CEG4188 Lab 5 Part 1 Report

Griffin Taylor – 300122587

Exercise 1

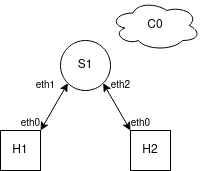
1. The default topology has 4 nodes. 2 hosts, 1 switch, and 1 ch ovs-ofctl add-flow s1 tcp,tcp\_dst=80,actions=output:3ontroller



2. There is 1 switch and 1 controller

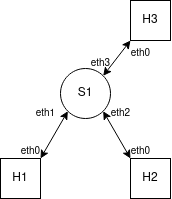


3.

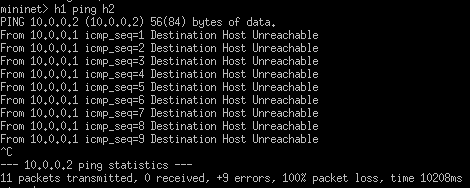


Exercise 2

4.



5. The ping does not work because since there are no flow rules setup for traffic between h1 and h2



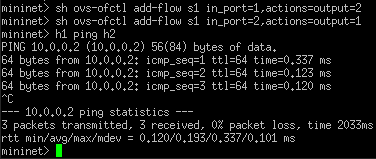
6. The flow table for S1 is empty.

7. Since there is no controller, the flows have to be set manually to allow for traffic.

8. ip\_src looks at the IPv4 header’s source IP. tcp\_src looks at the TCP header’s source port field. eth\_src looks at the Ethernet header’s source address field to get the source MAC address

9. By executing those commands it adds 2 flows to the switch, one to forward any packets from its input port 1 out of output port 2, and one to forward packets from input port 2 to output port 1.

10. The ping does work, since the rules above specify that any packets incoming to eth1 on the switch are to be forwarded out of eth2.

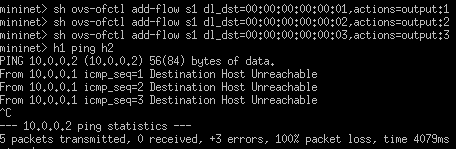
Exercise 3

11. The MAC addresses are H1: 00:00:00:00:00:01, H2: 00:00:00:00:00:02, and H3: 00:00:00:00:00:03

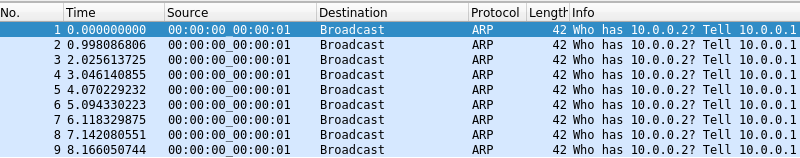




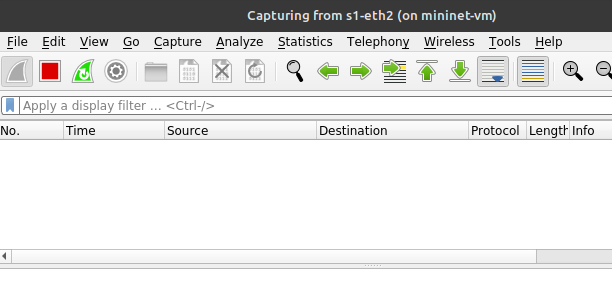
12. The ping did not work.



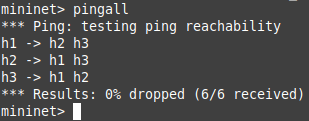
13. There are ARP broadcasts but no replies.



14. There are no ARP packets reaching S1-eth2

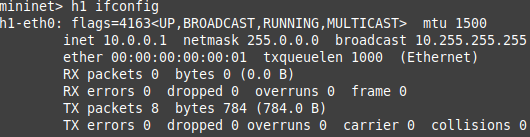


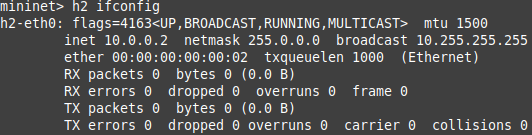
15. The ARP broadcasts are never reaching H2, so H1 never receives info on where to send the ICMP packets. The ARP traffic is being blocked in S1.

