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

Homework 1
Static Routing and setting SDN Policies

Required Files

- `CS5229_VM_Image.ova`
- `MyTopology.py`
- `Policy.py`
- `mininet_add_queue.py` OR `mininet_add_queue-NEW.py`

Deliverables

(to be submitted on Coursemology)

- Completed `Policy.py`

Background:

The goal of this homework is to learn how to set static routes and apply simple match action rules using the Floodlight controller. We will be setting these policies on the SDN-enabled switches we set up in Homework 0. Software-defined networking (SDN) technology is an approach to network management that enables dynamic, programmatically efficient network configuration that you will study about in more detail in Week 7. However, for the purposes on this assignment, you only need to understand the basic organization of an SDN network.

An SDN network (Figure 1) is organized into the *Data Plane* and the *Control Plane*. The Data Plane consists of the switching fabric, i.e. the switches themselves. In traditional SDN, the Control plane represents the programmable layer of the network[1]. The Control Plane itself also has two layers, the Network Control Plane (which is where our Floodlight controller runs) and the Application Control Plane. Controllers like Floodlight allow us to run applications in the Application Control Plane which can push rules and policies onto the switches in the Data Plane. This is where we be will writing and executing our program for this assignment.

More on SDN: `https://www.opennetworking.org/sdn-resources/sdn-definition`

[1]But as we will see in Week 7, today the Data Plane has become programmable too!

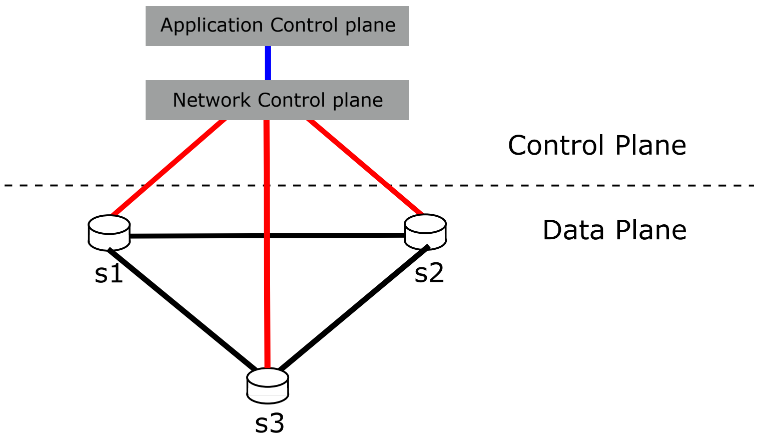

Figure 1: An SDN network.

Part 1: Setting up static routes

Log into the VM and download the `MyTopology.py`, `Policy.py`, and `mininet_add_queue.py` files from Coursemology and place them in ∼`/CS5229/`. `MyTopology.py` implements the custom topology you were asked to implement at the end of Homework 0 (Figure 2). We also discussed how the default routes set by Floodlight are sub-optimal because `h1` and `h2` share the link between `s1` and `s2`. If we want the network to perform optimally, we should route the traffic between `h1` and `h5` via `s3`. If we make this modification and let all the other flows take their default shortest routes, we should see an improvement in the throughput.

We will be making this modification via the `Policy.py` file. The `Policy.py` runs as an application over the Floodlight controller, which is responsible for pushing the rules set by you in the `Policy.py` to the respective switches. To give you an example of how these rules are installed, we have already added the rules for static routing between the `h3` and `h6` via `s3` and `s2`.

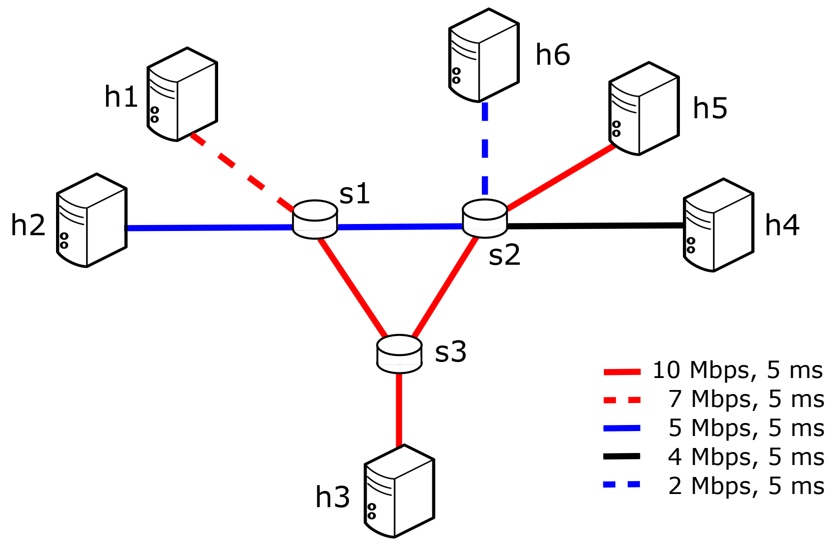

Figure 2: Custom Topology from Homework 0.

