Rede Definidas por Software
TP1 - OpenFlow

Version: 1

Departamento de Informática

Universidade do Minho

March 2022

João Fernandes Pereira

Goal

The goal of this work is to introduce you to the OpenFlow protocol. To that end, this assignment focus in the development of Applications on top of SDN controllers.
This assignment is composed by two exercises. In the first one, the students must develop an application that implements a ″Layer-2 self-learning switch″ on the OVSwitch. In the second exercise, the students must, following the presented topology, develop an application that implements a ″Layer-3 switch″ on the OVSwitch. The routing approach can be static or dynamic.

Report & Submission

All the develop code must be submitted. The report should consist in a small pdf document.

- Introduction
- Exercise 1
  - Strategy
  - Dry-Run
  - Tests
- Exercise 2
  - Strategy
  - Dry-Run
  - Tests
- Conclusion

The submission is made via blackboard with a zip file containing all the files.
RDS_TP1_GroupX.zip

Required Software

- Mininet
- Ryu (python) or FloodLight (java)
- Wireshark

Exercise 1

The goal of this exercise is create an application which functions as an L2 MAC self-learning switch (Hint: the app should keep a record from which port (interface) a given MAC address communicates (in_port)). The figure 1 show the topology.

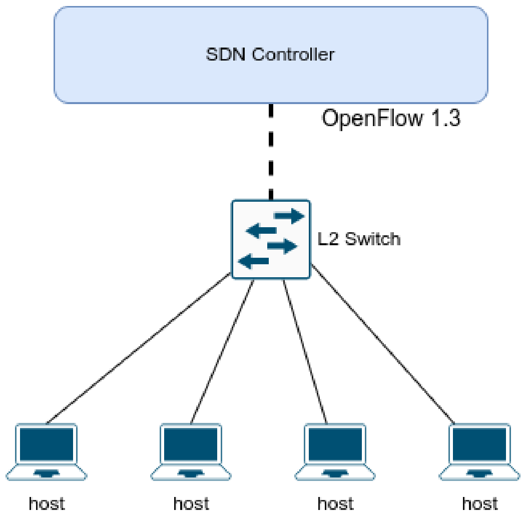

Figure 1: topology for exercise 1.

Tips

The first thing you have to do is learn the port for the source MAC, which essentially means to populate the dictionary with the input packet's source MAC Address and input port.
Then use that same dictionary and with the destination MAC address find out the packet's destination port. If is not null, you now have what you need to create a of_flow_mod object with a match field (destination MAC address) and a action field (sent to 'destination port'). Otherwise, send the packet to all ports ("Flood" behavior). Other of_flow_mod fields:

- idle_timeout — Idle time before discarding (seconds)
- hard_timeout — Max time before discarding (seconds)
- priority — Priority level of flow entry
- buffer_id — Buffered packet to apply to (or OFP_NO_BUFFER)

Exercise 2

In this exercise, you'll make a Layer-3 forwarder/switch. It's not exactly a router, because it won't decrements the IP TTL and recompute the checksum at each hop (so traceroute won't work). However, it will match on masked IP prefix ranges, just like a real router. This "router" can be completely static or dynamic. Figure 2 shows the topology that this exercise must follow. The idea is to use the Layer2 self-learning application done in previous exercise for the L2 Switches, and only develop a new application for the L3 Switch.
This exercise shows that with an OpenFlow-enabled forwarding device, the network is effectively layerless; you can mix switch, router, and higher-layer functionality.

Tips

If a packet is destined for some IP address for which the router knows the next hop, it should modify the layer-2 destination and forward the packet to the correct port.

