        Switch8 = self.addSwitch('s8')
        Switch9 = self.addSwitch('s9')
        Switch10 = self.addSwitch('s10')
        Switch11 = self.addSwitch('s11')
```

```
#create links between switches
self.addLink(Switch1, Switch2)
self.addLink(Switch2, Switch4)
self.addLink(Switch3, Switch1)
self.addLink(Switch2, Switch5)
self.addLink(Switch6, Switch4)
self.addLink(Switch6, Switch7)
self.addLink(Switch8, Switch1)
self.addLink(Switch6, Switch9)
self.addLink(Switch10, Switch5)
self.addLink(Switch11, Switch6)
self.addLink(Switch10, Switch3)
self.addLink(Switch4, Switch7)
self.addLink(Switch1, Switch11)
self.addLink(Switch7, Switch9)
self.addLink(Switch11, Switch2)
self.addLink(Switch10, Switch8)
self.addLink(Switch11, Switch3)
self.addLink(Switch11, Switch4)
self.addLink(Switch11, Switch7)
self.addLink(Switch11, Switch10)
self.addLink(Switch3, Switch9)
self.addLink(Switch8, Switch2)
self.addLink(Switch10, Switch3)

#create links between hosts and switches
self.addLink(Host1, Switch2)
self.addLink(Host2, Switch4)
self.addLink(Host3, Switch1)
self.addLink(Host4, Switch5)
self.addLink(Host5, Switch10)
self.addLink(Host6, Switch7)
self.addLink(Host7, Switch11)
self.addLink(Host8, Switch9)
self.addLink(Host9, Switch8)
self.addLink(Host10, Switch6)
self.addLink(Host11, Switch3)
```

```
topos = {'mytopo': (lambda: MyTopo())}
```

توضیحات کد: با استفاده از تابع `addHost()` که تابعی برگرفته از کتابخانه `mininet.topo` است که هاست اضافه میکند. تابع `addSwitch()` هم سوئیچ اضافه میکند و در نهایت با استفاده از تابع `addLink()` لینک‌های ارتباطی بین هاست‌ها و سوئیچ‌ها را برقرار میکنیم. در آخر با استفاده از قطعه کد خط آخر شبکه طراحی شده را با نامی مثلا در اینجا `MyTopo` تثبیت میکنیم تا بتوانیم آن را به عنوان یک `Custom` توپولوژی به مینیت معرفی کرد. سپس با دستور زیر، آن توپولوژی را همراه با کنترلر فلودلایت در مینیت ایجاد کردم:

```
sudo mn --custom Topology.py --topo mytopo --controller=remote,ip=127.0.0.1,port=6653
```

سپس در مینیت قبل از اضافه کردن هر `entity` در `flowtable` سوئیچ‌ها یک پینگ از هاست ۹ به هاست ۸ می‌گیریم. در این زمان که هیچ `entity` خودمان اضافه نکرده‌ایم، خود فلودلایت، کوتاه‌ترین مسیر را انتخاب می‌کند.

```
mininet> h9 ping h8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=122 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=0.208 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=0.102 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=0.038 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=0.038 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=0.048 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=0.066 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=0.080 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=0.040 ms
^C
--- 10.0.0.8 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8056ms
rtt min/avg/max/mdev = 0.038/13.651/122.246/38.394 ms
mininet>
```

سوال سوم) با استفاده از دستور `pingall` ارتباط همه میزبان‌ها با یکدیگر را بررسی کردم، در هر بار ممکن است بعضی از بسته‌ها دراپ شوند اما این امر ثابت نیست و در دفعات مختلف اجرای این دستور ممکن است بسته‌های هاست‌های متفاوتی دراپ شوند اما مجموعاً ما به همه بسته‌ها پینگ داریم. پس در نهایت نتیجه می‌گیریم لینک به تمام هاست‌ها از طریق سوئیچ‌های مختلف برقرار است.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Results: 0% dropped (110/110 received)
mininet>
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 X
h7 -> h1 h2 h3 X h5 h6 h8 h9 h10 h11
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11
h10 -> h1 h2 h3 X h5 h6 h7 h8 h9 h11
h11 -> h1 h2 h3 h4 h5 X h7 h8 h9 h10
*** Results: 3% dropped (106/110 received)
mininet>
```

