Lab5: Software Defined Networks

Course on Data Communication and Networking

Chalmers and Gothenburg University

by Dimitris Palyvos-Giannas

February 23, 2023

Contents

1 Introduction

This document contains guidance for carrying out the work for the project and assignments in it. Please submit a report per student-group, with the answers to all the questions in the document.

Software Defined Networks In this project, you are going to get hands-on experience on Software Defined Networks (SDNs) and especially the OpenFlow Protocol [1]. To complete the project, it is strongly recommended that you study Chapters 4.4, 5.5 from the coursebook. After reading the chapters, please answer the following questions briefly:

Q1. What is a Software Defined Network (SDN)?

Q2. What is an SDN controller?

Q3. What is an SDN switch?

Q4. What is the *match-plus-action* paradigm and how is it different from *destination-based routing*?

[1] `https://www.opennetworking.org/technical-communities/areas/specification/open-datapath/`

Mininet In order to realistically simulate real-world networks, you will use a virtual machine which is running the mininet network emulator:

> Mininet is a network emulator, or perhaps more precisely a network emulation orchestration system. It runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. It uses lightweight virtualization to make a single system look like a complete network, running the same kernel, system, and user code.
>
> *A Mininet host behaves just like a real machine; you can [...] run arbitrary programs (including anything that is installed on the underlying Linux system.)* The programs you run can send packets through what seems *like a real Ethernet interface, with a given link speed and delay.* Packets get processed by what looks like a real Ethernet switch, router, or middlebox, with a given amount of queueing.[2]

VM Setup To get started, download the virtual machine image[3] and load it into a virtualization system such as VirtualBox, VMWare, etc. You can log in to the virtual machine as user `mininet` with password `mininet`. It is strongly suggested that you complete the first part of the *Mininet Walkthrough*[4] before starting the actual work. If you run into trouble, you can always consult the documentation[5].

2 Creating Flows and using a Controller

In the first part of the project, you will experiment with a simple topology, create some very basic flows, and observe how an OpenFlow controller works. From here onwards, it is assumed that you have started the virtual machine and logged in.

2.1 Experimenting with Custom Flows

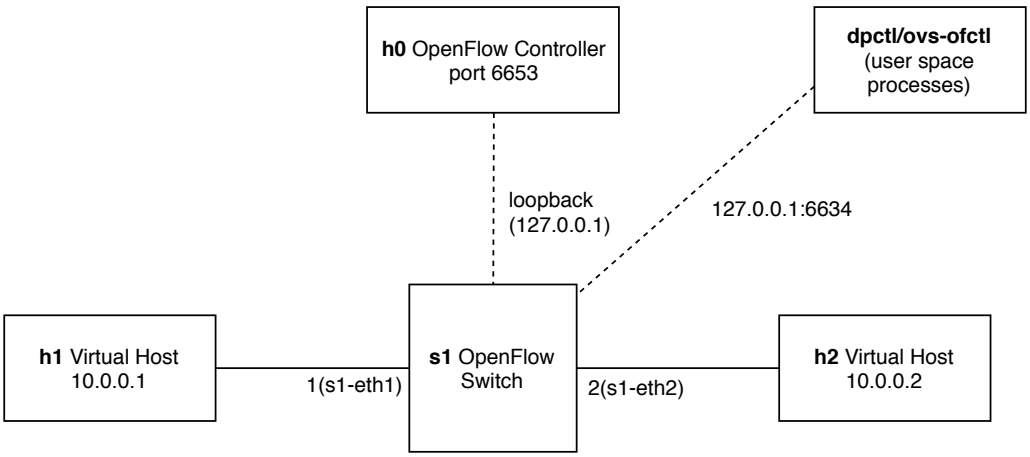

Figure 1: Simple Topology with one vSwitch and two nodes.

To begin with, launch a terminal in the guest machine and run the following command:

```
sudo mn --topo single,2 --mac --switch ovsk,port=6634 --controller remote,port=6653
```

Note: If mininet crashes at any point, run `sudo mn -c` to clean the old state before restarting it.

