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

Homework 2
Dynamic Routing

Required Files

- `CS5229_VM_Image.ova`
- `com3-net.py`
- `Policy_template.py`
- `run_flows.zip`

Deliverables

(to be submitted on Coursemology)

- `Automonitor.py`
- `Policy1.py`, `Policy2.py`, and `Policy3.py`

In this assignment, we will explore how Dynamic Routing can have a positive effect of network throughput. You already have an idea of this from the alternate routing proposed in Homework 1 (Part 1), where re-routing a flow via a non-congested path improved network throughput.

Part 1: Set up

NUS's new computing building, COM3, is expected to finish construction by the end of this year. Because of your recent experience with setting routes and policies using SDN, NUS has hired you as a traffic engineer for the new COM3 network. While the real COM3 network has hundreds of hosts, we will reduce it to a network with 5 hosts for the sake of this assignment.

Log into the VM and download the `com3-net.py`, `Policy_template.py`, and `run_flows.zip` from Coursemology and place them in `~/CS5229/`. `com3-net.py` implements the simplified COM3 topology that we will be working with in this assignment. The entire network has three switches organized in a triangle topology and a total of 5 hosts (Figure 1). Each of the hosts represent a real entity in the COM3 network (Table 1)

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
```

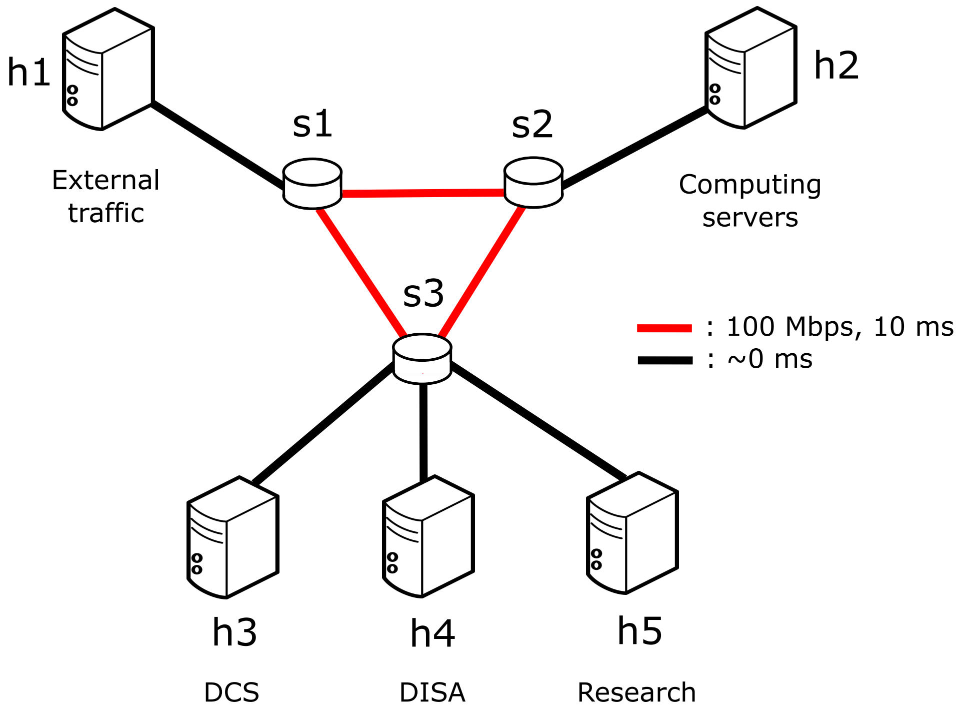

Figure 1: The new COM3 network.

Table 1: Hosts and their real network counterparts in the in `com3-net.py`.

Hostname	Corresponding real entity
H1	External servers (Internet)
H2	Internal servers (NUSnet)
H3	Clients from the Dept. of Computer Science
H4	Clients from the Dept. of Information Systems and Analytics
H5	Clients from a dedicated research network