سوال چهارم) دستور **nodes**: در خروجی این دستور نودهای موجود در شبکه مانند سوئیچها و هاستها نمایش داده می شود. (طبق توضیحات بیشتر در پروژه قبلی)

```
mininet> nodes
available nodes are:
c0 h1 h10 h11 h2 h3 h4 h5 h6 h7 h8 h9 s1 s10 s11 s2 s3 s4 s5 s6 s7 s8 s9
mininet>
```

دستور **net**: دستور **net** برای نشان دادن روابط میان نودهاست. (هر نود به چه نودی و از طریق چه چیزی متصل است).

```
mininet> net
h1 h1-eth0:s2-eth6
h2 h2-eth0:s4-eth5
h3 h3-eth0:s1-eth5
h4 h4-eth0:s5-eth3
h5 h5-eth0:s10-eth6
h6 h6-eth0:s7-eth5
h7 h7-eth0:s11-eth8
h8 h8-eth0:s9-eth4
h9 h9-eth0:s8-eth4
h10 h10-eth0:s6-eth5
h11 h11-eth0:s3-eth6
s1 lo: s1-eth1:s2-eth1 s1-eth2:s3-eth1 s1-eth3:s8-eth1 s1-eth4:s11-eth2 s1-eth5:h3-eth0
s2 lo: s2-eth1:s1-eth1 s2-eth2:s4-eth1 s2-eth3:s5-eth1 s2-eth4:s11-eth3 s2-eth5:s8-eth3 s2-eth6:h1-eth0
s3 lo: s3-eth1:s1-eth2 s3-eth2:s10-eth2 s3-eth3:s11-eth4 s3-eth4:s9-eth3 s3-eth5:s10-eth5 s3-eth6:h11-eth0
s4 lo: s4-eth1:s2-eth2 s4-eth2:s6-eth1 s4-eth3:s7-eth2 s4-eth4:s11-eth5 s4-eth5:h2-eth0
s5 lo: s5-eth1:s2-eth3 s5-eth2:s10-eth1 s5-eth3:h4-eth0
s6 lo: s6-eth1:s4-eth2 s6-eth2:s7-eth1 s6-eth3:s9-eth1 s6-eth4:s11-eth1 s6-eth5:h10-eth0
s7 lo: s7-eth1:s6-eth2 s7-eth2:s4-eth3 s7-eth3:s9-eth2 s7-eth4:s11-eth6 s7-eth5:h6-eth0
s8 lo: s8-eth1:s1-eth3 s8-eth2:s10-eth3 s8-eth3:s2-eth5 s8-eth4:h9-eth0
s9 lo: s9-eth1:s6-eth3 s9-eth2:s7-eth3 s9-eth3:s3-eth4 s9-eth4:h8-eth0
s10 lo: s10-eth1:s5-eth2 s10-eth2:s3-eth2 s10-eth3:s8-eth2 s10-eth4:s11-eth7 s10-eth5:s3-eth5 s10-eth6:h5-eth0
s11 lo: s11-eth1:s6-eth4 s11-eth2:s1-eth4 s11-eth3:s2-eth4 s11-eth4:s3-eth3 s11-eth5:s4-eth4 s11-eth6:s7-eth4 s11-eth7:s10-eth4 s11-eth8:h7-eth0
c0
```

