Lab 4

Siyuan Li, N16908535, sl6462

Karl Gharbi,N15820300, kg1454

a) Write a pseudo code to implement spanning tree in SDN network.

Step 1: Let Spanning Tree T defined to be an empty set
Step 2: For each vertex v of Graph G, make the empty set out of v
Step 3: Sort the (u,v) edges of G in the ascending order
Step 4: For each edge (u, v) from the sorted list of Step 3
    a) If u and v belong to different sets, then add (u,v) to T
    b) Get together u and v in one single set
Step 5: Return T

b) List the advantages of using OpenVSwitch and SDN controller compared to IP networks.

Advantages of OpenVSwitch:

1. OpenVSwitch supports both configuring and migrating : slow (configuration) and fast network state between instances.
2. OpenVSwitch state is typed and backed by a real data-model and hence allows for the development of structured automation systems.
3. OpenVSwitch is designed such that managing VM network configuration and monitor state spread across many physical hosts in dynamic virtualized environments.
4. It supports a number of features that allow a network control system to respond and adapt as the environment changes including simple accounting and visibility support like NetFlow.
5. OpenVSwitch includes multiple methods for specifying and maintaining tagging rules, all of which are accessible to a remote process for orchestration.

Advantages of SDN :

1. Software-defined networking is an architecture which is dynamic, manageable, cost- effective, and adaptable, seeking to be suitable for the high-bandwidth, dynamic nature of today's applications.
2. Network control is directly programmable because it is decoupled from forwarding functions.

3. Agile : Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
4. Centrally managed : Network intelligence is logically centralized in SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
5. Programmatically configured : SDN lets network managers configure , manage, secure and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend

c) Controller file: attached in zip file

d) Topology file: also attached in zip file

```
class CustomizedTopo(Topo):
    "topo for lab4:controller"
    def build(self):

        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')
        s4 = self.addSwitch('s4')


        h1 = self.addHost('h1', ip = '10.0.0.1/24', mac = '00:00:00:00:00:01')
        h2 = self.addHost('h2', ip = '10.0.0.2/24', mac = '00:00:00:00:00:02')
        h3 = self.addHost('h3', ip = '10.0.0.3/24', mac = '00:00:00:00:00:03')
        h4 = self.addHost('h4', ip = '10.0.0.4/24', mac = '00:00:00:00:00:04')

        self.addLink(h1, s1, 1, 1)
        self.addLink(h2, s2, 1, 1)
        self.addLink(h3, s3, 1, 1)
        self.addLink(h4, s4, 1, 1)
        self.addLink(s1, s2, 2, 2)
        self.addLink(s1, s4, 3, 2)
        self.addLink(s2, s3, 3, 2)
        self.addLink(s3, s4, 3, 3)

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

e) Describe how you generate traffic to test your controller and switch behavior :

ICMP traffic: simply use pingall command in mininet;

HTTP traffic: let one host act as a HTTP server, another host as a client;

Or just iperf h1 h3

```
mininet> h4 python -m SimpleHTTPServer 80 &
mininet> h2 iperf -c h4 -p 80
```

UDP traffic: use iperfudp command to generate UDP packets between hosts.

f) Screenshots:

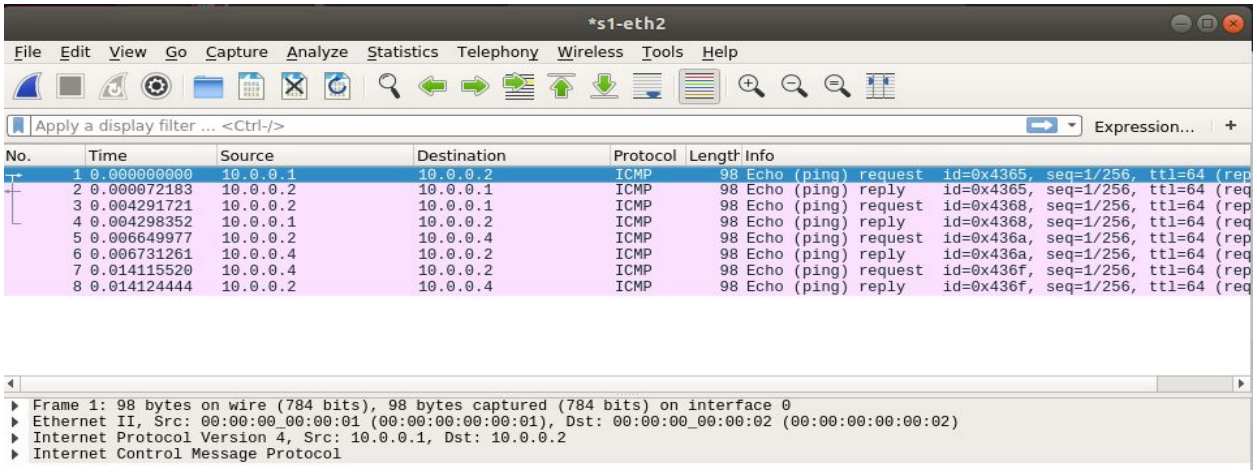
(1)Ping among all the hosts after setting up the platform.