2.1.1 Exploring the topology

The above command instructs mininet to generate a new topology with two virtual hosts connected to the same virtual switch (which can be reached at TCP port 6634). To ease understanding, we make the

[2] `https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what`
[3] `https://chalmersuniversity.box.com/s/f8dtdvy1moba2sr242p36fgdw6g9mopd`
[4] `http://mininet.org/walkthrough/#part-1-everyday-mininet-usage`
[5] `https://github.com/mininet/mininet/wiki/Documentation`

MAC addresses of the hosts match their IPs (e.g., IP 10.0.0.1 → MAC 00:00:00:00:00:01). Lastly, we instruct mininet not to use its internal SDN controller but to use a custom one which should be reachable at TCP port 6653.

Figure 1 depicts the components of the topology. Notice that, when using OpenFlow, the network interfaces of the switch (also called ports) can be referred to using their full name (e.g., s1-eth1) or just their number (e.g., 1).

Mininet offers multiple tools to report the structure and state of the network. Run the following commands inside the mininet console and report their output:

Q5. `dump`

Q6. `links`

Additionally, you can run regular Linux commands on each mininet virtual host and switch using the format `<host-name> <command>`.

Run the following commands inside the mininet console and report their output:

Q7. `h1 ifconfig -a`

Q8. `h2 ifconfig -a`

Q9. `s1 ifconfig -a`

Now it is time to test the connectivity between the hosts. Try to ping h2 from h1 using the commands below (The argument `-c4` means that ping retries four times). What is the result? Explain.

Q10. `h1 ping -c4 h2`

Q11. `h2 ping -c4 h1`

Q12. `pingall`

As mentioned above, when we generated the topology, we instructed mininet to use a custom remote OpenFlow controller. However, right now, no such controller is active, and thus the virtual switch has no flows active. You can verify this by running the command `dpctl dump-flows` in the mininet terminal.

Q13. What is the output of this command?

2.1.2 Adding Custom Flows

To allow the two hosts to communicate, we will add the appropriate flows manually, using the tool `ovs-ofctl`. Leave the mininet terminal running. Start a different terminal in the guest machine and execute the following commands:

```
sudo ovs-ofctl add-flow s1 priority=1,in_port=1,actions=output:2
sudo ovs-ofctl add-flow s1 priority=1,in_port=2,actions=output:1
```

Afterward, observe the flows by issuing the following command in the same terminal:

```
sudo ovs-ofctl dump-flows s1
```

You can get the same information for all the available virtual switches by issuing the following command **in the mininet terminal**:

```
dpctl dump-flows
```

Q14. What is the output of the dump-flows command?

Q15. Describe what each field of the flow table means.

Q16. Check connectivity again using the pingall command. Does it work? Why/why not?

2.2 Experimenting with an OpenFlow controller

In this part, you are going to examine the behavior of the RYU OpenFlow controller. Clear the previous flows by issuing the following command in the mininet terminal:

```
dpctl del-flows
```

Q17. Can the two hosts communicate now? Why/why not?

Launch Wireshark in the guest machine (there is a shortcut on the left shortcut bar) and start capturing the loopback interface (lo). Afterward, in a new terminal of the guest machine, run the following command to start the controller:

```
./simpleSwitch.sh
```

To hide unrelated messages and heartbeats, use the following display filter on Wireshark:

```
openflow_v4 && ! (openflow_v4.type in {2 3})
```

Q18. What types of OpenFlow messages are sent when the controller starts?

Q19. In the mininet console, execute `h1 ping -c4 h2`. Which messages can you see in Wireshark? Which messages alter the flows? Explain their most important fields of these messages.

3 Creating Advanced Flows

In this part of the project, you will have to create your own flows in a more complicated topology. Exit the previous mininet instance using the **`exit`** command in the mininet terminal and exit the controller by pressing `<Ctrl + C>` in its terminal.