دستور **dump**: دستور **dump** نیز برای نمایش نام نود، **network interface** به همراه آدرس **ip** آن به علاوه پروسس آیدی آن در سیستم را نشان می دهد.

```

mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=2835>
<Host h2: h2-eth0:10.0.0.2 pid=2838>
<Host h3: h3-eth0:10.0.0.3 pid=2841>
<Host h4: h4-eth0:10.0.0.4 pid=2844>
<Host h5: h5-eth0:10.0.0.5 pid=2847>
<Host h6: h6-eth0:10.0.0.6 pid=2850>
<Host h7: h7-eth0:10.0.0.7 pid=2853>
<Host h8: h8-eth0:10.0.0.8 pid=2856>
<Host h9: h9-eth0:10.0.0.9 pid=2859>
<Host h10: h10-eth0:10.0.0.10 pid=2862>
<Host h11: h11-eth0:10.0.0.11 pid=2865>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None pid=2871>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,s2-eth5:None,s2-eth6:None pid=2874>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None,s3-eth4:None,s3-eth5:None,s3-eth6:None pid=2877>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-eth4:None,s4-eth5:None pid=2880>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None,s5-eth3:None,s5-eth4:None,s5-eth5:None pid=2883>
<OVSSwitch s6: lo:127.0.0.1,s6-eth1:None,s6-eth2:None,s6-eth3:None,s6-eth4:None,s6-eth5:None pid=2886>
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None,s7-eth4:None,s7-eth5:None pid=2889>
<OVSSwitch s8: lo:127.0.0.1,s8-eth1:None,s8-eth2:None,s8-eth3:None,s8-eth4:None pid=2892>
<OVSSwitch s9: lo:127.0.0.1,s9-eth1:None,s9-eth2:None,s9-eth3:None,s9-eth4:None pid=2895>
<OVSSwitch s10: lo:127.0.0.1,s10-eth1:None,s10-eth2:None,s10-eth3:None,s10-eth4:None,s10-eth5:None,s10-eth6:None pid=2898>
<OVSSwitch s11: lo:127.0.0.1,s11-eth1:None,s11-eth2:None,s11-eth3:None,s11-eth4:None,s11-eth5:None,s11-eth6:None,s11-eth7:None,s11-eth8:None pid=2901>
<RemoteController{'ip': '127.0.0.1', 'port': 6653} c0: 127.0.0.1:6653 pid=2828>

```

سوال پنجم)

همه لینک‌ها برقرار هستند همان‌طور که از طریق دستور **pingall** نیز بررسی شد، اما برای بررسی سرعت و وضعیت لینک‌ها به مثال زیر توجه کنید: به عنوان مثال وضعیت لینک‌های هاست یک با همه ۱۰ هاست دیگر در تصویر زیر بررسی شده است: سرعت لینک‌ها بین هاست‌های یک با ده هاست دیگر چیزی بین ۱۲.۴ گیگ بر ثانیه و ۱۷.۳ گیگ بر ثانیه است.

```

mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['16.8 Gbits/sec', '16.7 Gbits/sec']
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['16.4 Gbits/sec', '16.4 Gbits/sec']
mininet> iperf h1 h4
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['17.3 Gbits/sec', '17.3 Gbits/sec']
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['12.4 Gbits/sec', '12.4 Gbits/sec']
mininet> iperf h1 h6
*** Iperf: testing TCP bandwidth between h1 and h6
*** Results: ['16.2 Gbits/sec', '16.3 Gbits/sec']
mininet> iperf h1 h7
*** Iperf: testing TCP bandwidth between h1 and h7
*** Results: ['13.0 Gbits/sec', '13.0 Gbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['12.9 Gbits/sec', '13.7 Gbits/sec']
mininet> iperf h1 h9
*** Iperf: testing TCP bandwidth between h1 and h9
*** Results: ['15.1 Gbits/sec', '15.1 Gbits/sec']
mininet> iperf h1 h10
*** Iperf: testing TCP bandwidth between h1 and h10
*** Results: ['16.1 Gbits/sec', '16.1 Gbits/sec']
mininet> iperf h1 h11
*** Iperf: testing TCP bandwidth between h1 and h11
*** Results: ['13.2 Gbits/sec', '13.1 Gbits/sec']
mininet> █

