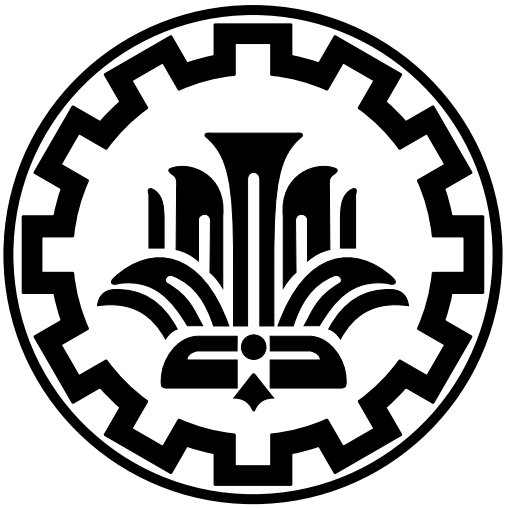


Isfahan University of Technology

CE Department

Network 2: Project 2

MohammadMahdi Ahmadi

9816743

Fall 2023

Dr. Hashemi

Contents

1

ابزار Wireshark برای شناسایی مشکلات شبکه، بسته‌های شبکه را تجزیه و تحلیل می‌کند. الگوهای ترافیکی، پروتکل ها و مشکلات مربوط به performance را به دقت مورد بررسی قرار میدهد. با قابلیت‌های ،filtering نمودارها و آمارها، به تعیین دقیق مشکلات کمک میکند. همچنین میتواند روندهای شبکه را visualize کند.

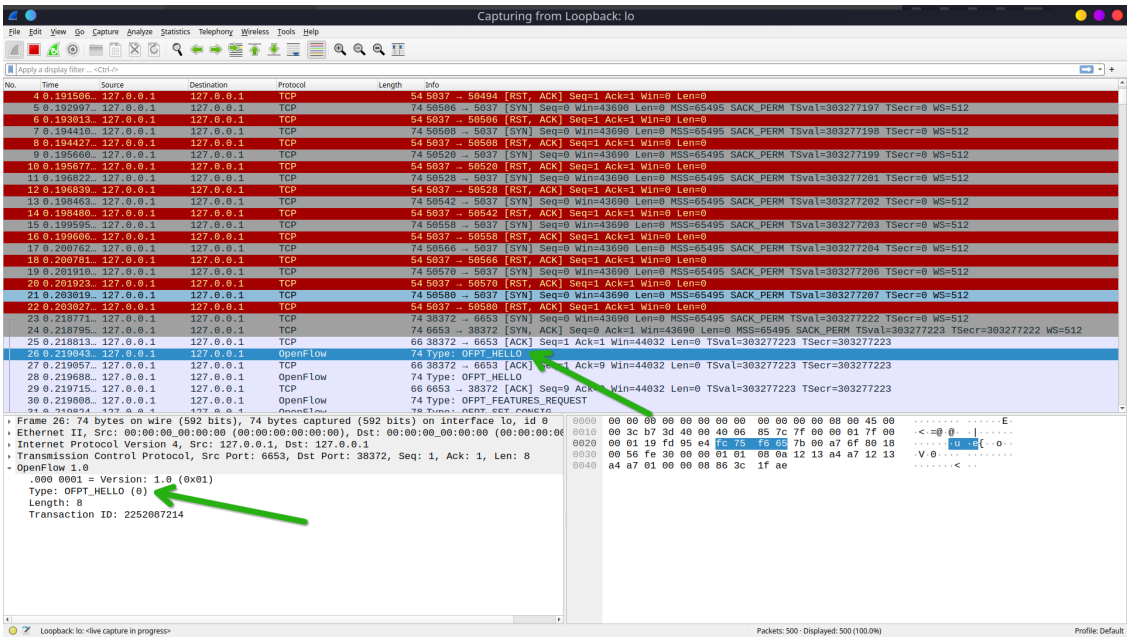
در زمینه شناسایی تهدیدهای امنیتی، Wireshark به عنوان یک ابزار قدرتمند برای شناسایی anomalies ها، تلاش‌های دسترسی غیرمجاز و فعالیت‌های malicious عمل می‌کند. این ابزار قادر به تجزیه و تحلیل محتوای هاpacket است، که می‌تواند تهدیدهای امنیتی مثل ،exploit malware و حملات denial-of-service را آشکار کند. هرچند Wireshark قادر به رمزگشایی ترافیک رمزنگاری شده بدون کلیدهای مناسب نیست، اما هنوز می‌تواند هدرهای غیررمزنگاری شده را تجزیه و تحلیل کند. علاوه بر این، این ابزار در مانیتورینگ ارتباطات دستگاه‌های اینترنت اشیاء (IoT) نیز مفید است و به شناسایی مشکلات امنیتی مرتبط با این دستگاه‌ها کمک می‌کند.

