National University of Singapore
School of Computing
CS5229: Advanced Computer Networks
Semester I, 2021/2022

Homework 0
Setting Up and Introduction to Mininet

Release date: Friday, 13th August 2021
Due: Friday, 25th August 2021, 20:29

Required Files

- `CS5229_VM_Image.ova`
- `Topology.py`

Deliverables

(to be submitted on Coursemology)

- `MyTopology.py`

The goal of this homework is to familiarise you with Mininet, a popular network emulator, and to recall some basic concepts from your undergraduate networking class.

Part 1: Setting up the VM environment

Before you begin, you will have to set up the necessary environment. Please follow the following instructions carefully.

Step 1: Installing VirtualBox

Download and install **VirtualBox** from `https://www.virtualbox.org` for easier installation.

Step 2: Installing the OVA

You will need to download the `CS5229_VM_Image.ova` file for this assignment. You can do so from here. After downloading, check the .ova file for correct size and MD5 checksum (size: 3,138,822,144 bytes, md5sum: `2e75a7abae28ff289f99ae1d6a004f30`). This virtual appliance contains a VM with all the necessary software (Floodlight, Mininet, CBench, and other network utilities) required to complete this and upcoming the assignments.

Once you have downloaded the .ova file, launch VirtualBox and go to `File` → `Import Appliance` and select the downloaded `CS5229_VM_image.ova` file. Keep the default settings for the VM configuration. Once you have imported the VM, boot it and login using the following credentials:

Username: `cirlab`
Password: `cirlab5229`

Part 2: Running a simple Mininet topology

Inside the VM, download the `Topology.py` file (from Coursemology) and place it in the `~/CS5229` folder. This file describes a very simple network with two hosts and two switches:

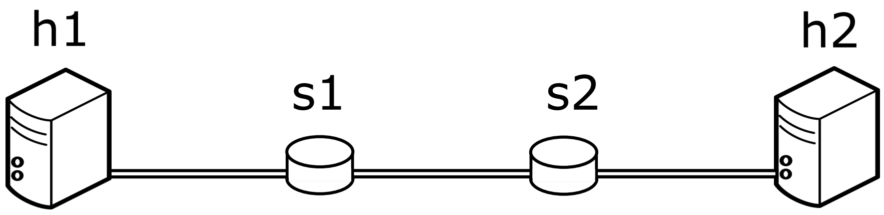

Figure 1: Simple Topology

The topology is defined inside the `LineTopo` class. The network components (hosts/switches) are first defined using the `addSwitch()` and `addHost()` functions, and then linked via wired links using the `addLink()` function. Each link can be modified to have different `bandwidth`, `delay`, etc. The topology shown in Figure 1 has already been defined for you with a certain bandwidth and delay for each link.

In this example, the forwarding rules on our switches are installed by a remote Floodlight controller. In this homework, we will only be working with Mininet, and so you do not have to bother about the network controller. However, to make your Mininet topology work, you will need to run Floodlight locally on port `5229`. To do so, open a new terminal and run the following commands:

```
cd ~/CS5229/floodlight-1.2
java -Dlogback.configurationFile=logback.xml -jar target/floodlight.jar
```

Next, we can go ahead and run our Mininet topology defined in `Topology.py`. Execute the following command in a new terminal:

```
sudo python ~/CS5229/Topology.py
```

After executing the above command you should see the Mininet console come up. Your network is now active. Verify that the switches are connected to the Floodlight controller correctly. You should see the following lines in the Floodlight console:

```
WARN [n.f.c.i.C.s.notification] Switch 00:00:00:00:00:00:00:02 connected.
WARN [n.f.c.i.C.s.notification] Switch 00:00:00:00:00:00:00:01 connected.
```

Here, `00:00:00:00:00:00:00:01` is the fixed datapath ID of switch S1 and `00:00:00:00:00:00:00:02` is the ID of switch S2. Congratulations on setting up your first Mininet topology! Now let's make sure everything is working as expected. Open a terminal inside one of your hosts:

```
mininet> xterm h1
```

This should open a new terminal console inside the host h1. You can check the IP address of this host by running `ifconfig`. For example, inside the “`Node:h1`” terminal:

```
root@cs5229:~CS5229# ifconfig
h1-eth0 Link encap: Ethernet HWaddr 00:00:00:00:00:01
inet addr:10.0.0.1 Bcast:10.255.255.255. Mask:255.0.0.0
...
```

Similarly, verify that the host `h2` has the IP address `10.0.0.2`.