```
siyuan@siyuan-VirtualBox:~/Desktop/Data Center and Cloud Computing/Lab4$ sudo mn --custom topo.p
y --topo mytopo --mac --switch ovsk --controller remote
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s1, s2) (s1, s4) (s2, s3) (s3, s4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

(2)TCP, UDP and ICMP packets on their respective paths.

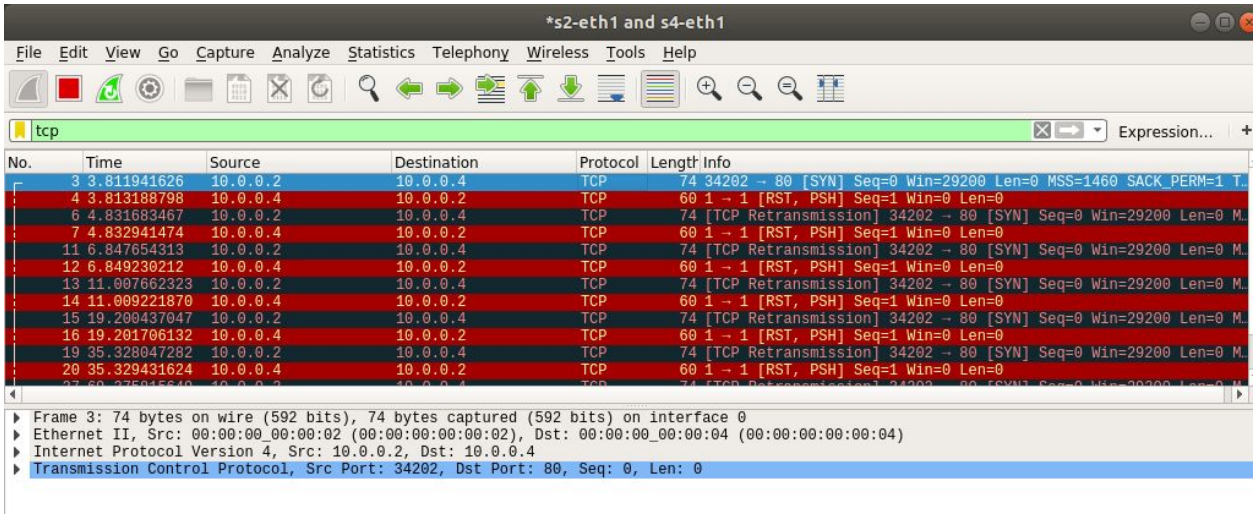
ICMP:

By listening on link between S1 and S2, we can see ICMP traffic between S1--S2, S1--S4, S2--S4, but nothing between S1--S3, which is exactly what we are asked to do.