2

2.1

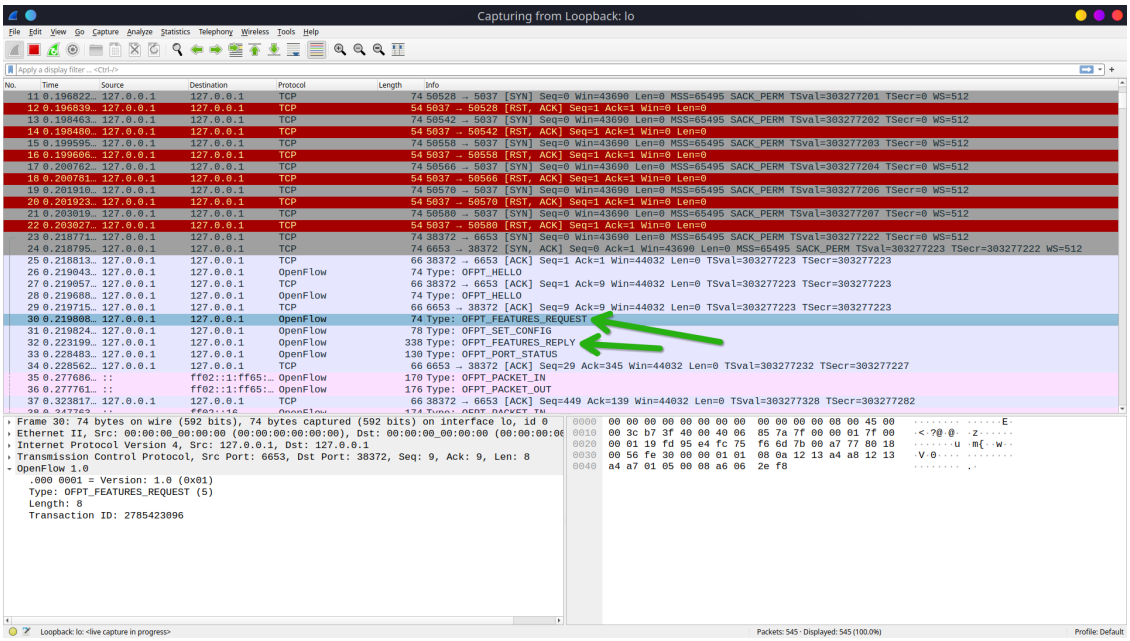
از پروتکل TCP و یا OpenFlow استفاده میکنند

