CS4226 Programming Assignment Report

Name: Tan Kai Qun, Jeremy Matriculation Number: A0136134N

Task 1

In mininetTopo.py:

- A new *TreeTopo* subclass is created which is inherited from the *Topo* class.
- According to Mininet's documentation, the *build* method will be called when the *Topo* class is instantiated. As such, the *build* method can be overridden to build our topology.
- "topology.in" is read and parsed in the overridden *build* method and the corresponding hosts, switches and links are initialized accordingly
- Finally, a *Mininet* object is initialized with the created *TreeTopo* instance, and a connection is established with the controller.

In controller.py:

- A simple implementation is used where it will command the switches to flood any packets they received to all ports.

Task 2

In controller.py:

- An in-memory *ForwardTable* is created where it facilitates in keeping track of the port to forward a receiving packet (given a specific mac address) for each switch.
- If an entry does not exist in the *ForwardTable*, the controller will create a new entry using the packet's source mac address and input port of the switch. Thereafter, the controller will command the switch to flood the packet.
- However, if an entry exists, the entry is then converted into a rule to be inserted in the flow table and propagated to the switch in the *Mininet*. This will allow the switch to forward the packet to a specific port according to the newly inserted rule.

Task 3

In controller.py:

- A TTL value (30 seconds) is set for the entries in the in-memory *ForwardTable* as well as for the corresponding forwarding rules in the flow tables of the switches in the *Mininet*.
- Additionally, the created timestamp of the *ForwardTableEntry* is also being kept track of so that the difference between the current timestamp and the created timestamp can be computed and compared against the TTL value to decide if the entry has expired.

- In other words, the entries/rules will expire after a while and this allows the in-memory *ForwardTable* to learn new entries (when the topology has changed) and propagate the new entries as new rules to the flow tables of the switches in the *Mininet*.

Task 4

In controller.py:

- "policy.in" is read, parsed and stored in memory when the *Controller* is instantiated.
- When a switch connection is first detected (*_handle_ConnectionUp* is called), the stored firewall policies are converted into rules that will be propagated to the switch in the *Mininet*.
- The rules will be inserted into the flow table of the switch and they would make the switch drop TCP packets which match the firewall policies <src ip (optional), dst ip, dst port>.
- Additionally, to resolve any potential conflicts between firewall and forwarding rules, firewall rules are given FIREWALL_PRIORITY which has a higher value (priority) than FORWARD_PRIORITY which is given to forwarding rules.

Task 5

In controller.py:

- When an entry exists in the in-memory *ForwardTable*, the destination ip address of the packet is retrieved and checked against the stored premium traffic hosts which was previously read from "policy.in".
- If the destination ip address matches with any of the premium traffic hosts, then the forwarding rule will be created with instruction to dictate the switch to forward the packet onto the queue with queue_id PREMIUM_TRAFFIC. Else, the switch will forward the packet onto the queue with queue_id NORMAL_TRAFFIC.

In mininetTopo.py:

- From the switches in the *Mininet*, all the host-switch links are retrieved while switch-switch links are ignored (with the assumption that links between switches will never become the bottleneck).
- The link capacity (bw) for every host-switch link is also retrieved.
- 2 queues representing the normal and premium traffic with different bandwidths (some fractions of the link capacity) will be set up at every host-switch link interface. Normal traffic queue has queue_id NORMAL_TRAFFIC while premium traffic queue has queue_id PREMIUM_TRAFFIC.