Lab3 Report: SDN Open Virtual Switches

** Please **fill in the report** and submit the **pdf** to NYU Brightspace*

Name: Yihua Yang, Ziyi Liang ID: yy5028, zl5604 Date: 3/5/2024

1. Objectives

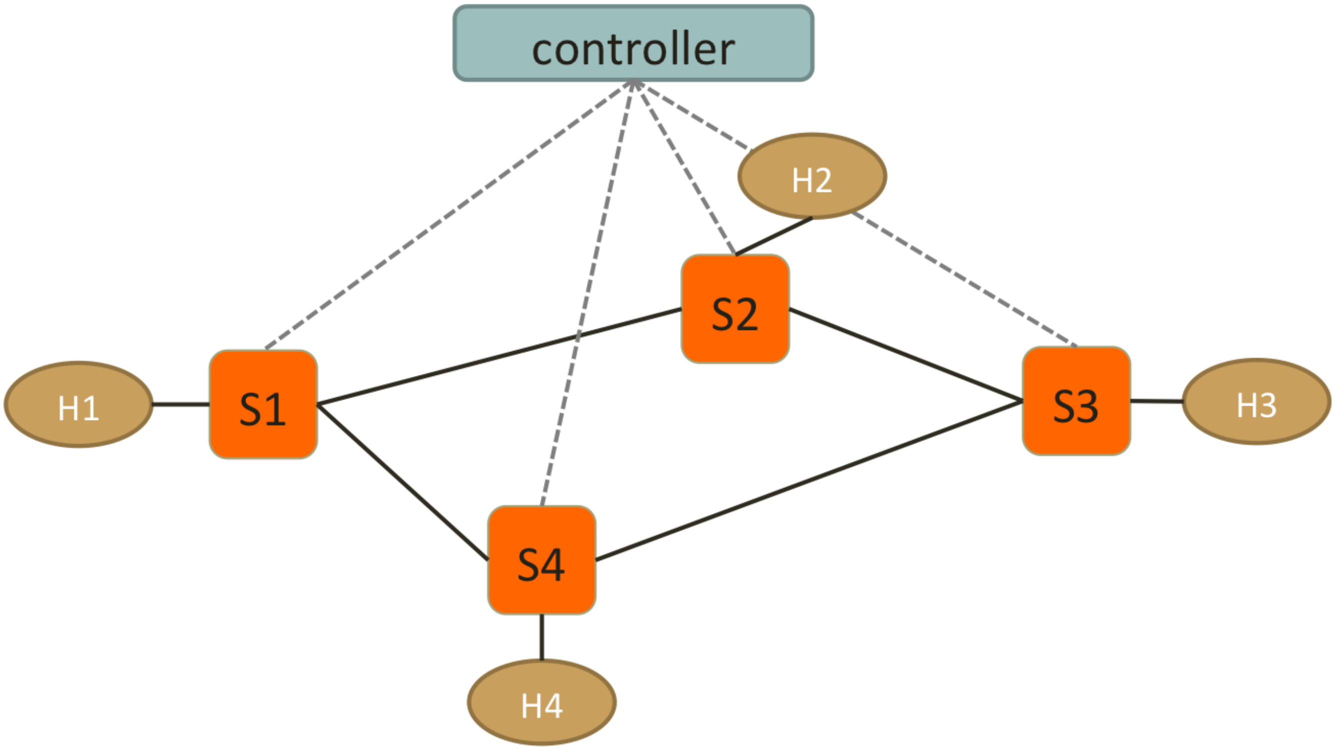
- Understand SDN and get familiar with controllers.

2. References

- https://github.com/faucetsdn/ryu/blob/master/ryu/app/simple_switch_13.py
- https://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html
- Slides

3. Experiments

1. Use Mininet to create the following topology: (4 Hosts, 4 OVSes) with a remote controller
2. Use RYU to implement the controller (you can use other controller such as BEACON, POX, etc...)