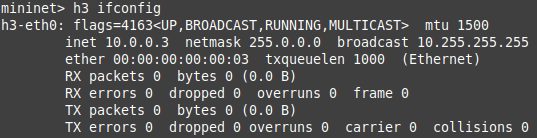
16. 

Exercise 4

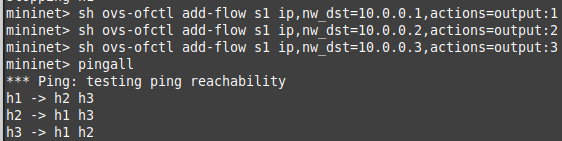
17. The IP addresses are H1: 10.0.0.1, H2: 10.0.0.2, and H3: 10.0.0.3





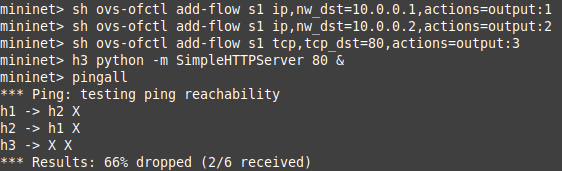


18.

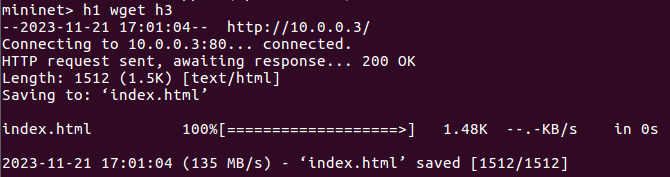


Exercise 5

19.



20. It did work, since the HTTP requests are sent to the server but responses (since the flow rules check from layer 1 -> layer 7) are sent correctly back to the original requester.



Exercise 6

21. I have 4 rules in my setup. The first three are to allow the devices to communicate via Layer 3 (I used –mac and –arp to allow layer 2 automatically), while the 4th is to block TCP traffic with a dst port of 80 from reaching h3’s ip.

1. sh ovs-ofctl ip,nw\_dst=10.0.0.1,actions=output:1

2. sh ovs-ofctl ip,nw\_dst=10.0.0.2,actions=output:2

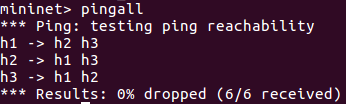
3. sh ovs-ofctl ip,nw\_dst=10.0.0.3,actions=output:3

4. sh ovs-ofctl priority=40000,tcp,nw\_dst=10.0.0.3,tp\_dst=80 actions=drop

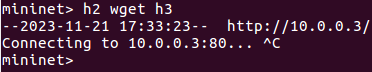
The priority in the last command is set to 40000 since the default priority is ~32k, so putting a higher priority makes sure this rule is checked first.

22.

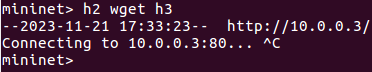
Checking L3 reachability, which passed



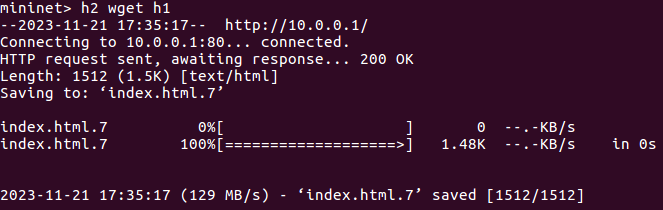
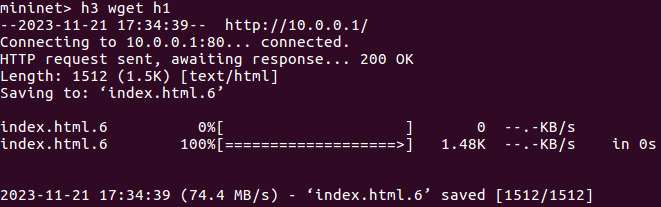
H1 cannot wget from H3, which is running the SimpleHTTPServer