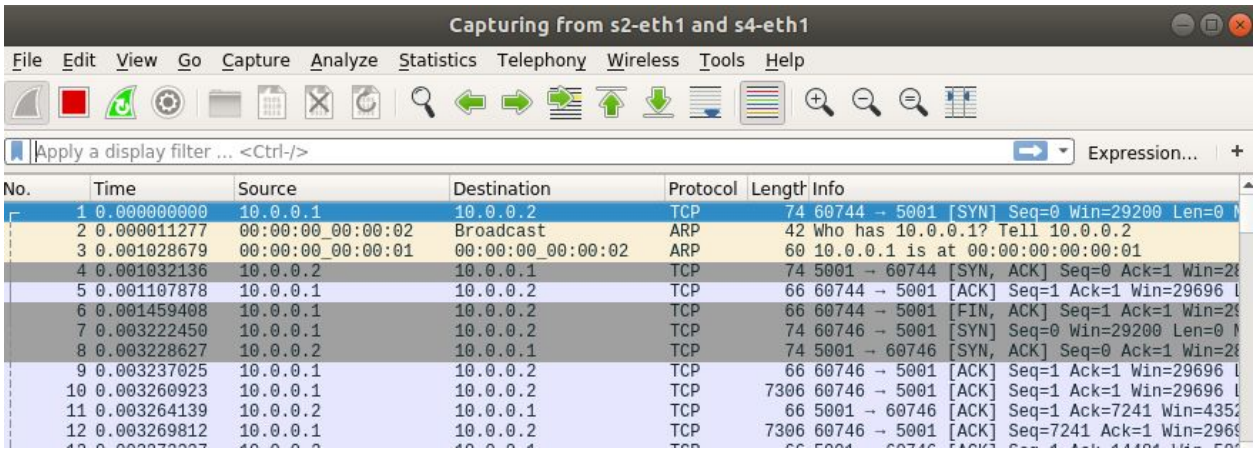


TCP:

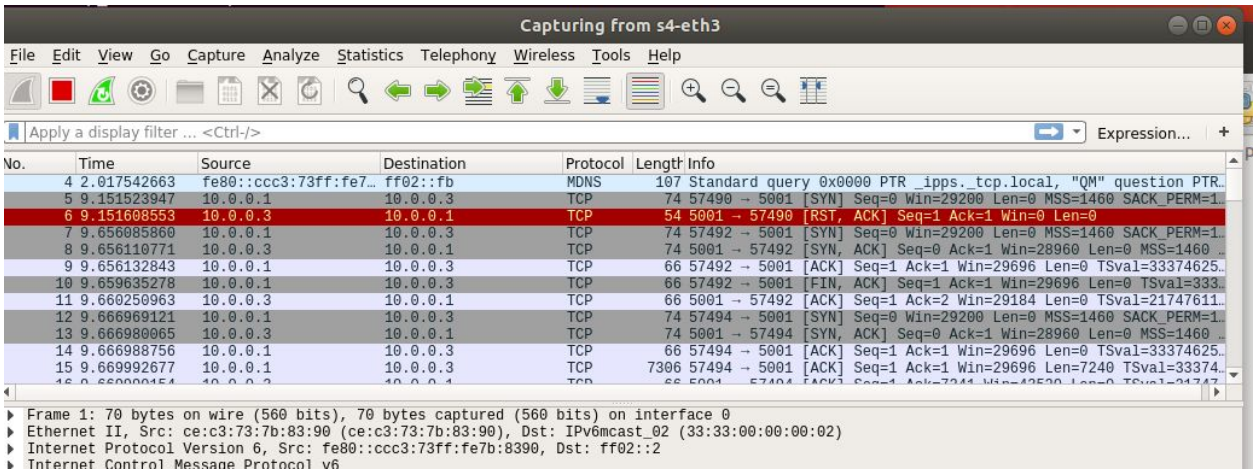
By listening on S2 and S4, we can see that TCP connection with DST port 80 is send back with RST = 1 for TCP between h2 and h4.



But for h1 and h2, the TCP connection can be established normally.