3. Test Connectivity using ping. (Hint: take care of ARP packets in the controller and install proper rules for them.)
4. Enforce *these policies*:
    - **Everything follows shortest path**
    - **When there are two shortest paths with equal costs available**
        - ICMP and TCP packets take the clockwise path
            - e.g. S1-S2-S3, S2-S3-S4
        - UDP packets take the counterclockwise path
            - e.g. S1-S4-S3, S2-S1-S4
        - H2 and H4 cannot send HTTP traffic (TCP with dst_port:80)
            - New connections are dropped with a TCP RST sent back to **H2 or H4**
            - To be more specific, when the first TCP packet (SYN) arrives **S2 or S4**, forwarded it to controller, controller then create a RST packet and send it back to the host.
        - H1 and H4 cannot send UDP traffic
            - simply drop packets at switches

Important! Handle the flow rules in Packet-In and let the controller handles the rules dynamically.

If you use static rules for those policies or handle them in SwitchFeatureHandler, your lab score will be removed.

4. Reports

(a) Screenshots of your mininet with "pingall", **before** and **after starting the controller**.

(b) How do you generate different traffic? Which tools do you use to generate: ICMP, TCP, UDP and HTTP traffic?

(c) Generate ICMP flows from H4 to H3, and take **screenshots** of the flow table on **S2** and S3 before and after the flow is generated to show that your flow follow the right path. (ovs-ofctl dump-flows)

	Before ICMP flow is generated	After ICMP flow is generated
S2		

S3		

(d) Generate TCP flows (dst_port: 8080) from H4 to H2, and take **screenshots** of the flow table on S1 and S3 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the TCP traffic.

	Before TCP flow is generated	After TCP flow is generated
S1		
S3		
	Generates TCP traffic	Receives TCP traffic
Mininet or hosts		

(e) Generate UDP flows from H2 to H4, and take **screenshots** of the flow table on S1 and S3 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the UDP traffic.

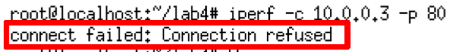
	Before UDP flow is generated	After UDP flow is generated
S1		
S3		
	Generates UDP traffic	Receives UDP traffic
Mininet or hosts		

(f) Generate HTTP traffic from H2 to H1, and take **screenshots** of the flow table on S2 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the HTTP traffic.

	Before HTTP flow is generated	After HTTP flow is generated
S2		
	Generates HTTP traffic	Receives HTTP traffic

Mininet or hosts		

Note: “**Connection refused**” means the RST packets is successfully sent back to S2. Otherwise, you need to check if your RST packets is correct. e.g.,



```
root@localhost:~/lab4# iperf -c 10.0.0.3 -p 80
connect failed: Connection refused
```

(g) Generate UDP traffic from **H4 to H2**, and take **screenshots** of the flow table on S4 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the UDP traffic.

	Before UDP flow is generated	After UDP flow is generated
S4		
	Generates UDP traffic	Receives UDP traffic
Mininet or hosts		

(h) Please find what is “Spanning Tree” and “Spanning Tree Protocol”? What’s the purpose of the protocol?

Spanning Tree is a graph that includes all the vertices of the graph with the minimum possible number of edges. It will form a tree that spans all the vertices in the graph, so loops can be prevented.

Spanning Tree Protocol is a network protocol that organizes the network’s switches and bridges to dynamically form a spanning tree and block any additional paths that could potentially create loops. It ensures that there is always only one active path between any two network devices, which avoids the occurrence of cycles.

(i) Is it necessary to implement spanning tree in SDN for packet forwarding? Why?

It is not necessary to implement spanning tree in SDN for packet forwarding, because SDN can view the entire network topology and stats and make decisions to prevent loops. Both of them serve similar functions, so there is no need to implement both of them on the same network.

(j) If you want to find spanning tree in SDN, how will you implement and what is the difference between traditional “Spanning Tree Protocol” and the one in SDN?

The spanning tree in SDN can be done with minimal spanning tree algorithms like Kruskal or Prim algorithm. Such algorithms converge faster than STP and the SDN controller can switch between different algorithms to satisfy different need. Also, since SDN control all nodes in a centralized manner, the spanning tree in SDN can manage larger network, and be more scalable than the traditional one. In conclusion, spanning tree in SDN is more centralized controlled, have better flexibility, and have better efficiency than traditional STP.

(k) List three advantages of using OpenVSwitch and SDN controller compared to IP networks. Briefly explain why

(l) Include the controller’s code.

(Upload with your report or attach a sharable link)

(m) Include the topology file

(Upload with your report or attach a sharable link)

(n) 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 problem.

We have zero tolerance to forged or fabricated data!! A single piece of forged/fabricated data would bring the total score down to zero.