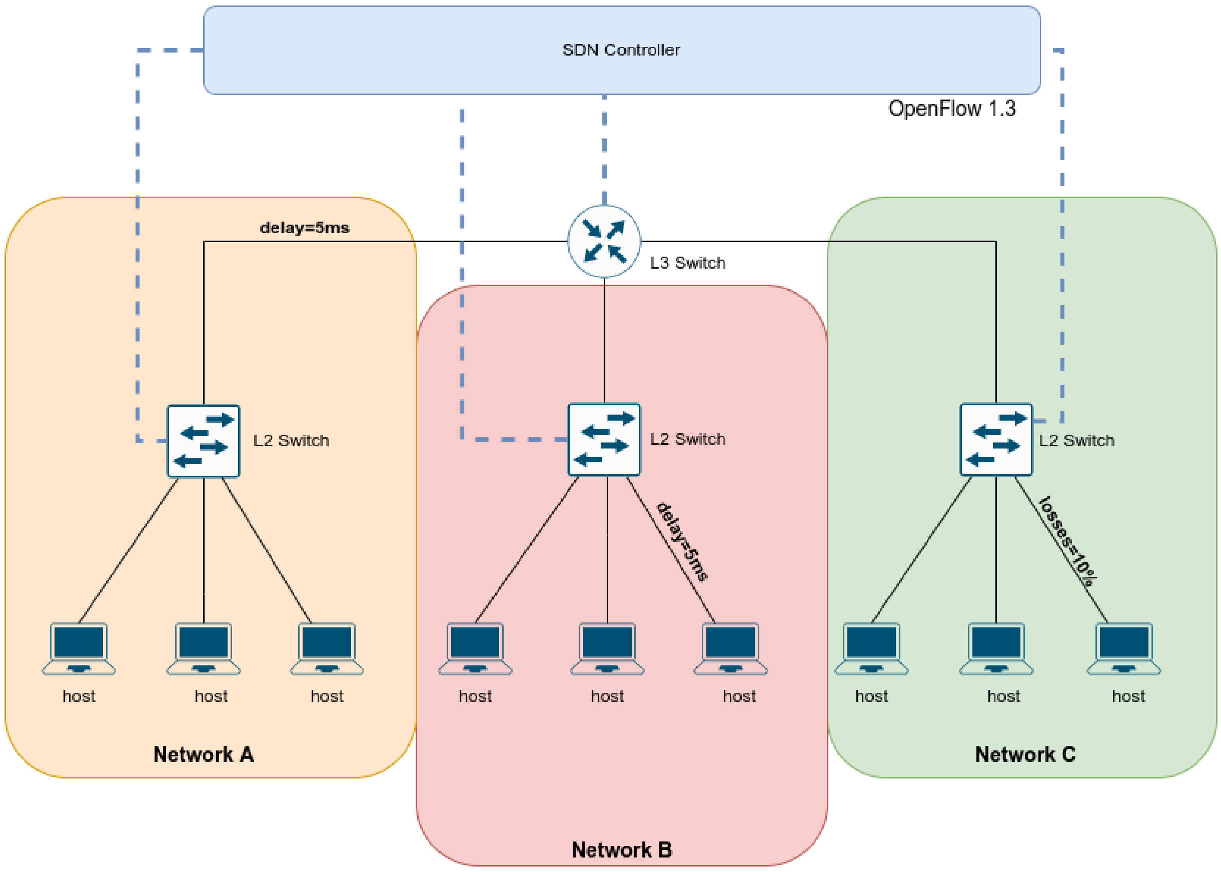

Figure 2: topology for exercise 2.

ARP

A router generally has to respond to ARP requests. You will see ethernet broadcasts which will (initially at least) be forwarded to the controller. Your application should construct ARP replies and forward them out the appropriate ports. Structures you will need:

* routing table (a structure with all of the information statically assigned)
* CAM table — content addressable memory table
* message queue (while the router waits for an ARP reply)

Static Routing

The router will need to handle routing for the static configuration. Since we know what is connected to each port, we can match on IP address (or prefix, in the case of the remote subnet) and forward out to the appropriate port.
We need to handle all ipv4 traffic that comes through the router by forwarding it to the correct subnet. The only change in the packet should be the source and destination MAC addresses.
If the router does not know the MAC address of the destination, it will have to arp for that host.

ICMP

Additionally, your app may receive ICMP echo (ping) requests for the router, which it should respond to. Lastly, packets for unreachable subnets should be answered with ICMP *network unreachable* messages.