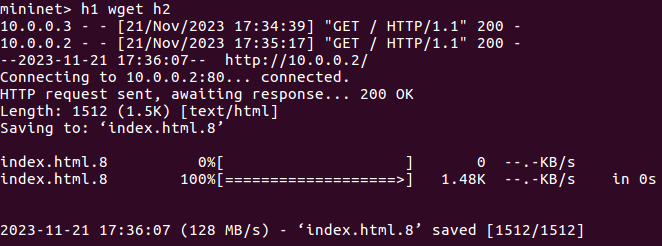
H2 also cannot wget from H3



However, H3 and H2 can wget from H1

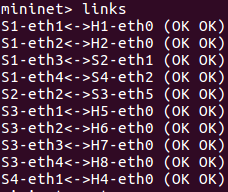


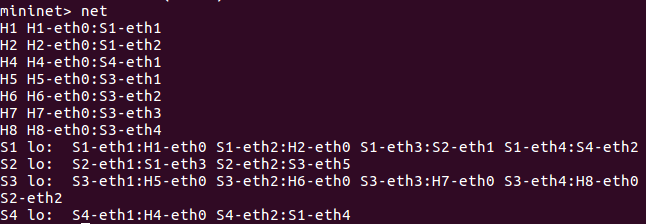
And H1 and H3 can both wget from H2



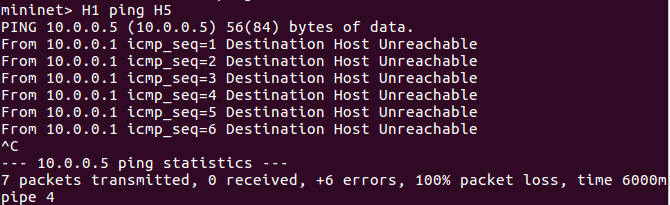
Part 2

2.

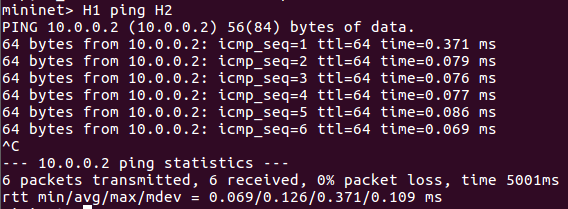




3. The ping does not work since there are no flow rules set yet



4.



To add flow rules for S1:

sh ovs-ofctl add-flow S1 ip,nw\_dst=10.0.0.1, actions=output:1 // Forward packets for 10.0.0.1 to output 1

sh ovs-ofctl add-flow S1 ip,nw\_dst=10.0.0.5, actions=output:3 // Forward packets for 10.0.0.5 to S2 via output 3

sh ovs-ofctl add-flow S1 dl\_type=0x0806,actions=flood // For ARP

To add flow rules for S2:

sh ovs-ofctl add-flow S2 in\_port=1,actions=output:2 // Forward packets from input 1 to output 2

sh ovs-ofctl add-flow S2 in\_port=2,actions=output:1 // Forward packets from input 2 to output 1

To add flow rules for S3:

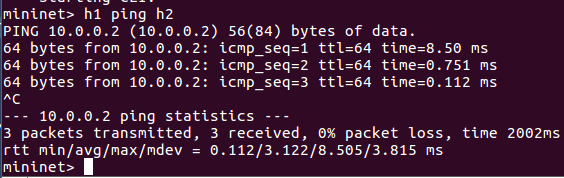
sh ovs-ofctl add-flow S1 ip,nw\_dst=10.0.0.5, actions=output:1 // Forward packets for 10.0.0.5 to output 1

sh ovs-ofctl add-flow S1 ip,nw\_dst=10.0.0.1, actions=output:5 // Forward packets for 10.0.0.1 to S2 via output 5