```

روش دیگر برای انجام تست پهنای باند:

```
"Node: h1"
root@amir-VirtualBox:~/Desktop# iperf -c 10.0.0.8 -p 5566 -t 5
Client connecting to 10.0.0.8, TCP port 5566
TCP window size: 85.3 KByte (default)
-----
[ 51] local 10.0.0.1 port 42320 connected with 10.0.0.8 port 5566
[ ID] Interval      Transfer    Bandwidth
[ 51] 0.0- 5.0 sec  23.2 GBytes 39.8 Gbits/sec
root@amir-VirtualBox:~/Desktop#

"Node: h8"
root@amir-VirtualBox:~/Desktop# iperf -s -p 5566 -i 1
Server listening on TCP port 5566
TCP window size: 85.3 KByte (default)
-----
[ 52] local 10.0.0.8 port 5566 connected with 10.0.0.1 port 42320
[ ID] Interval      Transfer    Bandwidth
[ 52] 0.0- 1.0 sec  4.39 GBytes 37.7 Gbits/sec
[ 52] 1.0- 2.0 sec  4.76 GBytes 40.9 Gbits/sec
[ 52] 2.0- 3.0 sec  4.77 GBytes 41.0 Gbits/sec
[ 52] 3.0- 4.0 sec  4.54 GBytes 39.0 Gbits/sec
[ 52] 4.0- 5.0 sec  4.70 GBytes 40.4 Gbits/sec
[ 52] 0.0- 5.0 sec  23.2 GBytes 39.8 Gbits/sec
root@amir-VirtualBox:~/Desktop#

mininet> xterm h1
mininet> xterm h8
mininet>
```

اطلاعات مشاهده شده در پنجره‌های بالا بیانگر دیتای انتقال یافته از h1 به h8 طی مدت ۵ ثانیه است که در این روش یکی از آنها سرور و یکی دیگر کلاینت شده و با میانگینی تعداد پکت‌ها، پهنای باند در کلاینت محاسبه شده است.

سوال ششم)

قصد داریم، entityهایی را به نحوی به flowtableها اضافه کنیم که به صورت یک مسیر دلخواه پیکربندی شوند. مسیرهای انتخابی من به صورت زیر می‌باشند:

مسیر شماره (۱)

H4 → S5 → S10 → S3 → S9 → S6 → S7 → H6

مسیر شماره (۲)

H8 → S9 → S6 → S11 → S10 → S5 → S2 → S8 → H9

مسیر شماره (۳)

H10 → S6 → S7 → S4 → S1 → S8 → S2 → S5 → S10 → H5


```

# Amirreza Hosseini 9820363
# CN2 Project
# Floodlight Controller - flows

import httplib
import json

class StaticEntryPusher(object):
    def __init__(self, server):
        self.server = server
    def get(self, data):
        ret = self.rest_call({}, 'GET')
        return json.loads(ret[2])
    def Set(self, data):
        ret = self.rest_call(data, 'POST')
        return ret[0] == 200
    def remove(self, objtype, data):
        ret = self.rest_call(data, 'DELETE')
        return ret[0] == 200
    def rest_call(self, data, action):
        path = '/wm/staticentrypusher/json'
        header = {
            'Content-type': 'application/json',
            'Accept': 'application/json'
        }
        body = json.dumps(data)
        Conn = httplib.HTTPConnection(self.server, 8080)
        Conn.request(action, path, body, header)
        response = Conn.getresponse()
        ret = (response.status, response.reason, response.read())
        print(ret)
        Conn.close()
        return ret

pusher = StaticEntryPusher('127.0.0.1')

# PATH1 = H4 > S5 > S10 > S3 > S9 > S6 > S7 > H6
# Route from H4 to H6
flow1 = {
    "switch": "00:00:00:00:00:00:00:05",
    "name": "flow1",
    "eth_type": "0x0800",

