CS4426 Project Summary Report

Hu Haojie (A0181682Y)

Implementation of traditional learning switch and firewall

There are two major methods that are implemented in traditional learning switch: forward method and firewall installation.

The forward method is the core that responsible for forwarding packets based on MAC address learning, reducing unnecessary network traffic that would result from flooding. The main idea is to record the source MAC address and associated port in a MAC table.

1. If the destination MAC is unknown, the packet is flooded to all ports.
2. If the destination MAC is known, the packet is forwarded to the corresponding port using flow rule: On a successful MAC address lookup, a temporary flow rule (60 seconds) is installed to directly forward subsequent packets to the destination, enhancing the efficiency of the network.

The firewall installation method implements a network-wide firewall by installing a permanent flow rule that drops packets matching specific criteria: source IP address, destination IP address, and destination TCP port.

1. It constructs a “TrafficSelector” to match Ethernet frames carrying IPv4 payloads from a source IP to a destination IP on a specified TCP port.
2. Then, “TrafficTreatment” builder to create an action that drops the packets.

Implementation of intent-based learning switch.

Two main parts are implemented in intent Forwarding: “process” method and “setUpConnectivity” method.

The “process” method checks if the destination host of an incoming packet is known. If so, it sets up connectivity (if needed) and forwards the packet to the destination. If the destination is unknown, it floods the packet to all neighbors.

The method “setUpConnectivity” is one of the core methods where intents are either retrieved, resubmitted, or submitted for the first time. It handles the insertion of flow rules based on the intents and deals with the “HostToHostIntent”.

1. If the intent is in a WITHDRAWN_STATES, it's resubmitted.
2. If the intent has failed, a temporary drop rule is inserted.
3. If there's no intent, a new HostToHostIntent is created and submitted.

Campare between “learning switch” and “intent-based forwarding”.

The "learning switch" and "intent-based forwarding" are two approaches to handling packet flows within a network.

A learning switch listens to packets on the network, learns the source addresses, and associates them with the switch port on which the packets are received. When it needs to forward a packet, it looks up its table of learned addresses to find the port associated with the destination MAC address. If it doesn't know the destination, it floods the packet out of all ports (except the one it came in on). The learning process is dynamic, and the switch builds its forwarding table on the fly, updating it as the network topology changes.

However, intent-based forwarding works in totally different way. In intent-based forwarding, a higher-level abstraction is used. Instead of reacting to packets individually, the system relies on a predefined policy or "intent" that describes how certain types of traffic should flow through the network. These intents are set up in advance and can include a variety of conditions and actions. For example, they can specify that traffic from a specific source to a specific destination should always take the most direct path or should be prioritized for certain services.

When facing varied topology, learning switches must adapt continually as hosts move and as the network topology changes. They are sensitive to broadcast storms because unknown destinations cause flooding. We can conclude that learning switches may suffer from convergence issues when the topology changes, taking time to learn the new paths.

On the contrary, Intent-based forwarding is less reactive to individual packet arrivals and instead focuses on maintaining the high-level policies represented by intents, which indicates that it is generally more robust to changes in the network topology, since the intent defines the desired outcome, and the system works out how to achieve that as the topology changes.

Difficulties and resolutions in ONOS programming

Issue Description:
During the course of my assignment, I encountered a persistent problem related to the implementation of flow rules within the ONOS framework. Specifically, the challenge emerged when conducting pingall tests to verify network connectivity. Contrary to expectations, the test packets were not being handled as anticipated for a long programming time.

Problem Identification:
My initial approach to troubleshooting involved utilizing online resources such as stackoverflow, ChatGPT, and Google. Despite extensive searches and attempting various suggested solutions, the issue persisted, eluding resolution. Subsequently, I sought insight through active discussions on Piazza with teaching assistant, which played a pivotal role in directing my focus towards the potential root of the problem.

Heeding the advice of a TA, I delved into the system logs using logtail, which was a turning point, as it highlighted the critical nature of the makeTemporary(60) parameter within the flow rule definition – an aspect I had previously overlooked. Upon integrating the makeTemporary(60) into the flow rules, a notable transformation occurred. The once elusive success in my “pingall” tests was now achieved, indicating that the flows were correctly installed and expired as intended, thereby facilitating the desired network behavior.