Advanced Computer Networks 2022
Vrije Universiteit Amsterdam
Lin Wang, George Karlos, Florian Gerlinghoff

Lab4: Who Is Watching My Video?

Type:	Group
Maximum points:	10
Submission:	Canvas upload (code and report)
Deadline:	Friday December 2, 2022
Contact:	vu.acn.ta@gmail.com

The goals of this lab are

- Implement an IPv4 router in P4
- Intercept RTP video streaming with P4
- Write a report to document your approach and results

Please pull the `lab4` directory from GitHub by running `git pull` in the `acn22-labs` directory. You are supposed to work in the `lab4` folder for this lab.

Note: Please use the template (i.e., `acn22-lab4-report.tex`) we provide for writing the report. There are specific instructions regarding what should be included in the report. For the submission, please put your report (in PDF) in the `lab4` folder and rename the `lab4` folder to `lab4-groupx` where `x` is your group number. (Please follow this naming convention **strictly** to make our life easier.) Then, please compress the folder into a `.zip` file. Please rename the folder before compression so that the folder name is correct when we decompress it. Please submit the zipped folder on Canvas to the assignment "Lab4".

1 Implementing an IPv4 Router in P4

In this part of the lab, you are supposed 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.

Your first task is to implement an IPv4 router with P4 in the code template `routing.p4`. As we learned in the lecture, you can define packet headers, specify packet header parsers, and define match-action tables with P4. Here, you need to properly define the packet headers including Ethernet (check Wikipedia for the Ethernet frame structure) and IPv4 (check RFC791 for details: https://tools.ietf.org/html/rfc791) headers, define parsers to properly parse the packet headers you have just defined, and define a match-action table that matches on the destination IP address (or prefix) and forwards the packet to the output port based on a forwarding table that will be supplied by the control plane. You may want to refresh what you have learned in Lecture 8 on a programmable data plane with P4.

For IPv4 routing, please pay attention to what operations you need to perform on a router. You can refer to the following document if you want to refresh your knowledge about IPv4 routing: https://people.cs.umass.edu/ arun/cs491g/lectures/IPForwarding-Lab3.pdf.

Once you finish your code you can compile it by running `make` in the lab4 directory. You will

see a Mininet prompt if the compilation is successful. If so, an instance of the network shown in Figure 1 is now built in Mininet. (The tiny numbers on the routers denote the port numbers.)

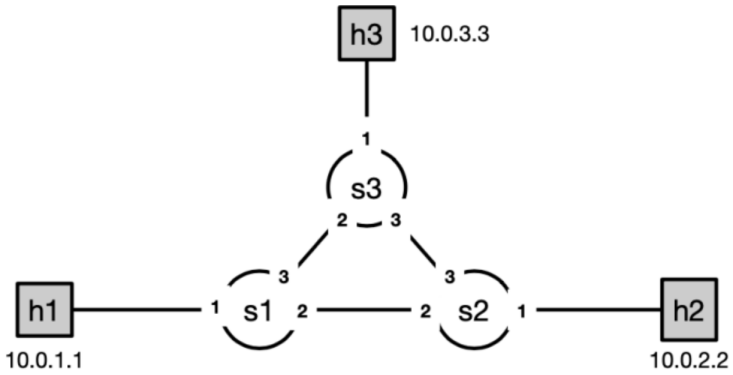

Figure 1: Topology used for routing with P4.

Your second task is to populate the table entries for the routers you have just built with a control plane. The routers (`s1`, `s2`, and `s3`) in this network work exactly in the way you have specified them with P4 in routing.p4. However, the routers have empty forwarding tables and you need to tell them how to forward packets by specifying table entries. This is similar to what you have done with Ryu where you install the flow entries by writing a control script for Ryu. However, here we take a simplified approach where we specify the table entries manually for the routers.

To specify table entries for the routers in the network, you can navigate to directory `lab4/routing/topo/` where you can find three files with name in the form `s*-runtime.json`, each corresponding to one of the routers matching their names. In these files, you can find a segment called `table_entries` and this is where you can specify the table entries that you want to install for your routers. Note that the format for the table entry is dictated by the way you define the match-action table in `routing.p4`. You may want to check the IP/MAC addresses that have been assigned to the hosts and you can find such information in the file `lab4/topo/topology.json`. The hosts have already been configured with default gateways and ARP tables, so you do not need to do anything to them.

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

```
$make
mininet> pingall
```

2 Intercepting RTP Video Streaming with P4

In this part of the lab, 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. Your first task is 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

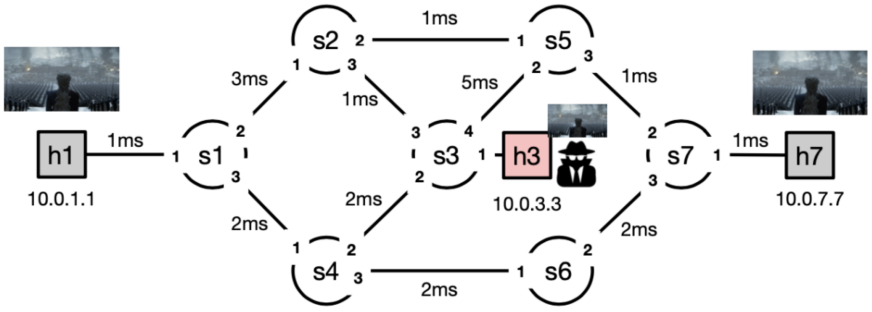

Figure 2: Topology used for video interception.

`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 to 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 explain it in detail in the report. The goal is to be able to receive a copy of all the packets 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`. You might notice 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. You do not need to solve the problem as long as the video plays on `h3`.

3 Assessment Criteria

This lab will be assessed based on the following criteria:

- **Correctly implementing the IPv4 router (4 points).** Your code should correctly implement the IPv4 router in P4 and the `pingall` test should succeed.
- **Correctly implementing the RTP streaming interception.** Your code should correctly implement the streaming from `h1` to `h7` and you should be able to play the streaming at `h7` **(3 points)**. Also, your code should allow you to play the streaming at `h3` **(3 points)**.
- **Code and report quality (1 point included for each task above).** Your report should document the necessary details for your implementation, including your approach, setups, and your observations, or other interesting findings/thoughts (if any).