```



```
    "ipv4_src": "10.0.0.04",
    "ipv4_dst": "10.0.0.06",
    "priority": "32768",
    "in_port": "3",
    "active": "true",
    "actions": "output=2",
}
flow2 = {
    "switch": "00:00:00:00:00:00:00:0a",
    "name": "flow2",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.04",
    "ipv4_dst": "10.0.0.06",
    "priority": "32768",
    "in_port": "1",
    "active": "true",
    "actions": "output=5",
}
flow3 = {
    "switch": "00:00:00:00:00:00:00:03",
    "name": "flow3",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.04",
    "ipv4_dst": "10.0.0.06",
    "priority": "32768",
    "in_port": "5",
    "active": "true",
    "actions": "output=4",
}
flow4 = {
    "switch": "00:00:00:00:00:00:00:09",
    "name": "flow4",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.04",
    "ipv4_dst": "10.0.0.06",
    "priority": "32768",
    "in_port": "3",
    "active": "true",
    "actions": "output=1",
}
flow5 = {
    "switch": "00:00:00:00:00:00:00:06",
    "name": "flow5",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.04",
```

```
    "ipv4_dst": "10.0.0.06",
    "priority": "32768",
    "in_port": "3",
    "active": "true",
    "actions": "output=2",
  }
}
flow6 = {
  "switch": "00:00:00:00:00:00:00:07",
  "name": "flow6",
  "eth_type": "0x0800",
  "ipv4_src": "10.0.0.04",
  "ipv4_dst": "10.0.0.06",
  "priority": "32768",
  "in_port": "1",
  "active": "true",
  "actions": "output=5",
}

# PATH2 = H8 > S9 > S6 > S11 > S10 > S5 > S2 > S8 > H9
#route from H8 to H9
flow7 = {
  "switch": "00:00:00:00:00:00:00:09",
  "name": "flow7",
  "eth_type": "0x0800",
  "ipv4_src": "10.0.0.08",
  "ipv4_dst": "10.0.0.09",
  "priority": "32768",
  "in_port": "4",
  "active": "true",
  "actions": "output=1",
}
}
flow8 = {
  "switch": "00:00:00:00:00:00:00:06",
  "name": "flow8",
  "eth_type": "0x0800",
  "ipv4_src": "10.0.0.08",
  "ipv4_dst": "10.0.0.09",
  "priority": "32768",
  "in_port": "3",
  "active": "true",
  "actions": "output=4",
}
}
flow9 = {
  "switch": "00:00:00:00:00:00:00:0b",
  "name": "flow9",
```

```
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.08",
    "ipv4_dst": "10.0.0.09",
    "priority": "32768",
    "in_port": "1",
    "active": "true",
    "actions": "output=4",
}
flow10 = {
    "switch": "00:00:00:00:00:00:00:0a",
    "name": "flow10",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.08",
    "ipv4_dst": "10.0.0.09",
    "priority": "32768",
    "in_port": "7",
    "active": "true",
    "actions": "output=1",
}
flow11 = {
    "switch": "00:00:00:00:00:00:00:05",
    "name": "flow11",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.08",
    "ipv4_dst": "10.0.0.09",
    "priority": "32768",
    "in_port": "2",
    "active": "true",
    "actions": "output=1",
}
flow12 = {
    "switch": "00:00:00:00:00:00:00:02",
    "name": "flow12",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.08",
    "ipv4_dst": "10.0.0.09",
    "priority": "32768",
    "in_port": "3",
    "active": "true",
    "actions": "output=5",
}
flow13 = {
    "switch": "00:00:00:00:00:00:00:08",
    "name": "flow13",
    "eth_type": "0x0800",
```

```
    "ipv4_src": "10.0.0.08",
    "ipv4_dst": "10.0.0.09",
    "priority": "32768",
    "in_port": "3",
    "active": "true",
    "actions": "output=4",
}

# PATH3 = H10 > S6 > S7 > S4 > S1 > S8 > S2 > S5 > S10 > H5
#route from H10 to H5
flow14 = {
    "switch": "00:00:00:00:00:00:00:06",
    "name": "flow14",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "5",
    "active": "true",
    "actions": "output=2",
}
flow15 = {
    "switch": "00:00:00:00:00:00:00:07",
    "name": "flow15",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "1",
    "active": "true",
    "actions": "output=2",
}
flow16 = {
    "switch": "00:00:00:00:00:00:00:04",
    "name": "flow16",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "3",
    "active": "true",
    "actions": "output=4",
}
flow17 = {
    "switch": "00:00:00:00:00:00:00:0b",
```

