Lab4: SDN Open Virtual Switches

Name: \_\_\_Ranran Lyu\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_rl3783\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_04/10/2020\_\_\_\_\_\_\_

Name: \_\_\_Lingfeng Zhao\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_lz1973\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_04/10/2020\_\_\_\_\_\_\_

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

Step1: T = new empty set # T is the final spanning tree

Step 2: for each vertex v in G:

make the empty set out of v

Step 3: sort(E) #sort edges of G in ascending order

Step 4: for each edge(u, v) in E:

if set(u) != set(v): # if u and v don’t belong to the same set

add(edge(u ,v)) to T

put u and v in the same set

Step 5: return T

1. **List the advantages of using OpenVSwitch and SDN controller compared to IP networks.**

**Advantages of using OpenVSwitch:**

1. OpenVSwitch has support for both configuring and migrating both slow (configuration) and fast network state between instances.

2. OpenVSwitch supports a number of features that allow a network control system to respond and adapt as the environment changes.

3. OpenVSwitch includes multiple methods for specifying and maintaining tagging rules, all of which are accessible to a remote process for orchestration.

4. OpenVSwitch’s forwarding path (the in-kernel datapath) is designed to be amenable to “offloading” packet processing to hardware chipsets.

5. OpenVSwitch being a software switch can handle more number of flows than hardware switches.

6. OpenVSwitch also supports complex lookups and header manipulations.

**Advantages of using SDN:**

1. Centralised Provisioning.

2. SDN gives the user more scalability.

3. More granular security.

4. Reduced Hardware Footprint

5. Lower operating costs.

6. Cloud Abstraction.

7. Hardware savings and reduced capital expenditures.

1. Include the controller’s code.
2. Include the topology file
3. **Describe how you generate traffic to test your controller and switch behavior**

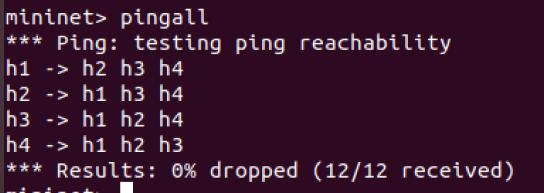
1. In the graph, adjacent switches ping each other to verify “Everything follows shortest path”. For example, p1 ping p2, if it does not follow the shortest path, it will go through path “s1-s4-s3-s2”. Otherwise, it goes through path “s1-s2”.

2. To verify “When there are two shortest paths available, ICMP and TCP packets take the lower/left path, UDP packets take the upper/right path”” H1 and H4 cannot have UDP traffic”, we can set filter to capture packets. For example, for ICMP and TCP packets, h2 to h4 takes path “S2-S1-S4”. So, we use “s1-eth2”(We have already set the port in .py file) as filter in Wireshark and enter “h2 ping h4” in terminal to capture packets. Then we will capture packets. However, if we set filter “s3-eth2”, we will not capture packets.

3. To verify “H2 and H4 cannot have HTTP traffic (TCP with port:80)”, we can enter “h4 python -m http.server 80 &”, “h2 curl h4” in the terminal and then see the result.

1. **Screenshots:**

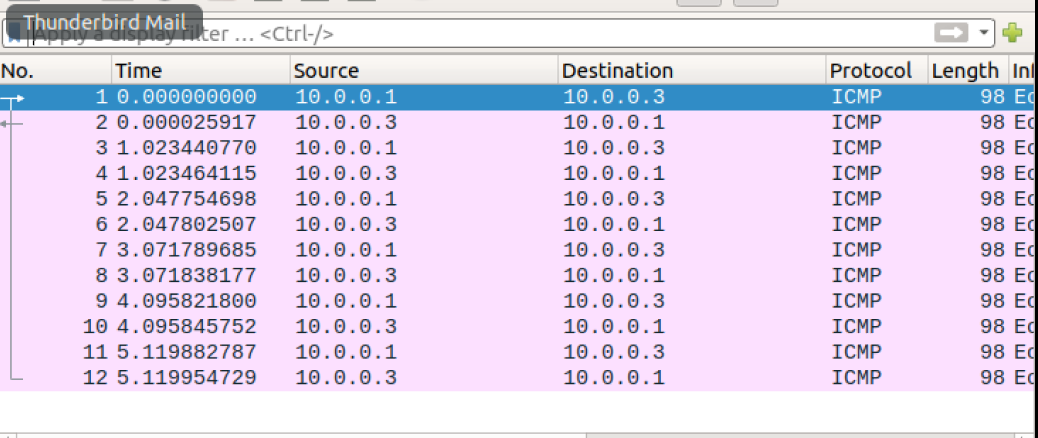
* Ping among all the hosts after setting up the platform.