```
sudo python ~/CS5229/com3-net.py
```

Now, apply `Policy_template.py` to the topology. This file does nothing special right now, just pushes the default shortest paths to the switches. It is provided to you as a starting point to implement the other policies in this assignment.

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

Now that your topology is up and running, verify that the all the hosts are reachable via the `pingall` command and make sure the latencies are close to what should be expected given the topology. You should note that by default, Floodlight will do shortest path routing (as we have seen in the previous assignments).

Part 2: Setting Dynamic Routes.

The goal of this assignment is to *Dynamically* set routes that can improve the network throughput. For example, let's say both `h3` and `h4` are downloading some file from `h1`.

If both these downloads are happening simultaneously, its clear how re-routing one of these flows via `s2` will improve the network's throughput. However, if these downloads are not happening at the same time, it's better instead for the flows to go via the shortest path (`s1`, `s3`).

This is a complex problem. Therefore, to make your life easier, you can operate under the following assumptions:

1. We will be testing how well your policies performs via **three test cases** (henceforth called *scenarios*) that will be similar to the two flow example discussed earlier. You need to write a policy for each of these scenarios. You can infer these scenarios from the files provided to you in `run_flows.zip`.
2. In all scenarios, there will be a maximum of 3 flows (with upto two of them being constant rate UDP flows and a maximum of one TCP flow). UDP flows are configured to transmit at a constant bit-rate. The TCP flow will perform rate adaptation and congestion control.
3. The flows will *always* originate from either H1 (external servers) or H2 (internal servers). H3, H4, and H5 will **not** send out traffic to anyone, including each other. They will, however, generate ACKs just like any normal receiver.
4. In each scenario, there can be at most one flow to a destination host (H3, H4, H5). A flow with H3 as its destination will **always be a TCP flow**. All flows to H4 and H5 will be constant rate UDP flows.
5. The default routing is shortest-path/hop.

Questions

Please submit the following files on Coursemology:

1. `Automonitor.py`: You should implement a monitoring script (similar to Homework 1) that can monitor and export the following metrics:
   (a) Link utilization (LU) and packet drop rate (Drops) at links S1-S3 and S2-S3.
   (b) Throughput received by H3 by monitoring the throughput of the link between S3 and H3.
2. `Policy1.py`: Dynamic routing policy for Scenario 1.
3. `Policy2.py`: Dynamic routing policy for Scenario 2.
4. `Policy3.py`: Dynamic routing policy for Scenario 3.

Testing your Policies

You can utilize the scripts provided to you in `run_flows.zip` to run through each scenario. Each scenario folder has a total of 5 files, each corresponding to each of the five hosts. These scripts will set up the iperf servers/clients necessary to run each scenario. You should note that since H3, H4, and H5 are always the receivers, you launch the servers on these hosts before you run the clients on H1 and H2. In other words, when testing the scenario, run `Run_H3.sh`, `Run_H4.sh`, and `Run_H5.sh` **before** running `Run_H1.sh` and `Run_H2.sh`.

You can also test your Policies *outside* of the scenarios given to you. You can do so by setting up iperf servers and clients as you wish on the Topology. To set up an iperf server on port 3000, run the following command on whichever host you wish to set your iperf server on:

```
iperf -s -p 3000
```

To send TCP traffic to this server for 60 seconds, run the following command on the desired host:

```
iperf -c <server_ip> -p 3000 -t 60
```

To send UDP traffic to this server instead, add the `-u` flag. To rate limit this UDP flow, use the `-b` as follows:

```
iperf -c <server_ip> -p 3000 -t 60 -b 80M -u
```

This launches a UDP flow that is rate limited to 80 Mbps.

Hints

1. **Study the scenarios before writing your policies.** Since your policies have to be scenario specific, study the scenarios provided to you carefully. For example, Scenario 1 can be sketched out as follows:

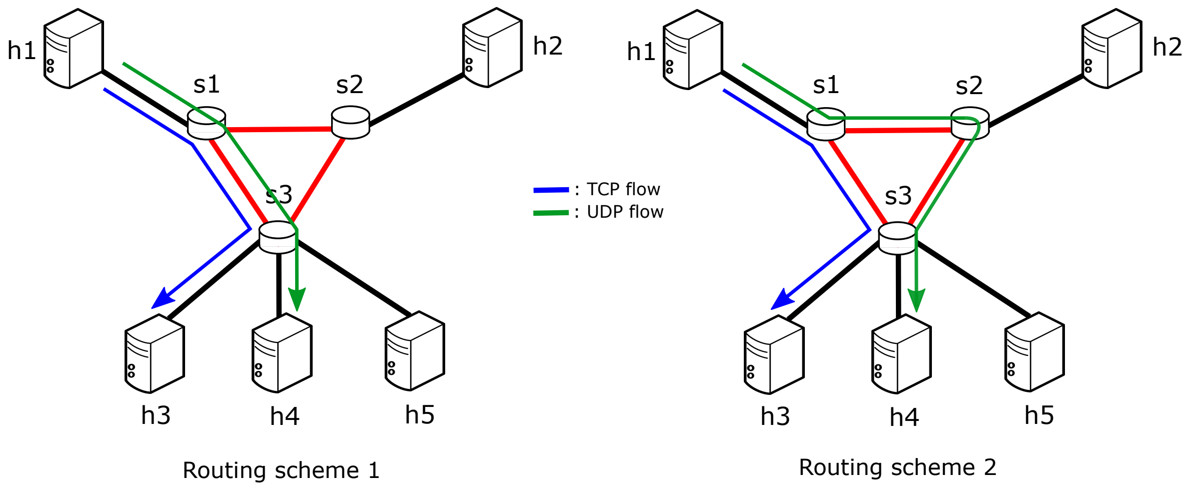

Figure 2: Scenario 1.

Figure 2 sketches out 2 possible routing schemes for the two flows (there can be others). You can then decide between these routing schemes in your `Policy.py` file.

2. **Setting static routing rules is not the right answer.** You might be tempted to think that setting static routes for *Routing scheme 2* in Figure 2 is the way to go for `Policy1.py`, however this is not the best answer. Since the TCP flow performs rate adaptation, there is no need to switch to *Routing scheme 2* till the link between S1 and S3 is completely utlilized. In fact, while the link between S1 and S3 is not utilized, *Routing scheme 1* is the better answer since it offers better latency for the UDP flow.

3. **Utilize `Automonitor.py` stats.** As #2 suggests, your routing rules need to be *dynamic* and depend on the link utilization. To figure out when is a good time to switch

between routing schemes, utilize the stats collected by `Automonitoring.py`. (Packet loss if your friend here)

4. It's a good idea to re-run your mininet topology and floodlight controller everytime you test a new policy so that the rules don't conflict.
5. Make sure you add in your policies with a higher priority than the default static forwarding rules.
6. As usual, you can check if your policies have been applied via the floodlight dashboard.

```
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.