2.2

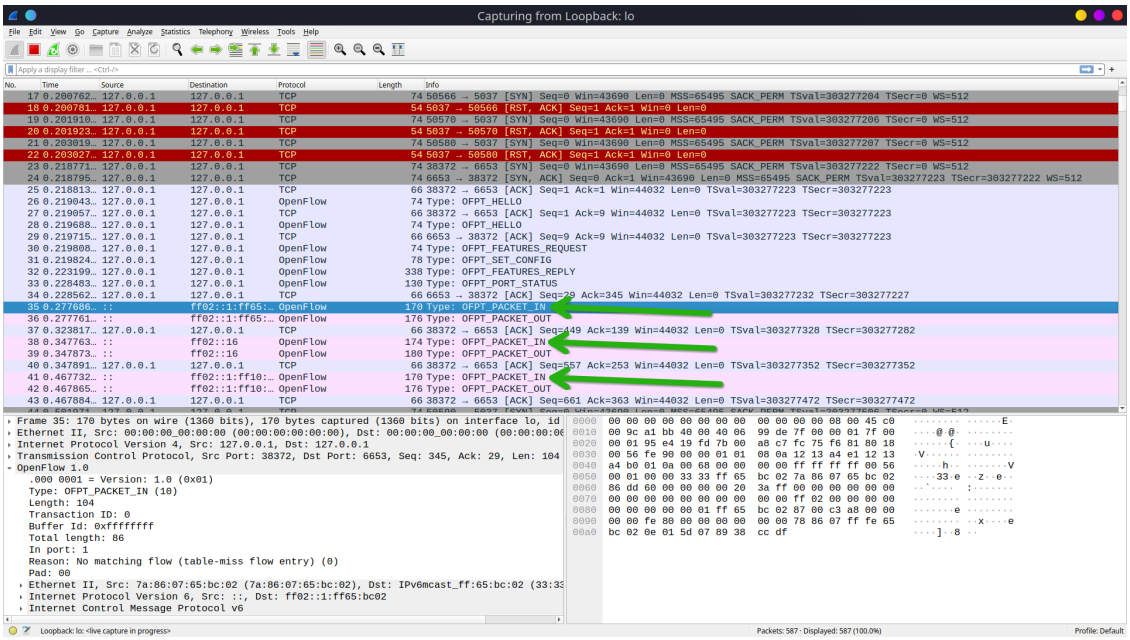


2.3

وقتی یک سوییچ به شبکه SDN اضافه میشود، یک پیغام Feature_Request از کنترلر دریافت میکند که به معنای این است که کنترل میخواهد اطلاعات آن سوییچ را دریافت کند. در پاسخ سوییچ اطلاعات خود را در قالب پیام Feature_Reply به کنترلر ارسال میکند. این اطلاعات شامل ورژن OpenFlow سوییچ، ماکزیمم اندازه ورودی Flow ها، action های پشتیبانی شده و ... است.



2.4



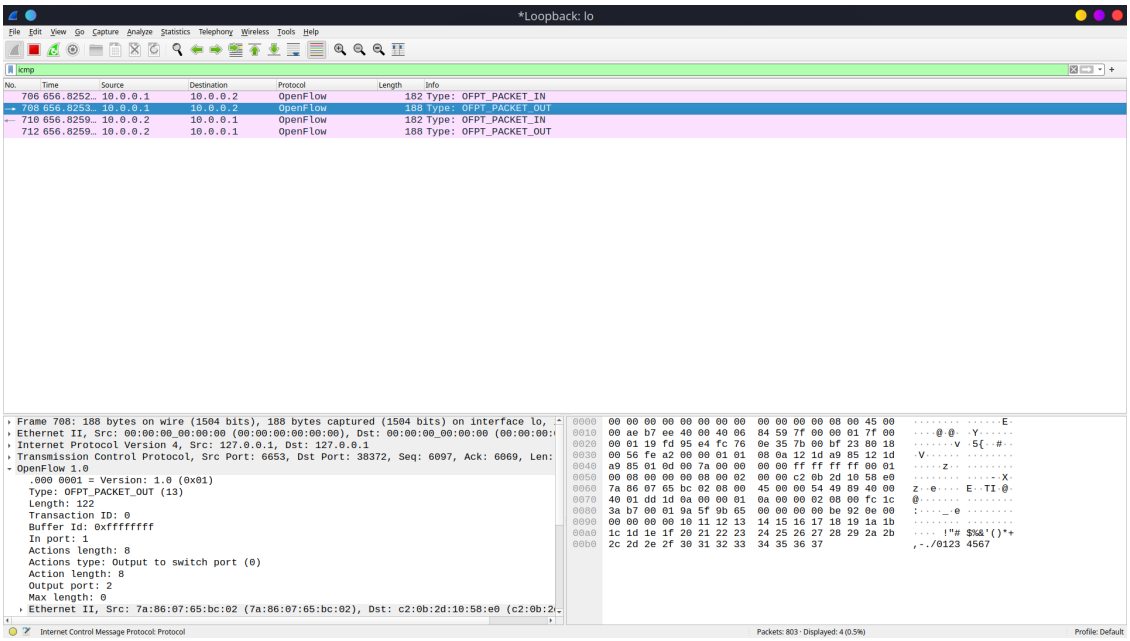
2.5

دلیل اول زمانی است که سوییچ نمیداند باید با بسته ورودی چه کند، پس بسته را برای کنترلر میفرستد تا کنترلر تصمیم بگیرد. در نهایت ممکن است کنترلر FlowTable ها را در سوییچ ها آپدیت کند. دلیل دوم زمانی است که کنترلر صراحتا گفته باشد که بسته های خاصی را برای من بفرست.

2.6

برای بسته ها پروتکل OpenFlow استفاده شده است.

