Assignment #6

NES 580, Fall 2022, Dr. Ahmad T. Al-Hammouri

Due date: Saturday 14/1/2023 at 11:55pm.

Objectives:

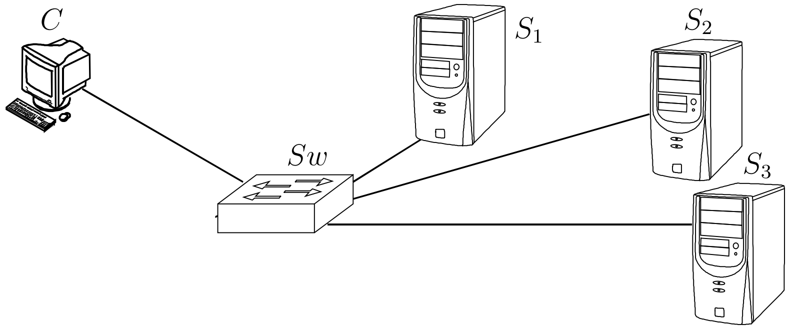
To write a P4 program that implements a simple representative load balancer.

Problem Statement:

In this assignment, you will be developing a P4 program that implements a simple representative **load balancer**.

The program requirements are as follows:

1. The P4 program must be written for the bmv2 software switch.
2. The network topology under consideration is shown below



3. The ARP table of each host is populated with the necessary information about **ALL** other hosts.
4. Client, C, only knows the identity of the front-end server, S_1, and it sends requests only to S_1.
5. All traffic sent from C directly to S_2 or S_3 should not be allowed.
6. All traffic from C to S_1 is disallowed **except <u>UDP</u> traffic destined to port <u>80</u>**.
7. When UDP requests to port 80 from C arrive at Sw, Sw redirects them either to S_2 or S_3 as follows:
   - If the request's UDP **source** port number is X, the request is forwarded to S_2.
   - If the request's UDP **source** port number is Y, the request is forwarded to S_3.
   - The values of X and Y will be determined by the control plane at run-time.
   - Other traffic is disallowed.
8. Responses are sent back to the appropriate requester.
9. Because C should be unaware of both S_2 and S_3, responses should appear as if they are coming from S_1.

Hints:

- The P4 program performs/implements the following steps:
  - Defines the Header types for Ethernet, IPv4, and UDP.

  - Defines Parsers for Ethernet, IPv4 and UDP that populate the Header fields.
  - Inside the Ingress control block,
    * Defines an action to drop a packet, using '`mark_to_drop()`'. Look into `basic.p4` exercise for an example (or, the actual code).
    * Defines an action that
      · Sets the egress port for an output port.
      · Rewrites the Ethernet destination or source address.
      · Rewrites the IP destination or source address.
      · **Rewrites the UDP check sum to zero**. (This is the simplest hack to prevent the Linux kernel validating the UDP checksum and discarding packets with incorrect checksum.)
    * Defines a table that looks at the UDP source port number and decides the proper action to be invoked. (Maybe, other data is also needed to decide the action.)
  - Updates the IPv4 checksum. Again, look into `basic.p4` exercise for an example (or, the actual code).
  - Finally, constructs the proper header fields with the proper order, and appends them to the outgoing packet.

- To run a simple echo server on both h_3 and h_4 use `socat` as follows:

```
socat -v PIPE udp4-listen:80,reuseaddr,fork
```

Grading Policy:

- You must turn in **only working code**. If your code gives compile- or run-time errors, you will receive **zero** credit.
- Partial credit is given only for working code that does not implement all the requirements above.

Deliverables:

- Submit your homework as a **single compressed** file (`ID-xxxxxx.zip`) containing **ONLY**
  - The P4 source code file (`ID-xxxxxx.p4`).
  - The Makefile (`Makefile`).
  - The topology definition file (`topology.json`).
  - The switch's run-time configuration file (`s1-runtime.json`).
  - **A text file containing the rules to be added via the API CLI** (`ID-xxxxxx.txt`).

  where '`xxxxxx`' is your student ID.
- Upload the compressed file to the `elearning` via the provided link. Do **NOT** send it via e-mail or a message from within the `elearning` even before the deadline because it will be deleted tacitly.
- ONLY one student from each group must submit the file.