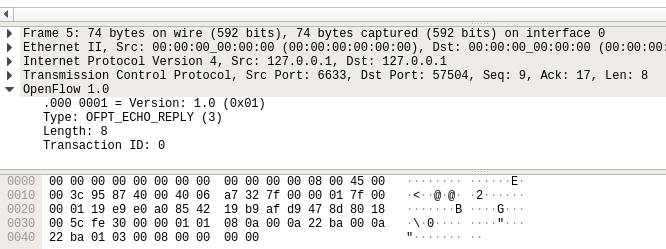
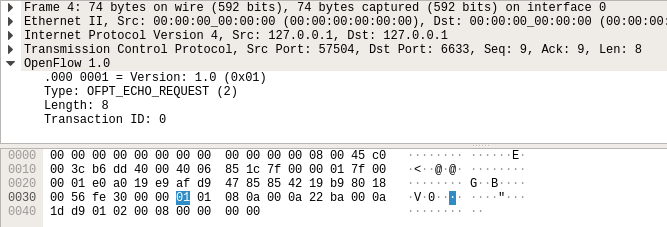
sh ovs-ofctl add-flow S1 dl\_type=0x0806,actions=flood // For ARP

Exercise 2:

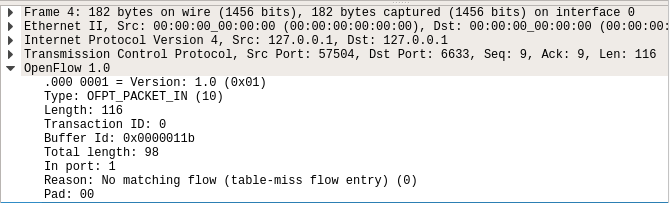
5. Yes it works, the controller automatically added a flow to connect them.

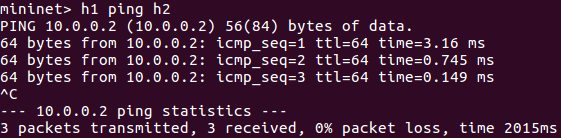
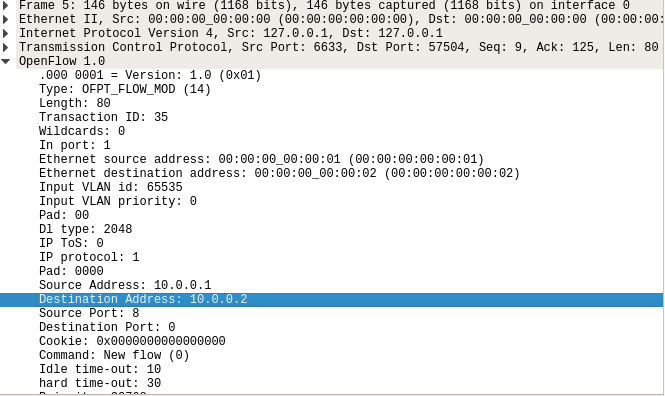


6. There are 2 packets, an OpenFlow Echo Request, and an OpenFlow Echo Reply. The first, the Echo Request, is sent by the mininet VM to the remote controller, trying to see if it is still there/responding. You can see that its going from the VM to the controller since the dst port is 6633, which is one of the 2 ports that the VM will try to reach the controller at by default. The second, the Echo Reply, is the controller confirming that it is still there, and reachable, which confirms the connection between the controller and the network.



7. There are 2 new packets, OpenFlow Packet In and OpenFlow Flow Mod. The Packet In is sent from the network/switch to the controller, asking how to route the packet because there was a miss in the flow table, so the controller needs to decide on the next hop. The following Flow Mod packet is just that, it is adding an entry to the flow table of the switch for the packet. This flow is kept, since past the first packet in the original ping the RTT is very low (~8.5ms down to ~0.75).





8. The behaviour of the switch is to build a map of destination MAC address -> output port within each switch. If there is a miss within the table (map), then it will flood the packet until the correct destination is found, and once it is it will update or add the correct table mapping. There are a couple of other behaviours, namely dropping a packet and a couple following it if the destination MAC address maps to the same port that the packet came in from (loop).