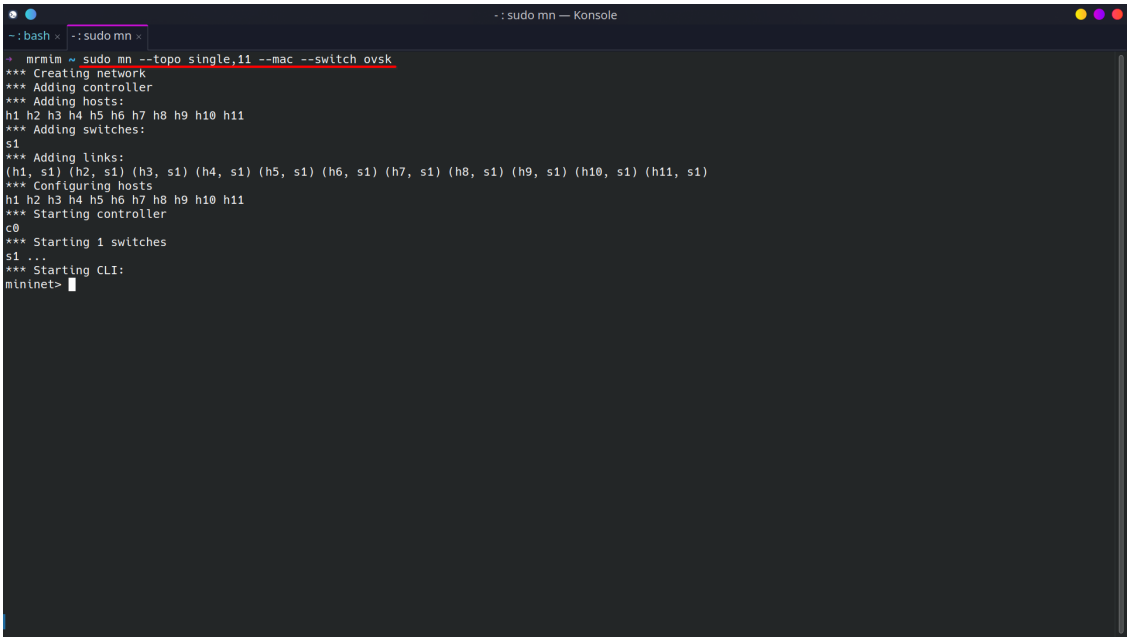
سوییچ نتوانسته است که مسیریابی را به درستی انجام بدهد و table-miss اتفاق افتاده است. پس بسته های icmp در بسته های OpenFlow گنجانده شده اند و به کنترلر فرستاده شدند تا کنترل مسیریابی را انجام بدهد.



3

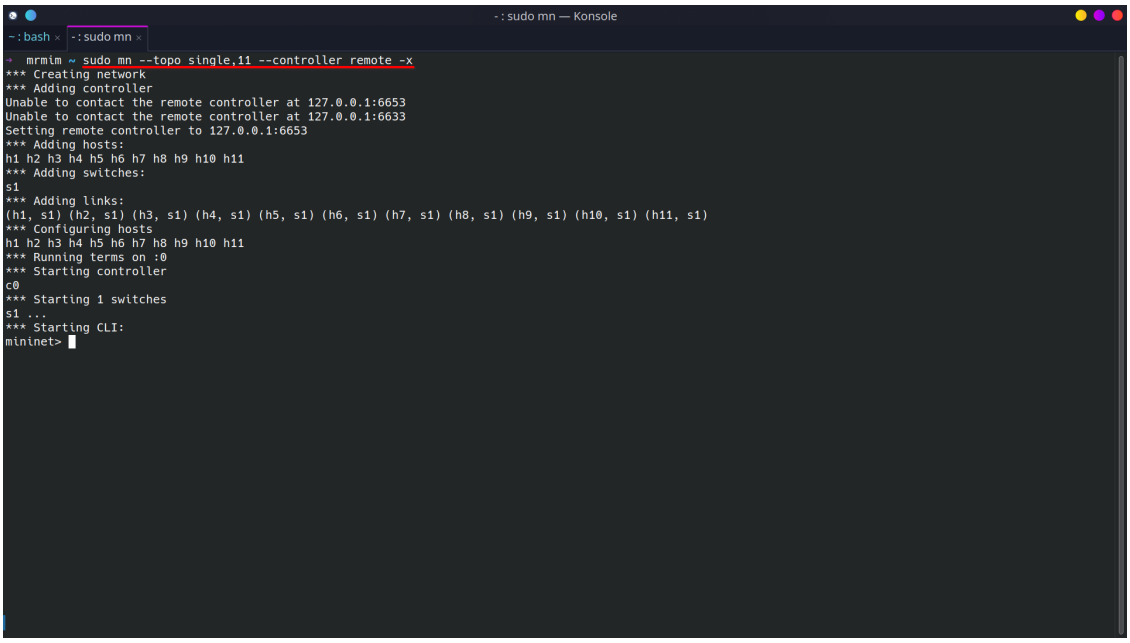
1. sudo mn –topo single,11 –mac –switch ovsk

