POLITECNICO DI TORINO

Laboratory on SDN and P4

“Computer Network Design and Management” class

Academic year 2021/22

Andrea Bianco, Paolo Giaccone, Alessandro Cornacchia, Matteo Sacchetto

Version: December 15, 2021

Chapter 1

Laboratory #1

In this laboratory you will experiment with Mininet and programmable switches. The aim of the lab is to get familiar with emulated environments, in particular after this lab you should know:

- basic Mininet commands;
- iperf tool;
- runtime manipulation of switch tables.

1.1 Starting the lab

1. Follow the provided instructions to install the VM
2. Run the VM
3. When the boot process completes, you will be prompted for the credentials. Use ***studente*** both as username and password.

1.2 Overview of shell commands

Open a terminal by clicking on "Terminal Emulator" icon under the left-top menu.

As preliminary requirement, you must become familiar with the text editor *gedit* (left command bar) and with basic terminal commands, available when clicking on *Terminal* (left command bar). The most useful terminal commands are reported in the following list. The student is recommended to revise each of them.

1. `ls` to list folder contents. E.g. `ls -lrt`
2. `cd` to change directory. E.g. `cd lab/forwarding`
3. `gedit` to edit a file. E.g. `gedit tx1.cc`
4. `man` to get the manual of a command. E.g. `man ls`
5. `cat` to print the file content. E.g. `cat forwarding.p4`
6. `command1&` to run `command1` in the background without blocking the terminal. E.g. `iperf -c 10.0.0.1 &`

7. `ifconfig` to show the configuration of the network interfaces

Hint: Use the *tab key* to complete file names and commands.

1.3 Getting familiar with the environment

1. Once the machine has booted open a new terminal window. Inside the terminal move to the folder containing the laboratory files by typing `cd lab/forwarding`.
2. Check the files contained in the folder by typing `ls` in your terminal window.
3. We'll be using network performance tool called `iperf` in order to generate synthetic traffic in our network. Read the documentation of the tool by typing `man iperf`.
4. We'll be emulating a real network by using Mininet. Instead of transferring packets on real wire, Mininet creates virtual network interfaces and emulates the communication between them. The performance in terms of bitrate of such an emulated network highly depends on the performance of the CPU of the host machine. Run `sudo mn --test iperf`, if prompted for a password use `studente`. The command starts an instance of Mininet with a simple topology comprising two hosts and a switch in the middle of them. After that, Mininet performs an iperf test between the two hosts and outputs the average datarate between the two hosts. Check the result of your test. You should observe a result like:
   ```
   Results: ['x bits/sec', 'y bits/sec']
   ```
   Write down your result:

   What is the meaning of the two values? What do they depend on?

1.4 Running the test topology

1. We'll be using a custom topology which is already provided inside a custom configuration script. In the same terminal, still under `lab/forwarding` folder, run `make clean` followed by `make`. The first command cleans any file and process left from previous runs while the second command starts the test topology. If prompted for a password use the previously used password. The startup may take few seconds. You'll notice that the network has successfully started when you end up with the following line:
   ```
   mininet>
   ```
   This is the command line interface (CLI) of Mininet and we'll be using it in order to interact with the emulated network.
2. Inside Mininet CLI type `help`, this will bring up a message with all the available commands in Mininet alongside with some hints about their usage. In order to get detailed information about a particular Mininet command type `help [command]`.
3. The test topology contains a certain number of hosts and switches interconnected by 10Mbps links. Hosts are denoted as `hID` while switches as `sID`. Using `nodes` and

`links` commands draw the test topology:

4. Each switch port is associated with a port ID. These IDs are used internally by switches in order to decide the network interfaces used for packet forwarding. Use `ports` command to find the association between each network interface and the corresponding port ID. Note: each network node, alongside with network interfaces, contains also a loopback interface `lo` which is always associated to port ID 0.

   Modify the figure above by adding to each network interface the corresponding port ID.

5. Discover the IP addresses of all the hosts using `ifconfig` running at each host (e.g., `h1 ifconfig`).

6. Inside Mininet CLI, check the connectivity between `h1` and `h2` by invoking `h1 ping h2`. Press `Ctrl+C` in order to terminate the test. Was the test successful? Can you explain in details the output of this test?

7. Repeat the previous test by checking the connectivity between `h1` and the remaining hosts. Did it work? Why in your opinion some of the tests were unsuccessful?

