Lab3 Report: SDN Open Virtual Switches

\* Please **fill in the report** and submit the **pdf** to NYUClasses

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name: |  | ID: |  | Date: |  |

# Objectives

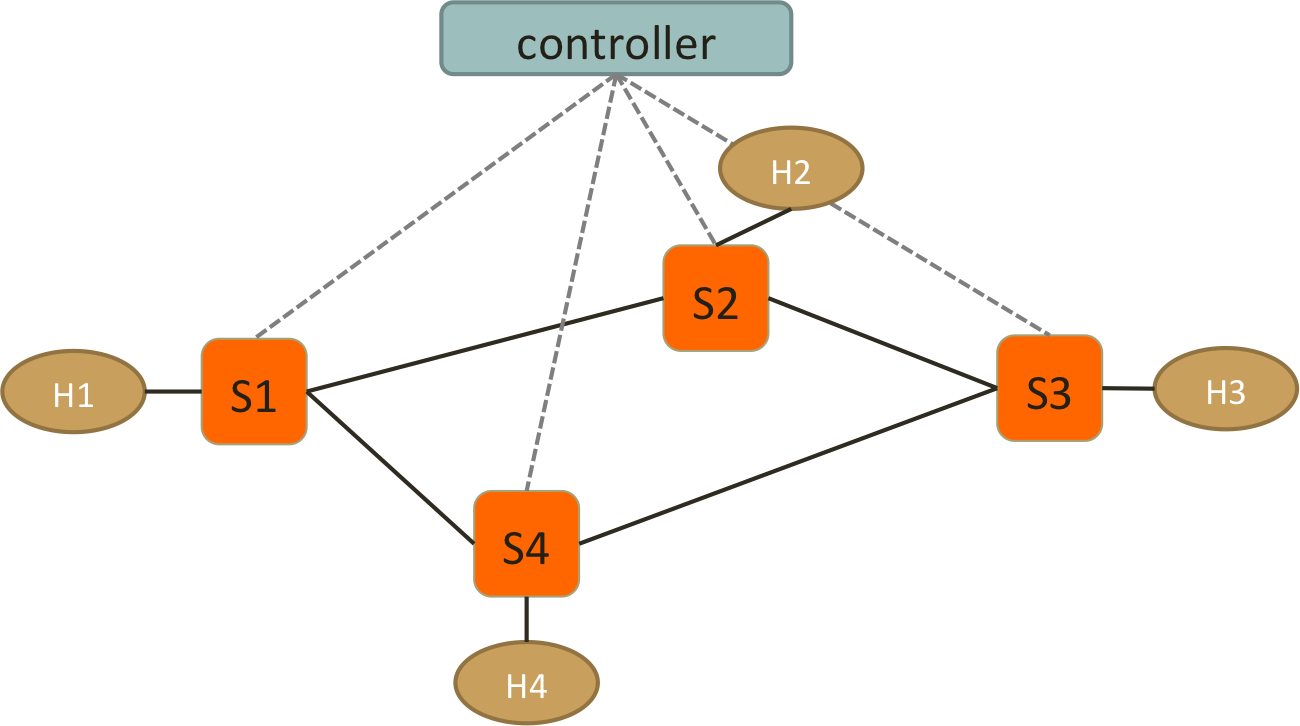
* Understand SDN and get familiar with controllers.

# 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

# 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...)



* 1. Test Connectivity using ping. (Hint: take care of ARP packets in the controller and install proper rules for them.)
  2. 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.**

# Reports

1. Screenshots of your mininet with “pingall”, **before** and **after starting the controller**.

|  |
| --- |
|  |

1. How do you generate different traffic? Which tools do you use to generate: ICMP, TCP, UDP and HTTP traffic?

|  |
| --- |
|  |

1. 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 |  |  |

1. 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.

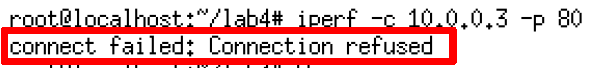
|  |  |  |
| --- | --- | --- |
|  | Before TCP flow is generated | After TCP flow is generated |
| S1 |  |  |
| S3 |  |  |
|  | Generates TCP traffic | Receives TCP traffic |
| Mininet or hosts |  |  |

1. 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 |  |  |

1. 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., 

1. 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 |  |  |

1. Please find what is “Spanning Tree” and “Spanning Tree Protocol”? What’s the purpose of the protocol?

|  |
| --- |
|  |

1. Is it necessary to implement spanning tree in SDN for packet forwarding? Why?

|  |
| --- |
|  |

1. 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?

|  |
| --- |
|  |

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

|  |
| --- |
|  |

1. Include the controller’s code.

|  |
| --- |
| (Upload with your report or attach a sharable link) |

1. Include the topology file

|  |
| --- |
| (Upload with your report or attach a sharable link) |

1. 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.