    - `--topo single,11`: Creates a single-switch topology with 11 hosts.
    - `--mac`: Assigns MAC addresses to hosts.
    - `--switch ovsk`: Specifies that Open vSwitch should be used as the switch.



2. sudo mn –topo single,11 –controller remote -x

   - `--topo single,11`: Creates a single-switch topology with 11 hosts.
   - `--controller remote`: Configures a remote controller for SDN functionality.
   - `-x`: Enables Mininet's CLI (Command Line Interface) after launching.

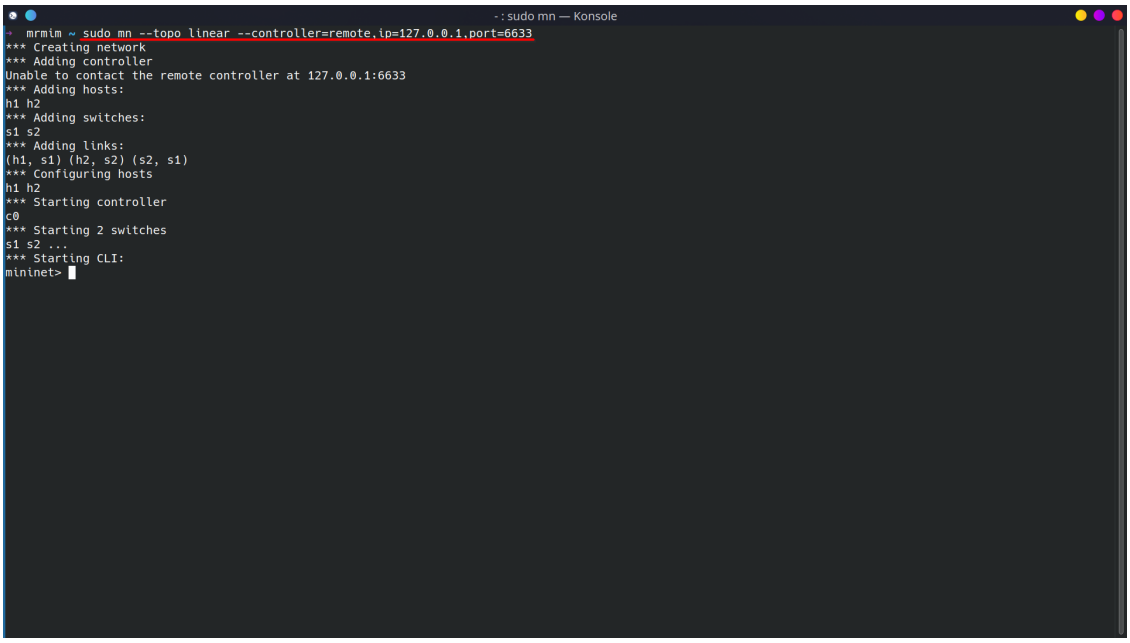


3. sudo mn –topo tree,11 –mac –arp

   - `--topo tree,11`: Creates a tree topology with 2^{11} hosts and $2^{11} - 1$ switches.
   - `--mac`: Assigns MAC addresses to hosts.
   - `--arp`: Enables ARP (Address Resolution Protocol) in the network.