Then, start the custom topology with the command:

```
sudo ./mininetSDN.py
```

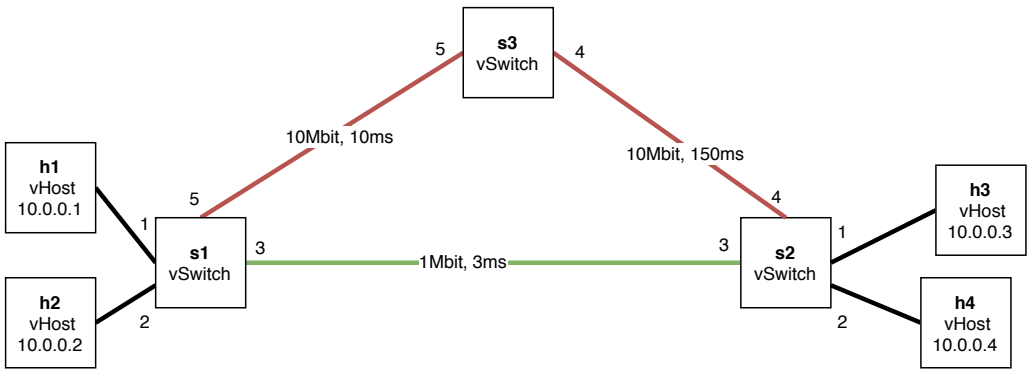

Figure 2: Simple Topology with one vSwitch and two nodes.

Figure 2 shows the layout of the topology, the interface numbers where each host/switch is connected to, the IPs of the hosts as well as the bandwidth and delay of the links between the vSwitches. All links between `vHost` ↔ `vSwitch` are 100Mbps/1ms. Note that interfaces `sX-ethY` are identical to OpenFlow ports `Y`. E.g., `s1-eth1` is the same as `OpenFlow port 1`. *OpenFlow ports have nothing to do with TCP/UDP ports!*.

In the following, you will have to manually create some flows to satisfy the QoS requirements of the network. The Appendix contains additional information that can help you with the syntax of the

messages. It is possible to use WireShark to observe the packets in each interface of the network, which might help debugging.

In order to keep track of your work, it is a good idea to save the commands you use to create your flows in a file called `my-flows.sh` inside the virtual machine. Then, if something goes wrong and you need to retrace your steps, you can just delete all flows and run `bash my-flows.sh` to add all your previously used flows.

In the report, please include the command(s) you executed to create the flows, as well as the contents of all the flow tables (the output of `dpctl dump-flows`) after running all commands of each task.

3.1 Exchanging ARP Messages

Q20. Launch WireShark and start capturing packets on s1-eth1. Try to ping host h4 from host h1 with `h1 ping -c4 h4`. What packets do you observe?

Q21. Explain the purpose of the Address Resolution Protocol (ARP) briefly. How does it relate to the packets you observed?

Q22. One way to propagate ARP information over the network is to have every switch to broadcast each ARP message it receives. What problem(s) will arise in this network if you try to do this?

Q23. Address the problems you identified using OpenFlow. Add the appropriate flows so that all ARP messages (*and no other type of message*) reach their intended destinations using whichever path(s) you prefer. Remember that you can use WireShark on any interface of the network to debug and verify your flows.
Hint: For each switch that you want to use, consider ARP messages arriving from the different input ports and where they should be forwarded in each case (use the picture!).

Q24. Run `pingall` in mininet. Does it succeed? Why/why not? (Remember, the flows you created should only target ARP messages).

Q25. The `pingall` above triggered an exchange of ARP requests and responses, populating the ARP tables of each host. Examine each table by running the command `arp -a` on each of the four hosts in mininet, e.g., `h1 arp -a`. Include the tables in your report and briefly explain the meaning of each column. A correct table will look like this:

```
mininet> h1 arp -a
? (10.0.0.4) at 00:00:00:00:00:04 [ether] on h1-eth0
? (10.0.0.2) at 00:00:00:00:00:02 [ether] on h1-eth0
? (10.0.0.3) at 00:00:00:00:00:03 [ether] on h1-eth0
```

3.2 High Bandwidth Connection

In this part, you need to create the appropriate flow(s) to connect hosts h1 and h4 using the highest bandwidth path. **You should not remove previously added flows, and you should not break exchange of ARP messages**. Keep in mind that you might need to add rules to more than one switch. After you are done, answer the following questions:

Q26. What flow(s) did you add? Report the relevant commands.

