PADERBORN UNIVERSITY

Advanced Networked Systems SS24

Lab4: Your Network, Your Say

Maximum points:	7
Submission:	zipped source code on PANDA
Deadline:	19.06.2024 23:59
Contact:	lin.wang@upb.de

1 Introduction

In the lecture, we discussed how to make the network plane programmable through the RMT hardware model and the P4 programming language. The goal of this lab is to get your hands dirty with P4 programming and related tooling. We will start with a router implementation and then extend it to support traffic interception on a network.

Please run `git pull` in the labs codebase to retrieve the code template for this lab. You will see a new folder `lab4` and you are supposed to work in that directory for this lab. Before you start, you should first run command `bash ./install_p4_env.sh` in the course VM to prepare the P4 development environment. Note that this script only works for Ubuntu 20.04. If you are using the course VM with Vagrant, there should not be any problem when directly running the prepared script; otherwise, you might face multiple issues. You may consult this page for troubleshooting.

2 "Longest-Path First" Routing

Your first task in this lab is to implement an IPv4 router with the P4 language. You should work in the directory `lab4/routing`. Please do not change the directory structure; otherwise, your code may not compile correctly. We will focus on the topology shown in Figure 1 for this implementation.

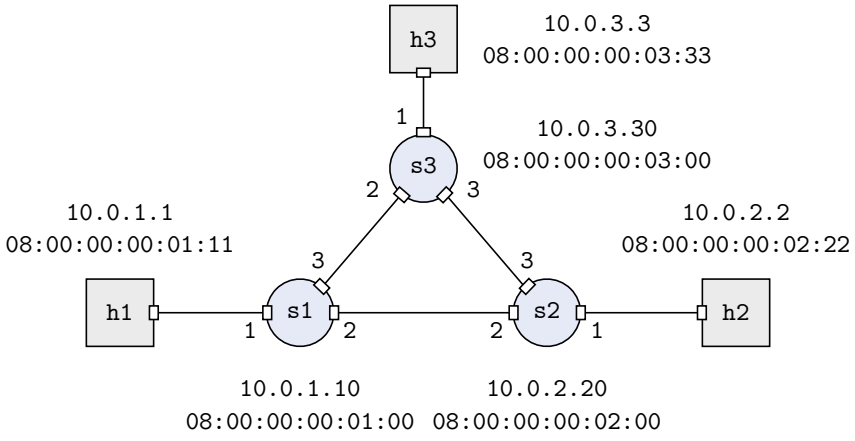

Figure 1: Network topology considered in this task.

Your router implementation consists of two parts: the data plane and the control plane. The data plane is implemented with P4 code. Specifically, you can define the packet header format, packet header parsing logic, and match-action tables in the `router.p4` file. Your router should fulfill the following requirements: (1) Packets are always routed via the longest path in the network, e.g., packets from `h1` to `h2` follow path `h1` → `s1` → `s3` → `s2` → `h2`. (2) Each router serves as the gateway for the host directly connected to it, which has already been configured in `topo/topology.json` file. This means that your router implementation should be able to

generate ARP responses for their attached hosts. (3) Routers should behave as normal IPv4 routers concerning the tasks involved in the packet forwarding process.

The control plane is normally implemented through the control plane interface exposed by the `P4Runtime`, through which we can dynamically add/remove table entries just like what we did for OpenFlow switches. Here, we simplify the setup by statically configuring the table entries in JSON files, which will be picked up by the control plane when setting up the network in Mininet. This process is already automated by the code template; you only need to modify files `topo/s*-runtime.json` to insert your table entries for the tables you have defined for router `s1/s2/s3` in the P4 code. You can assume all the IP addresses, MAC addresses, and port mappings are static and known when setting up the table entries.

Once you have completed your router implementation and inserted table entries, you can test if the routers are correctly implemented for IPv4 routing with the following command:

```
$ make
mininet> pingall
```

To verify that your router picks the "longest path" for packet forwarding, you should run `tcpdump` or `wireshark` to check how the packets are routed on the network when you run `ping` between a random pair of hosts. The log files `s*.log` under the `logs` directory contain detailed information about how packets are processed by your router, which might be helpful for your debugging. Remember to check it out.