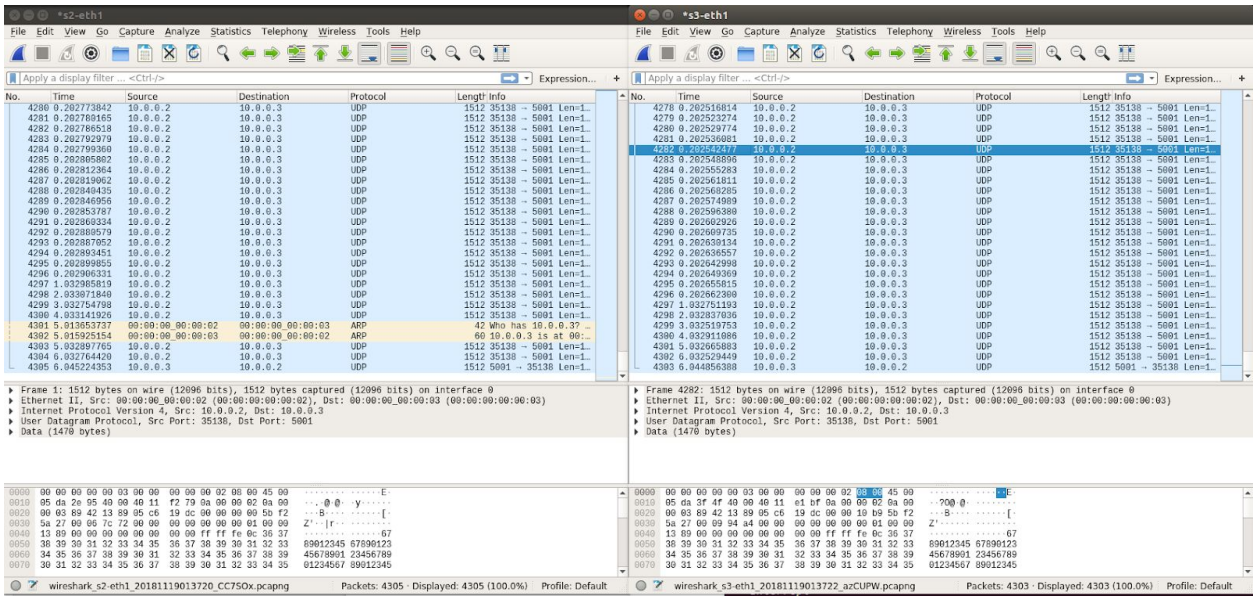


For h1 and h23, the TCP connection go through S3, as we expected.

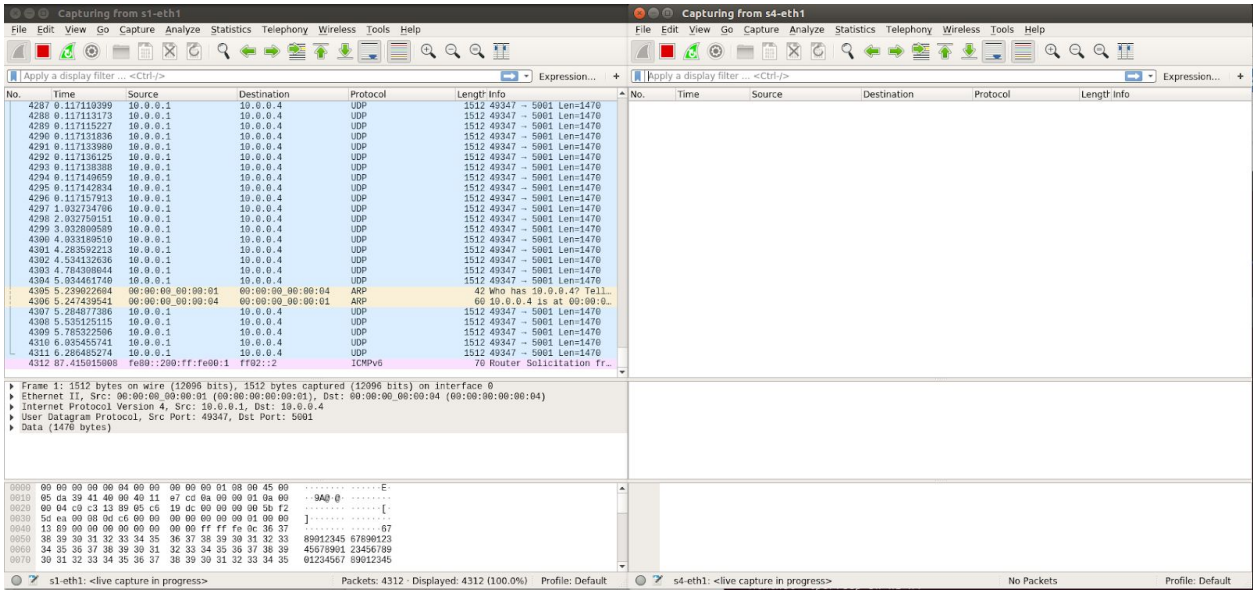


UDP:

To test UDP traffic, the iperfudp command was used. Traffic between h2 and h3, as well as between h1 and h4 was tested.



Traffic between h2 and h3 are able to traverse as expected.



Traffic originating at h1 shows up, but h4 does not see anything, as the traffic is dropped at s1.

