Programming Assignment

Tan Wei Jie (A0202017B)

INTRODUCTION

In this programming assignment, I have utilised 2 Ubuntu 18.04 servers to deploy Mininet (149.28.154.110) and POX (45.32.104.61). The version for Mininet is 2.2.2 and for POX is 0.6.0 (fangtooth). By using real servers for deployment, I get to experience the actual deployment process as compared to using virtual machines.

TASK 1

For this task, I used file reading method to retrieve the `topology.in` input file instead of redirecting it to stdin. The input file must be in the same directory as the Mininet configuration file `mininetTopo.py`. The program will create the hosts and switches according to the first line input, and hosts and switches are numbered from 1 up until the expected number. Subsequently, it will create the links described in the `topology.in` file. Additionally, the speed for the links were stored in a dictionary $\{ (src, dest) \rightarrow speed \}$ so that it can be later retrieved when creating the QoS queue.

TASK 2

For this task, the POX controller is deployed in such a way that whenever it receives a request from the switch, it will cache the information into a dictionary $\{ dpid \rightarrow \{ \ src \rightarrow port \ \} \}$. When a destination is not found within the dictionary key, the pox will return a flood action to the switch. Otherwise, the POX controller will update the new switch rules to forward the packet to the port as stored in the dictionary cache. I have also referenced to pox/pox/forwarding/l2_learning.py as a guide for my implementation.

TASK 3

For this task, I have updated the time-to-live (TTL) for the rules sent by the controller in Task 2. The entry in the route table will be removed after the TTL is up. The setting for the entry is idle_timeout = 10 seconds and hard_timeout = 30 seconds. By adding a TTL value, the switch can be fault tolerant to outdated links and remain resilient in finding another route to the destination after a maximum of 30 seconds of lost connection.

While it is not required in the task to keep track of hosts changing interface / configurations, the cache in Task 2 should have a TTL so that whenever a host changes configuration, the cache can be updated without having the host initially send a packet to force update the cache.

TASK 4

For this task, I used file reading to retrieve the information stored in `policy.in`. The policy file must be in the same directory as the controller file. During the controller's initialisation, the program will read the input file and store the relevant firewall rules into a list.

Whenever a new switch is connected to the controller, the POX controller will update the switch's routing table entry. Thus, whenever a restricted packet is encountered, it will drop the packet immediately. This rule also has been assigned the highest priority level, so no other rules can be matched to circumvent this rule.

TASK 5

When reading the `policy.in` file in Task 4, the premium users' IP address will also be stored in a list. For this task, I have updated the rules in Task 2 such that whenever the receiving packet's source IP address matches with one entry in the premium user's list, the packet will be assigned to the premium queue (Queue ID: 1). Otherwise, all packets will be assigned to the general queue (Queue ID: 0).

For the Mininet side, for every interface on a switch, 2 QoS queues will be deployed for each interface. These 2 queues are the general (ID: 0) queue and premium (ID: 1) queue. If the switch is connected to a host, it will apply the appropriate bandwidth. If the switch is connected to another switch, both queues will get the link speed, which is stated in `topology.in` input file.

CONCLUSION

In this programming assignment, I have learnt how to emulate a software defined network. I can extend it to understand that installing and maintaining a large network using an OpenFlow controller is simpler than traditional networks. Furthermore, complex rules can be used to fine-tune the traffic, which might be harder to do in traditional networks. An update to the switches can be rolled out seamlessly onto the controller. For example, when the old routing rules expired, the switch can fetch new firewall rules and premium IP addresses.