First, let's make sure everything is working as expected. Start the Floodlight controller and Mininet as we did in the last homework. In separate terminals, execute the following commands:

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

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

Next, place the `Policy.py` file in `~/CS5229` folder and run that as well:

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

Test your set up to ensure that `h3` is reachable from `h6`. If it is, you have successfully applied the static routing rules. Now add additional static routing rules to `Policy.py` inside `staticForwarding()` to set up the optimal static routes for the other two pairs of hosts as well, as described in Table 1:

Table 1: Static routes

	Source	Destination	Route
Flow 1	h1	h5	h1 → s1 → s3 → s2 → h5
Flow 2	h2	h4	h2 → s1 → s2 → h4
Flow 3	h3	h6	h3 → s3 → s2 → h6

Part 2: Setting simple match action rules.

In addition to defining static routes for our hosts, we will also apply some simple policies on our switches. Generally, a policy in a switch is applied via a `<match/action>` rule. As the name suggests, these rules filter packets based on some 'match' condition (example: `dest_port=22`) and then apply some 'action' to them (for example, dropping the packet). The `Policy.py` file already contains the basic things you need to write your policies. Your policies must go in the `S1toS2()`, `S2toS3()`, and `S1toS3()` functions.

For this assignment, you need to apply the following rules:

1. **Rule 1:** Restrict all traffic between `h3` and `h6` to 1 Mbps.
2. **Rule 2:** Block all traffic using destination UDP ports from 1000-1100 (ends inclusive) between `h2 and h4`
3. **Rule 3:** Allow only 100 MB to transfer between `h1` and `h5`. In other words, do nothing for the first 100 MB of data transferred between `h1` and `h5`, and then drop all packets once 100 MB has been transfered. You can assume a packet size of 1500 bytes.

Typically, a policy string should contain the following entries:

- Switch DPID
- Unique Policy Name
- Optional Cookie id
- Priority

- Matching Header components like eth_type/ipv4_src/ip_proto, etc
- Actions

Resources

Please refer to the following resources to understand the components and pre-requisite matches of a policy as well as other APIs of Floodlight:

1. `https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Static+Entry+Pusher+API`
2. `https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/21856267/How+to+Collect+Switch+Statistics+and+Compute+Bandwidth+Utilization`
3. `https://courses.cs.duke.edu/fall14/compsci590.4/notes/slides_floodlight_updated.pdf`

Answer the following questions on Coursemology:

1. Please submit your completed `Policy.py` file on Coursemology.
2. Once you have implemented your policy and finished all the preliminary checks, 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. 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 simultaneously while recording the throughput. Are these numbers an improvement over the throughput measured by you in Homework 0?
3. What is the average ping latency between `h1` and `h5`, `h2` and `h4`, `h3` and `h6`?

Hints

1. Make sure you add in your policies with a higher priority than the default static forwarding rules.
2. Make sure you read the `Topology.py` and `mininet_add_queue.py` files too before you start implementing your policies in `Policy.py`.
3. To implement some of the policies, you may need to use the queues defined by MyTopology.py. We have already defined these queues for you by making MyTopology.py call `mininet_add_queues.py`. Students with M1 Mac or Ubuntu 20.04 native setup should modify MyTopology.py to use `mininet_add_queues-NEW.py` instead.
4. For some policies, you may need to fetch statistics (e.g. packet counts) from the network before you can enforce a policy.
5. Once you write your functions in "Policy.py", you can run it first to apply the policies. Then to verify if the flows are added you can check them by executing the below command, which lists the flows in the switches:

   ```
   curl http://localhost:8080/wm/core/switch/all/flow/json|python -m json.tool
   ```

   Alternatively, you can also open "`http://localhost:8080/ui/index.html`" in the browser to view all the controller rules.