3 Who Is Watching My Video?

Your second task in this lab is to intercept video traffic on a given ISP network. You should work in the `lab4/streaming` directory. Please do not change the directory structure to make sure that your code will compile correctly.

Imagine an ISP has a P4-enabled network with a topology depicted in Figure 2. There is a customer (`h7`) of the ISP who is watching a new TV series streamed from server `h1` over the network using the RTP protocol. Now assume you are also interested in this TV series but do not have access to it. As a network engineer, you found a magic way to take control of the whole network and intercept the RTP connection `h1` - `h7` to get access to the TV series.

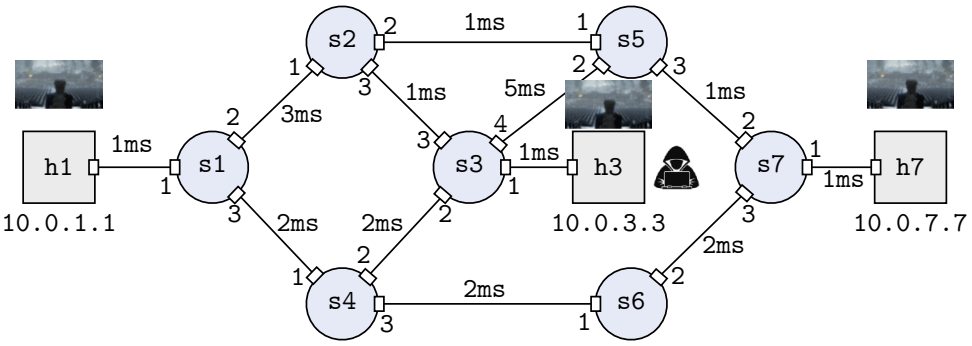

Figure 2: Network topology considered in this task.

First, you need to configure the network such that it works as it should. Let us assume the ISP was running a shortest-path routing protocol. The goal is to establish a connection between `h1` and `h7` with the lowest delay. Following the experience you gained from the previous tasks, please configure the routers in the network such that this goal can be achieved. (It is okay if you do manual calculations for the shortest path.)

Once you have set up forwarding correctly on all the routers, you can run `vlc-server.sh` on `h1` and `vlc-client.sh` on `h7` to see if the RTP streaming works as expected. You can use

```
mininet> xterm h1 h7
```

to open terminals for `h1` and `h7` and run the scripts from these terminals. These scripts are located under directory `lab4/streaming`.

Now comes the more interesting part: You want to intercept the traffic between `h1` and `h7` from `h3`. Think about what are the options and you are free to choose any method as long as you can justify it. The goal is to be able to receive a copy of all the packets correctly from `h1` to `h7` on `h3`. Once you have implemented your solution, you can test it by running

```
$ make
minint> h1 ping h7
```

Use `tcpdump` to listen on `h3-eth0` and check what you see from there. You can also test if you can receive and play the RTP stream on `h3` by running `vlc-client.sh` on `h3` and `vlc-server.sh` on `h1`. It might happen that `h3` is not able to play the video even though the packets have been forwarded to it. Think about why and find a way to handle it. If the video playing is sluggish, that is largely due to the lack of resources for the VM and you do not need to solve that problem as long as the video starts to play on `h3`.

4 Grading Criteria

The grading will be done based on an in-person interview-style oral examination. The detailed schedule for the interview will be announced when the submission deadline approaches. For fairness consideration, you must upload your code in a zip file. Please use the naming convention `Lab4_GroupX_LastName1_LastName2.zip` and rename the folder before you zip it. Here `X` is your group number and `LastName2` can be omitted if you are alone in the group. by the specified deadline and use the uploaded version for the interview. No interview will be scheduled for you if there is no code upload; this is a strict rule.

During the interview, you are supposed to show and explain the following:

- Your first router implementation must satisfy the requirements and the network is connected as a result of that. You should be able to explain the router implementation. (4 points)
- Your second router implementation allows video streaming from host `h1` to host `h7` and enables video interception on host `h3`. (3 points)