1.5 Providing full connectivity

1. Open a new terminal window. Enter `lab/forwarding/tables` folder. Open `s1-commands` file by writing `gedit s1-commands`. Analyze the first four lines of this file. What do you think is the meaning of those lines? The format of the command to add the rules in the table is the following:

   ```
   table_add [table_name] [action_name] [matching_key] => action_params
   ```

   where `table_name` identifies the table, `action_name` with parameters `action_params` the action to apply when matching `matching_key`, based on the definitions provided in `forwarding.p4`.

2. Open the remaining files (`s1..4-commands`). Some table entries are missing. Fill them according to the example provided in file `s1-commands` and the topology discovered in 1.4.3 to ensure full network connectivity. Report the table entries here:

s1:

s2:

s3:

s4:

3. In order to commit the changes you did in switches' tables, execute the `reloadsw` command inside Mininet CLI. Test the connectivity of your network by performing `pingall` command. Note: If your configuration is wrong, the command will advance very slowly as it will wait for multiple timeouts before checking other hosts. In that case, abort the command and check again the configuration inside your table files.

1.6 iperf, rate limiting

1. From another terminal window, open the bandwidth monitoring tool by going inside the lab folder and executing `python plotThroughput.py`. The script plots the outgoing

throughput measured at hosts interfaces. It provides three modes: interactive mode with fixed time axis (i), sliding window mode (w), and static mode (s). Try to execute the sliding window mode by typing `w` and pressing enter.

2. From Mininet CLI try to perform an iperf test between h1 and h3 by invoking `h1 iperf -c 10.0.0.6`.

   What do you observe inside the monitoring script? What is the bandwidth reported by the tool? Report down your results.

3. Open two virtual terminals for h1 and h2 by typing `xterm h1` and `xterm h2` inside Mininet CLI. Now try to emulate an incast scenario in which hosts h1 and h2 send data to h3. To do so type `iperf -c 10.0.0.3 -t 20` inside virtual terminals of h1 and h2. What do you observe inside the monitoring script? Try to repeat the experiment several times.

4. Suppose we lease our hosts to external users. Host h1 belongs to the set of premium users while all other hosts belong to normal users who must have their outgoing bandwidth throttled at **2Mbps**. Open again `s1-commands` and look at the entries in the "Rate limiting" section. These entries provide the definition of a rate limiter.

   The first line defines the scope of the rate limiting: in this case all flows with a source IP of 10.0.0.2/32. The parameter of the first entry is the meter index that will be used for the rate limiting.

   The second line defines the rates of the meter of index 0. Rates are defined according to a standard Two Rate Three Color Marker (trTCM), described in RFC 2698 (Color-Blind Mode), based on four parameters: Peak Information Rate (PIR) and its associated Peak Burst Size (PBS) and a Committed Information Rate (CIR) and its associated Committed Burst Size (CBS). The definition of trTCM follows the format:

   ```
   CIR(MBytes/s):CBS(Bytes) PIR(MBytes/s):PBS(Bytes)
   ```

   Note the unit of measurements in MBytes/s and not in Mbit/s.

   Depending on the arrival pattern of the monitored flow, the meter will output a value: 0 if the traffic pattern does not violate `CIR:CBS`, 1 if it violates `CIR:CBS` but not `PIR:PBS`, 2 if it violates `PIR:PBS`.

   The third line defines the actions associated with the meter. In this particular case, while the value of the meter is 2, all packets will be dropped. What are the current rates of the meter? For which traffic characteristics will it trigger? Set the rates in such a way to throttle the bandwidth of all the hosts, but h1, to 2Mbps. Try to play around with the scope of the meter and with its rates and observe what happens inside the monitoring script.

1.7 Extra assignment

1. Disable your rate limiter and try to perform an iperf test with h_i simultaneously sending data to h_{i+3} for i=1,2,3 (suggestion: make the test sufficiently long by using a suitable

value for -t parameter). Depending on the way you defined your routing tables you'll notice unfairness in the throughput among the three flows.

2. Take a look at the last section of `s1-commands`. This sections defines a load balancer able to balance flows across parallel links present in the test topology. This load balancer is able to guarantee fairness among concurrent flows. However, some parameters of the entries are wrong. You must first understand what is the meaning of these parameters. In order to do so open `forwarding.p4` file. This file contains the P4 code of the switches running in the test topology. Try to find the tables and the actions corresponding to the load balancer section. Try to understand what is done internally to the switch and try to spot and correct the errors in the provided flow rules.
3. Repeat the experiment of point 1. If you were able to correct the errors you will observe same rate for all three flows with their aggregate being 20Mbps.