CWM Programmable Networks

Exercise 5: Layer 2 Switch

Introduction

In this exercise you will write a P4 program that implements basic forwarding for IPv4. With IPv4 forwarding, the switch must perform the following actions for every packet:

1. Update the source and destination MAC addresses.
2. Decrement the time-to-live (TTL) in the IP header.
3. Forward the packet out the appropriate port.

Your switch will have a single table, which the control plane will populate with static rules. Each rule will map an IP address to the MAC address and output port for the next hop.

The repository contains a skeleton P4 program, l2switch.p4, which initially drops all packets. It also contains key pieces of logic replaced by TODO comments. Your assignment is to extend this skeleton program to properly forward IPv4 packets, replacing each TODO with logic implementing the missing piece. We have already defined the control plane rules, so you only need to implement the data plane logic of your P4 program.

The P4 program will be written for the v1model architecture implemented on P4.org's bmv2 software switch.

You are required to submit the p4 file with the code you have developed, and a link to your repository with the code. If you have written any additional code, you need to submit it as well.

Setup

The evaluation of this project requires 2 Raspberry Pis and one lab machine. Work in pairs to test your design(s).

The roles of the machines:

- Lab machine – client
- Raspberry Pi #1 – switch
- Raspberry Pi #2 – server

Implementation

Implement the switch based on the skeleton program l2switch.p4.

Replace all TODO comments with your code.

Make sure to implement parser, deparser, match-action control logic etc.

Make sure to implement the functionality:

1. Update the source and destination MAC addresses.
2. Decrement the TTL in the IP header.
3. Set the appropriate output port.

Testing

To configure the machines:

- Login to Raspberry Pi #2
- Setup static IP for the server in a different subnet:

```
ifconfig eth1 192.168.20.1 netmask 255.255.255.0
```

- Setup static ARP for the server (arp -s <ip> <mac>):

```
arp -s 192.168.20.1 xx:xx:xx:xx:xx:xx
```

- Login to Raspberry Pi #1
- Setup static IP on the switch eth1/eth2 as gateways & configure the static routes.

```
ifconfig veth0 169.254.249.11 netmask 255.255.255.255

route add -host 192.168.10.1/32 dev eth1

route add -host 192.168.20.1/32 dev veth0
```

- Compile the original P4 program using p4c.

```
p4c --target bmv2 --arch v1model --std p4-16 l2switch.p4
```

- Run the JSON file with p4runtime thrift-port enabled.

```
simple_switch -i 0@eth1 -i 1@eth2 l2switch.json
```

- Open a new terminal window to the switch (Raspberry Pi #1)
- Run runtime CLI :

```
python3 ./usr/lib/python3/dist-
packages/runtime_CLI.py --thrift-port 9090
```

- Add two forwarding rules from the runtime terminal:

```
table_add MyIngress.ipv4_lpm MyIngress.ipv4_forward
192.168.20.1/32 => xx:xx:xx:xx:xx:xx 1

table_add MyIngress.ipv4_lpm MyIngress.ipv4_forward
192.168.10.1/32 => yy:yy:yy:yy:yy:yy 0
```

when compiling the original P4 program, any attempt to ping will fail, because the switch is programmed to drop all packets on arrival. If your code was implemented and loaded correctly, you should be able to ping between devices. Try also running an iperf test!