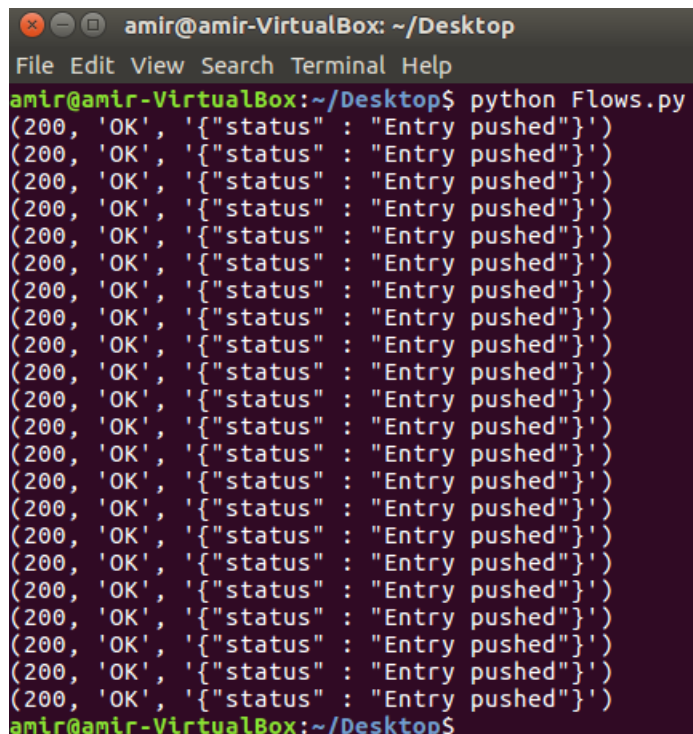
```
    "name": "flow17",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "5",
    "active": "true",
    "actions": "output=2",
}
flow18 = {
    "switch": "00:00:00:00:00:00:00:01",
    "name": "flow18",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "4",
    "active": "true",
    "actions": "output=3",
}
flow19 = {
    "switch": "00:00:00:00:00:00:00:08",
    "name": "flow19",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "1",
    "active": "true",
    "actions": "output=3",
}
flow20 = {
    "switch": "00:00:00:00:00:00:00:02",
    "name": "flow20",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "5",
    "active": "true",
    "actions": "output=3",
}
flow21 = {
    "switch": "00:00:00:00:00:00:00:05",
    "name": "flow21",
```

```
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "1",
    "active": "true",
    "actions": "output=2",
}
flow22 = {
    "switch": "00:00:00:00:00:00:00:0a",
    "name": "flow22",
    "eth_type": "0x0800",
    "ipv4_src": "10.0.0.10",
    "ipv4_dst": "10.0.0.05",
    "priority": "32768",
    "in_port": "1",
    "active": "true",
    "actions": "output=6",
}
```

```
pusher.Set(flow1)
pusher.Set(flow2)
pusher.Set(flow3)
pusher.Set(flow4)
pusher.Set(flow5)
pusher.Set(flow6)
pusher.Set(flow7)
pusher.Set(flow8)
pusher.Set(flow9)
pusher.Set(flow10)
pusher.Set(flow11)
pusher.Set(flow12)
pusher.Set(flow13)
pusher.Set(flow14)
pusher.Set(flow15)
pusher.Set(flow16)
pusher.Set(flow17)
pusher.Set(flow18)
pusher.Set(flow19)
pusher.Set(flow20)
pusher.Set(flow21)
pusher.Set(flow22)
```