Next, you need to check that both `h1` and `h2` are able to reach each other. You can do so by using the network utility `ping`. To ping `h2`, type `ping 10.0.0.2` into the terminal console of `h1`. In more complicated topologies, you can also type `pingall` into your Mininet console to check if all your hosts are able to reach each other.

Now, let’s transfer some data from `h1` to `h2`. Start an `iperf3` server on `h2` on port `3000`:

```
# Inside h2's terminal console
root@cs5229:~CS5229# iperf3 -s -p 3000
```

After this start an `iperf3` client on `h2` and note down the total throughput reported by both the server and client.

```
# Inside h1's terminal console
root@cs5229:~CS5229# iperf3 -c 10.0.0.2 -p 3000 -t 20
```

(**Hint:** Based on the simple topology and the `Topology.py` file, you should know what thorughput you can expect. Run this experiment multiple times to record the maximum achievable throughput between `h1` and `h2`.)

Questions

Please submit your answers to the following questions on Coursemology.

Q1: What is the average ping latency between `h1` and `h2`?

Q2: What is the average throughput between `h1` and `h2`?

Q3: You would’ve noticed that when you start your network and try pinging `h2` from `h1`, the first ping latency is considerably higher than the subsequent ping latencies. Can you explain this observation?

Part 3: Building your own Mininet topology

In this part, we will use what we’ve learnt from the simple example discussed in Part 2 to create a slightly more complicated topology. Your job is to implement the topology described in Figure 2 in a new file called `MyTopology.py`. The links in your topology must be parameterized based on the bandwidths and delays described in Figure 2. Each port should have a queue length of 1,000 packets and no stochastic packet loss. You should also enable `htb` (heirarchal token bucket) for all the links. For more help, please refer to the starter guide to Mininet.

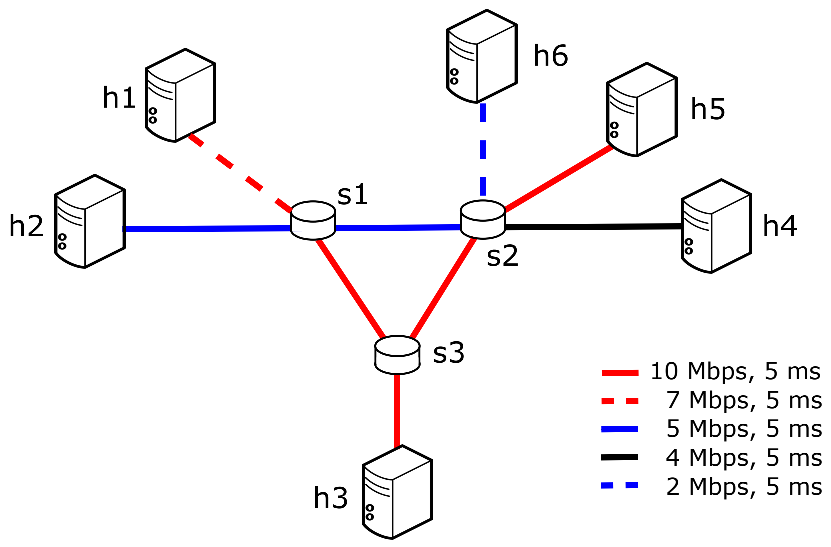

Figure 2: Custom Topology

Once you have implemented your topology and finished all the preliminary checks as done in Part 2, set up `iperf3` servers on `h4`, `h5`, and `h6`. Next, *simultaneously* launch three long-lived flows (at least 1 minute long) from `h1`, `h2`, and `h3` to the servers on `h5`, `h4`, and `h6` respectively. Note the average throughput for these flows.

Questions

Please submit your answers to the following questions on Coursemology.

Q4: Please submit your `MyTopology.py` file on Coursemology.

Q5: What is the average throughput achieved by TCP flows between `h1` and `h5`, `h2` and `h4`, `h3` and `h6`? Run this experiment multiple times to get a good estimate of the maximum achievable throughput. You should run all three flows simulataneously while recording the throughput.

Q6: What is the average ping latency between `h1` and `h5`, `h2` and `h4`, `h3` and `h6`?

Q7: Based on the throughputs and the latency between the hosts, you can probably guess the routes that the Floodlight controller has set for your topology. Can you think of any alternate routing scheme that can improve the per-flow throughputs in Q5? If there is such a routing scheme, how would the ping latencies in Q6 be affected by it?