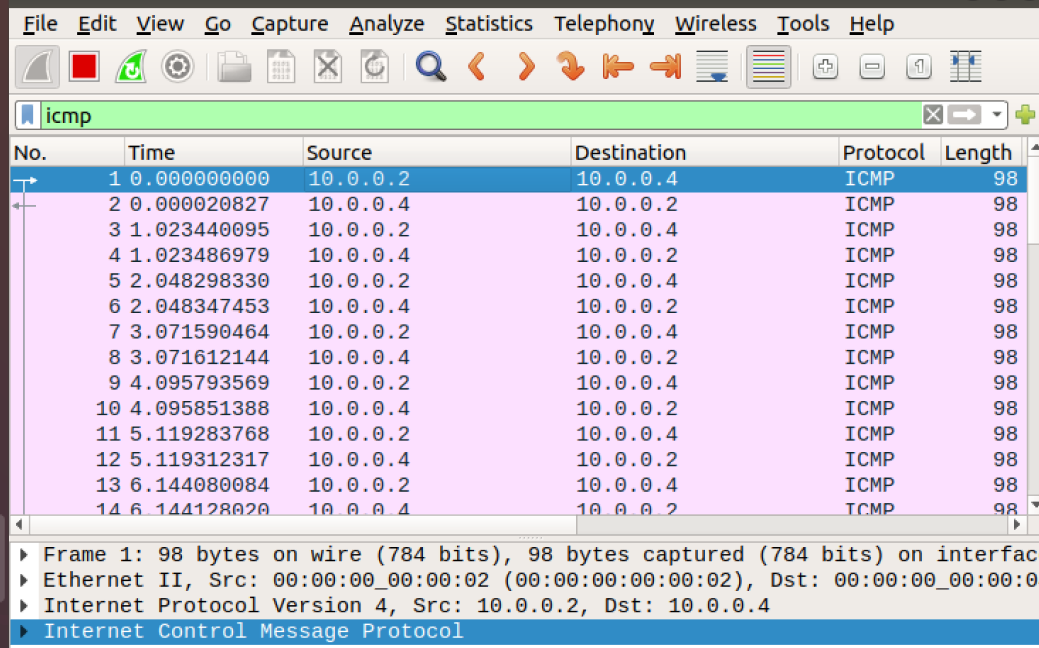
* TCP, UDP and ICMP packets on their respective paths.

ICMP

Set s4-eth2 as filter in Wireshark to verify ICMP path “s1-s4-s3”



Set s1-eth2 as filter in Wireshark to verify ICMP path “s2-s1-s4”



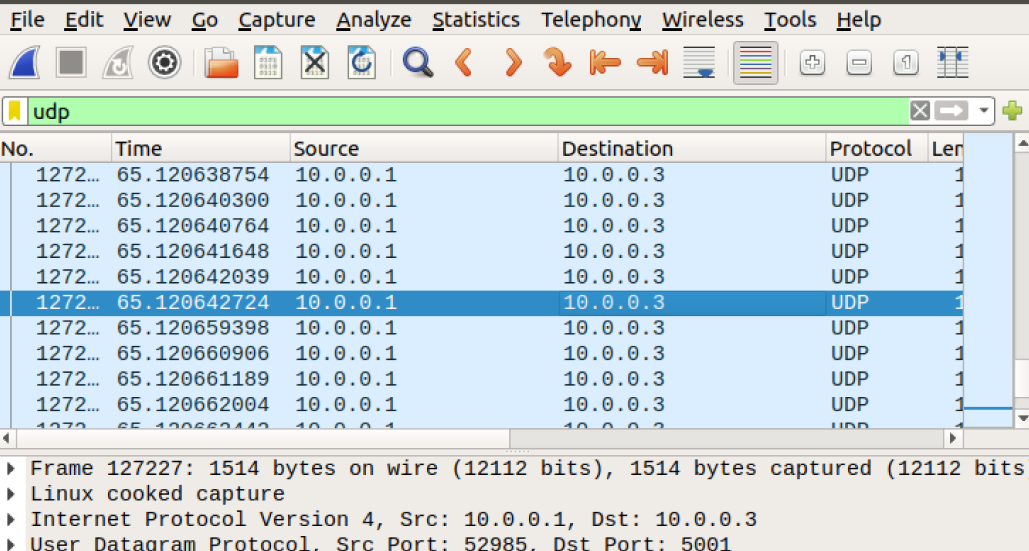
TCP：

Verify H2 and H4 cannot have HTTP traffic (TCP with port:80)

Set s1-eth2 as filter in Wireshark to verify TCP path “s2-s1-s4”

UDP:

Set s2-eth2 as filter in Wireshark to verify UDP path “s1-s2-s3”



Verify H1 and H4 cannot have UDP traffic

* Rules installed at each switch.

Switch 1

Switch 2

Switch 3

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 problems.