کد فوق در فایل `flows.py` در پوشه پاسخنامه ضمیمه شده است. سپس با دستور زیر `entity` ها را به `flowtable` سوئیچ ها اضافه کردم.

```
~$ python flows.py
```



```
amir@amir-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
amir@amir-VirtualBox:~/Desktop$ python Flows.py
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
(200, 'OK', '{"status": "Entry pushed"}')
amir@amir-VirtualBox:~/Desktop$
```

بعد از اضافه کردن `entity` ها به دلیل اینکه مسیر دلخواه ما از `Switch1` نیز عبور می کند پس `flowtable` این سوئیچ به روزرسانی شده و حاوی مقدار است.

برای مسیرهای طراحی شده من نیاز به ۲۲ سطر در `Flow Table` هاست :

به عنوان مثال فرض کنید بسته ای هاست شماره ۴ را به مقصد هاست شماره ۶ ترک کند، برای رسیدن به اولین سوئیچ که سوئیچ شماره ۵ است باید از پورت شماره ۳ آن وارد شود. طبق سطری که ما به جدول اضافه کرده ایم هنگامی که پکتی با مبدأ هاست ۴ و مقصد هاست ۶ از طریق پورت ۳ به سوئیچ ۴ وارد شود باید به عنوان عکس العمل از پورت ۲ سوئیچ خارج شود. در اثر این عمل از طریق پورت ۱ وارد سوئیچ شماره ۱۰ میشود. طبق سطری که ما به جدول اضافه کرده ایم هنگامی که پکتی با مبدأ هاست ۴ و مقصد هاست ۶ از طریق پورت ۱ به سوئیچ ۱۰ وارد شود باید به عنوان عکس العمل از پورت ۵ سوئیچ خارج شود. در اثر این عمل از طریق پورت ۵ وارد سوئیچ شماره ۳ میشود. طبق سطری که ما به جدول اضافه کرده ایم هنگامی که پکتی با مبدأ هاست ۴ و مقصد هاست ۶ از طریق پورت ۵ به سوئیچ ۳ وارد شود باید به عنوان عکس العمل از پورت ۴ سوئیچ خارج شود. در اثر این عمل از طریق پورت ۳ وارد سوئیچ شماره ۹ میشود. طبق سطری که ما به جدول اضافه کرده ایم هنگامی که پکتی با مبدأ هاست ۴ و مقصد هاست ۶ از طریق پورت ۳ به سوئیچ ۹ وارد شود باید به عنوان عکس العمل از پورت ۱ سوئیچ خارج شود. در اثر این عمل از طریق پورت ۳ وارد سوئیچ شماره ۶ میشود. طبق سطری که ما به جدول اضافه کرده ایم هنگامی که پکتی با مبدأ هاست ۴ و مقصد هاست ۶ از

Flow Table

Show 10 entries

Search:

Table No	Pkt.Count	Byte	Duration(s)	Priority	IdleTimeoutSec	HardTimeoutSec	Flags	Instructions
0x0	0	0	288	32768	0	0	SEND_FLOW_REM	output=2
0x0	0	0	288	32768	0	0	SEND_FLOW_REM	output=4
0x0	0	0	288	32768	0	0	SEND_FLOW_REM	output=2
0x0	53864	11014013	3666	0	0	0		output=controller

Showing 1 to 4 of 4 entries

Previous
1
Next