(3)Rules installed at each switch

```
mininet> sh ovs-ofctl dump-flows s1 --protocol=OpenFlow13
 cookie=0x0, duration=2359.853s, table=0, n_packets=11, n_bytes=1069, priority=10,tcp,nw_dst=10.0.0.1 actions=output:"s1-eth1"
 cookie=0x0, duration=2359.853s, table=0, n_packets=11, n_bytes=1069, priority=10,tcp,nw_dst=10.0.0.2 actions=output:"s1-eth2"
 cookie=0x0, duration=2359.853s, table=0, n_packets=9, n_bytes=73002, priority=10,tcp,nw_dst=10.0.0.4 actions=output:"s1-eth3"
 cookie=0x0, duration=2359.853s, table=0, n_packets=9, n_bytes=73002, priority=10,tcp,nw_dst=10.0.0.3 actions=output:"s1-eth3"
 cookie=0x0, duration=2359.853s, table=0, n_packets=6, n_bytes=588, priority=10,icmp,nw_dst=10.0.0.1 actions=output:"s1-eth1"
 cookie=0x0, duration=2359.853s, table=0, n_packets=4, n_bytes=392, priority=10,icmp,nw_dst=10.0.0.2 actions=output:"s1-eth2"
 cookie=0x0, duration=2359.853s, table=0, n_packets=4, n_bytes=392, priority=10,icmp,nw_dst=10.0.0.4 actions=output:"s1-eth3"
 cookie=0x0, duration=2359.853s, table=0, n_packets=2, n_bytes=196, priority=10,icmp,nw_dst=10.0.0.3 actions=output:"s1-eth3"
 cookie=0x0, duration=2359.853s, table=0, n_packets=0, n_bytes=0, priority=10,udp actions=drop
 cookie=0x0, duration=2359.853s, table=0, n_packets=86, n_bytes=8928, priority=0 actions=CONTROLLER:65535
```

At S1, we simply drop every UDP packet, since H1 and H4 can't have UDP, and H2 and H3 UDP packets won't show up here.(Same thing for H4)

```
mininet> sh ovs-ofctl dump-flows s2 --protocol=OpenFlow13
 cookie=0x0, duration=23.807s, table=0, n_packets=0, n_bytes=0, priority=30,tcp,nw_src=10.0.0.2,nw_dst=10.0.0.4 actions=CONTROLLER:65535
 cookie=0x0, duration=23.807s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.2 actions=output:"s2-eth1"
 cookie=0x0, duration=23.807s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.1 actions=output:"s2-eth2"
 cookie=0x0, duration=23.807s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.3 actions=output:"s2-eth3"
 cookie=0x0, duration=23.807s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.4 actions=output:"s2-eth2"
 cookie=0x0, duration=23.807s, table=0, n_packets=6, n_bytes=588, priority=10,icmp,nw_dst=10.0.0.2 actions=output:"s2-eth1"
 cookie=0x0, duration=23.807s, table=0, n_packets=2, n_bytes=196, priority=10,icmp,nw_dst=10.0.0.1 actions=output:"s2-eth2"
 cookie=0x0, duration=23.807s, table=0, n_packets=2, n_bytes=196, priority=10,icmp,nw_dst=10.0.0.3 actions=output:"s2-eth3"
 cookie=0x0, duration=23.807s, table=0, n_packets=2, n_bytes=196, priority=10,icmp,nw_dst=10.0.0.4 actions=output:"s2-eth2"
 cookie=0x0, duration=23.807s, table=0, n_packets=0, n_bytes=0, priority=10,udp,nw_dst=10.0.0.2 actions=output:"s2-eth1"
 cookie=0x0, duration=23.807s, table=0, n_packets=0, n_bytes=0, priority=10,udp,nw_dst=10.0.0.1 actions=output:"s2-eth2"
 cookie=0x0, duration=23.808s, table=0, n_packets=0, n_bytes=0, priority=10,udp,nw_dst=10.0.0.3 actions=output:"s2-eth3"
 cookie=0x0, duration=23.808s, table=0, n_packets=0, n_bytes=0, priority=10,udp,nw_dst=10.0.0.4 actions=output:"s2-eth3"
 cookie=0x0, duration=23.808s, table=0, n_packets=49, n_bytes=5832, priority=0 actions=CONTROLLER:65535
```