Q27. Verify your solution using iperf, e.g., `iperf h1 h4`. What is the reported bandwidth?

Q28. Use ping to measure the latency of the connection. What is the latency? Why?

3.3 Low latency connection

In this part, you need to create the appropriate flow(s) so that hosts h2 and h3 use the lowest latency path. **You should not remove previously added flows, and you should not break the connection**

that you created above, between h1 and h4. Keep in mind that, once again, you might need to add flows in more than one switch. After you are done, answer the following questions:

Q29. What flow(s) did you add? Report the relevant commands.

Q30. Verify your solution using iperf. What is the reported bandwidth?

Q31. Use ping to measure the latency of the connection. What is the latency? Why?

Q32. Run pingall. Are all hosts connected?

Q33. Can you ping h3 from h1? What is the latency? What is the throughput? Explain.

Q34. Try the other direction now, i.e., ping h1 from h3. Does the latency/throughput change? Why/why not? If h1 and h3 cannot communicate, explain very briefly if you could have defined the flows at 3.2, 3.3 differently so that h1 and h3 could communicate through some path, without additional flows. Consult the Appendix for ideas.

3.4 Creating a basic Firewall

Without removing the flows you added previously, create the appropriate flow(s) to block all communication between hosts h1 and h4 for 300 seconds. **Removing existing flows is not an acceptable solution. To make sure that you override any previous flows, use `priority=65535` for the flows you add in this part.**

After you are done, answer the following questions:

Q35. What flow(s) did you add? Report the relevant commands.

Q36. What is the output of `h1 ping h4`? Explain.

Q37. What is the output of `h4 ping h1`? Explain.

Q38. What is the output of `pingall`? Explain.

4 Appendix

4.1 Mininet Commands

If mininet crashes, run `sudo mn -c` to clean the old state before rerunning it.

4.2 Structure of OpenFlow Messages

When we add custom flows without a controller, we use the `ovs-ofctl` tool to send *FlowMod* messages to the controller. The general structure of the command is the following:

`sudo ovs-ofctl add-flow SWITCH priority=X,hard_timeout=Y,`$match_1$`,..,`$match_N$`,actions=`$action_1, .., action_M$

- If a packet matches more than one flow, then that which has the highest priority will be used. If you omit the priority part, the flow will have the **default priority**, that is **32768**.
- The `hard_timeout` defines how how long (in seconds) that flow will be in the table. It can be used when the flow needs to expire after some time.
- Examples of matches and actions that might be useful for this exercise can be seen on Table 1.
- More details about the structure of FlowMod messages can be found online (See Techhub HP support[6,7,8] or OpenFlow Documentation p. 23[9]).
- More details about matches can be found online (Cf. "Match fields"[10] or OpenFlow Documentation p. 22[11]).
- More details about actions can be found online (Cf. "Action List"[12] or OpenFlow Documentation p. 23[13]).

Matches

Description	Match
Coming from OpenFlow port (interface) 2	in_port=2
Destination IP 10.0.0.1	ip,nw_dst=10.0.0.1
Source IP 10.0.0.2	ip,nw_src=10.0.0.2
ARP packet	arp

Actions

Description	Action
Forward to OpenFlow port (interface) 3	output:3
Discard packets	drop

Table 1

[6] https://techhub.hpe.com/eginfolib/networking/docs/switches/5950/5200-4024_openflow_cg/content/index.htm

[7] https://techhub.hpe.com/eginfolib/networking/docs/sdn/sdnc2_7/5200-0910prog/content/c_sdnc-OpenFlow-subjects.html

[8] https://techhub.hpe.com/eginfolib/networking/docs/sdn/sdnc2_7/5200-0910prog/content/s_sdnc-OF-flowmods.html

[9] https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf#page=29

[10] https://techhub.hpe.com/eginfolib/networking/docs/switches/5950/5200-4024_openflow_cg/content/499752685.htm

[11] https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf#page=22

[12] https://techhub.hpe.com/eginfolib/networking/docs/switches/5950/5200-4024_openflow_cg/content/499752685.htm

[13] https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf#page=23