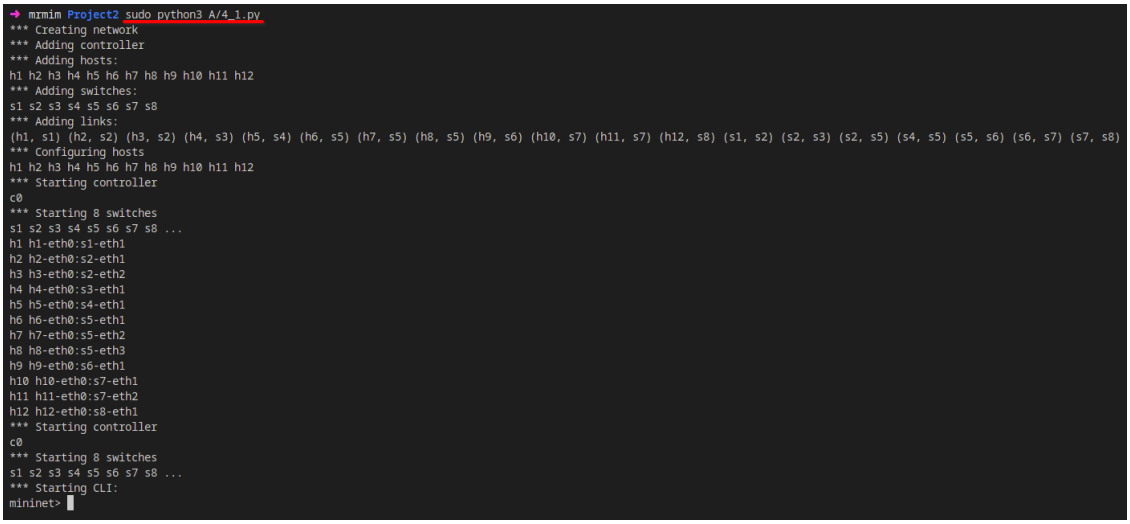
4. sudo mn –topo linear –controller=remote,ip=127.0.0.1,port=6633

    - `--topo linear`: Creates a linear topology (a straight line) with a default of 2 switches.
    - `--controller=remote,ip=127.0.0.1,port=6633`: Configures a remote controller with the specified IP address (127.0.0.1) and port (6633).



4

4.1




```python
from mininet.net import Mininet
from mininet.topo import Topo
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel


class MyTopology(Topo):
    def __init__(self):
        Topo.__init__(self)

        # Adding switches
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')
        s4 = self.addSwitch('s4')
        s5 = self.addSwitch('s5')
        s6 = self.addSwitch('s6')
        s7 = self.addSwitch('s7')
        s8 = self.addSwitch('s8')

        # Adding hosts
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
```

```
        h3 = self.addHost('h3')
        h4 = self.addHost('h4')
        h5 = self.addHost('h5')
        h6 = self.addHost('h6')
        h7 = self.addHost('h7')
        h8 = self.addHost('h8')
        h9 = self.addHost('h9')
        h10 = self.addHost('h10')
        h11 = self.addHost('h11')
        h12 = self.addHost('h12')

        # Connecting hosts to switches
        self.addLink(h1, s1)
        self.addLink(h2, s2)
        self.addLink(h3, s2)
        self.addLink(h4, s3)
        self.addLink(h5, s4)
        self.addLink(h6, s5)
        self.addLink(h7, s5)
        self.addLink(h8, s5)
        self.addLink(h9, s6)
        self.addLink(h10, s7)
        self.addLink(h11, s7)
        self.addLink(h12, s8)

        # Connecting switches
        self.addLink(s1, s2)
        self.addLink(s2, s3)
        self.addLink(s2, s5)
        self.addLink(s4, s5)
        self.addLink(s5, s6)
        self.addLink(s6, s7)
        self.addLink(s7, s8)


def runExperiment():
    topo = MyTopology()
    net = Mininet(topo)
    net.start()
```

```
63
64      dumpNodeConnections(net.hosts)
65      net.interact()
66
67      net.stop()
68
69  if __name__ == '__main__':
70      # Tell mininet to print useful information
71      setLogLevel('info')
72      runExperiment()
```

4.2

```
h2 h2-eth0:s2-eth1
h3 h3-eth0:s2-eth2
h4 h4-eth0:s3-eth1
h5 h5-eth0:s4-eth1
h6 h6-eth0:s5-eth1
h7 h7-eth0:s5-eth2
h8 h8-eth0:s5-eth3
h9 h9-eth0:s6-eth1
h10 h10-eth0:s7-eth1
h11 h11-eth0:s7-eth2
h12 h12-eth0:s8-eth1
*** Starting controller
c0
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8 ...
*** Starting CLI:
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=8.81 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.980 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.144 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.144/3.311/8.809/3.902 ms
mininet> h3 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=8.87 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=1.14 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.145 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.138 ms
^C
--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 0.138/2.572/8.872/3.659 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
mininet>
```

4.3

علت این است که سوییچ ها در ابتدا باید مسیریابی به مقصد را نمیشناسند و باید با ارتباط گرفتن با کنترلر، FlowTable خود را آپدیت کنند

4.4