At S2, we specially send TCP packet from H2 to H4 to controller.

```
mininet> sh ovs-ofctl dump-flows s3 --protocol=OpenFlow13
 cookie=0x0, duration=155.699s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.3 actions=output:"s3-eth1"
 cookie=0x0, duration=155.699s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.2 actions=output:"s3-eth2"
 cookie=0x0, duration=155.699s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.4 actions=output:"s3-eth3"
 cookie=0x0, duration=155.699s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.1 actions=output:"s3-eth3"
 cookie=0x0, duration=155.699s, table=0, n_packets=6, n_bytes=588, priority=10,icmp,nw_dst=10.0.0.3 actions=output:"s3-eth1"
 cookie=0x0, duration=155.699s, table=0, n_packets=2, n_bytes=196, priority=10,icmp,nw_dst=10.0.0.2 actions=output:"s3-eth2"
 cookie=0x0, duration=155.699s, table=0, n_packets=2, n_bytes=196, priority=10,icmp,nw_dst=10.0.0.4 actions=output:"s3-eth3"
 cookie=0x0, duration=155.699s, table=0, n_packets=2, n_bytes=196, priority=10,icmp,nw_dst=10.0.0.1 actions=output:"s3-eth3"
 cookie=0x0, duration=155.699s, table=0, n_packets=0, n_bytes=0, priority=10,udp,nw_dst=10.0.0.3 actions=output:"s3-eth1"
 cookie=0x0, duration=155.699s, table=0, n_packets=0, n_bytes=0, priority=10,udp,nw_dst=10.0.0.2 actions=output:"s3-eth2"
 cookie=0x0, duration=155.699s, table=0, n_packets=0, n_bytes=0, priority=10,udp,nw_dst=10.0.0.4 actions=output:"s3-eth3"
 cookie=0x0, duration=155.699s, table=0, n_packets=0, n_bytes=0, priority=10,udp,nw_dst=10.0.0.1 actions=output:"s3-eth2"
 cookie=0x0, duration=155.699s, table=0, n_packets=61, n_bytes=6838, priority=0 actions=CONTROLLER:65535
```

At S3, nothing interesting, simply forwarding as asked.

```
mininet> sh ovs-ofctl dump-flows s4 --protocol=OpenFlow13
 cookie=0x0, duration=185.340s, table=0, n_packets=0, n_bytes=0, priority=30,tcp,nw_src=10.0.0.4,nw_dst=10.0.0.2 actions=CONTROLLER:65535
 cookie=0x0, duration=185.340s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.4 actions=output:"s4-eth1"
 cookie=0x0, duration=185.340s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.1 actions=output:"s4-eth2"
 cookie=0x0, duration=185.340s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.3 actions=output:"s4-eth3"
 cookie=0x0, duration=185.340s, table=0, n_packets=0, n_bytes=0, priority=10,tcp,nw_dst=10.0.0.2 actions=output:"s4-eth2"
 cookie=0x0, duration=185.340s, table=0, n_packets=6, n_bytes=588, priority=10,icmp,nw_dst=10.0.0.4 actions=output:"s4-eth1"
 cookie=0x0, duration=185.340s, table=0, n_packets=4, n_bytes=392, priority=10,icmp,nw_dst=10.0.0.1 actions=output:"s4-eth2"
 cookie=0x0, duration=185.340s, table=0, n_packets=4, n_bytes=392, priority=10,icmp,nw_dst=10.0.0.3 actions=output:"s4-eth3"
 cookie=0x0, duration=185.340s, table=0, n_packets=2, n_bytes=196, priority=10,icmp,nw_dst=10.0.0.2 actions=output:"s4-eth2"
 cookie=0x0, duration=185.340s, table=0, n_packets=0, n_bytes=0, priority=10,udp actions=drop
 cookie=0x0, duration=185.340s, table=0, n_packets=64, n_bytes=7080, priority=0 actions=CONTROLLER:65535
```

At S2, we specially send TCP packet from H4 to H2 to controller.

g)Challenges you've encountered while doing this experiment, and explain how you manage to solve them. If you do not experience any problem, simply say no problems.

It took a lot of time to understand the objects, methods and variables in RYU, made some mistakes at the beginning. RYU lacks comprehensive documentation of its source code, meaning that it was challenging to discover all the functions needed to add flows. Thankfully,

there were many examples of RYU code available online. With all the different examples, similarities between them enabled an understanding of how to set up the controller.