National University of Singapore

School of Computing

CS4226 **Programming Assignment** Semester 1 AY22/23

Submission Deadline

12nd November 2023 (Sunday) 11:59pm sharp. 2 marks penalty will be imposed on late submission (late submission refers to submission or re-submission after the deadline). The submission folder will be closed on **19th November 2023 (Sunday) 11:59pm** sharp and no late submission will be accepted afterwards.

Objectives

Welcome to the CS4226 Laboratory Project. In this project, you'll complete a set of exercises designed to help you get more familiar to network switching and routing, and learn basic programming with ONOS, a state-of-the-art distributed network operating system.

This assignment is worth **20 marks** in total and they are split across **three exercises**. All the work in this assignment shall be completed **individually**.

Plagiarism Warning

You are free to discuss this assignment with your friends. **However, you should not share your answers, screenshots or network logs.** We highly recommend that you attempt this assignment on your own and figure things out along the way as many resources are available online and the troubleshooting skills you learn will be very useful.

We employ zero-tolerance policy against plagiarism. If a suspicious case is found, student would be asked to explain his/her answers to the evaluator in person. Confirmed breach may result in zero mark for the assignment and further disciplinary action from the school.

Question & Answer

If you have any doubts on this assignment, please post your questions on Piazza forum or consult the teaching team. However, the teaching team will NOT debug issues for students. The intention of Q&A is to help clarify misconceptions or give you necessary directions.

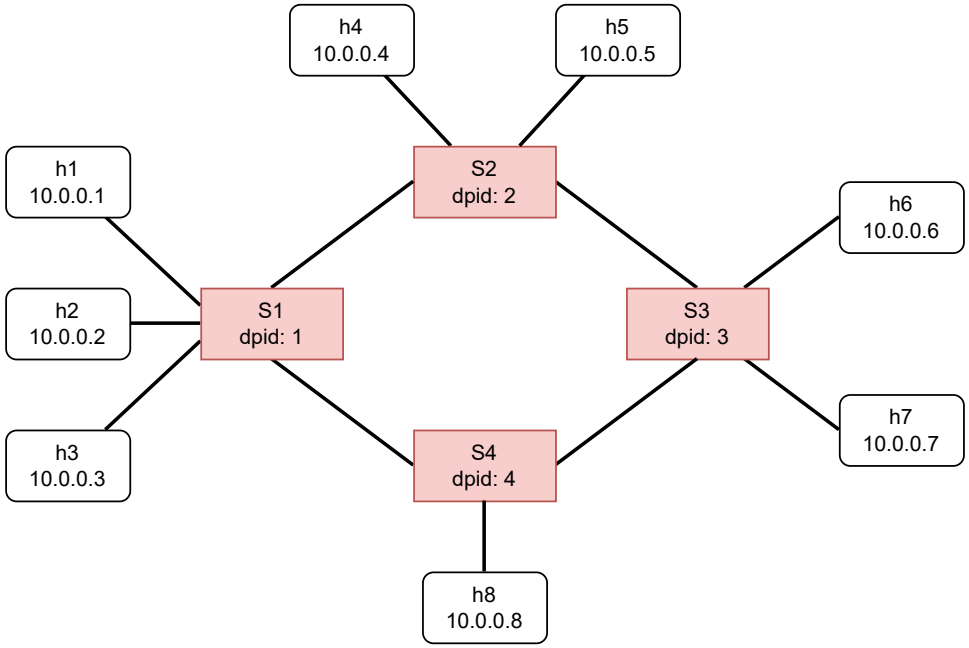


Figure 1: An example of network topology with 8 hosts, 4 switches and 12 links.

Submission

please submit a **zip** file that contains source programs and submit it to the `Programming Assignment (Source Code)` folder of Canvas Assignments. Please name your zip file as **<Student number>.zip**, where **<Student number>** refers to your matric number that starts with letter A. Note that the first and last letters of your student number should be capital letters. One example file name would be **A0112345X.zip**. Your zip file should contain the entire Java Maven project and a summary report. In addition, please submit a demo video that contains results of each test case to `Programming Assignment (Demo Video)` folder of Canvas Assignments. If you have further questions in submission, please send an email to your TA: maoyancan@u.nus.edu.

ONOS VM Image for Programming Assignment

You can download the ONOS + Mininet VM image for Programming Assignment here.

Exercise 1 - Building a Virtual Network (5 Marks)

In this section, your task is to build a virtual network in the Mininet network emulation environment. The network topology is given in an input file. Below is an example of a network topology, consisting of 8 hosts, 4 switches and 12 links that connect them together, as shown in Figure 1. In this network, the 8 hosts (with IP addresses from 10.0.0.1 to 10.0.0.8) are connected with 4 switches (with dpid from 1 to 4) by 8 links. The 4 switches are then connected together by 4 links.

Your program need to build a network with the topology and specifications described in the input file ".in". The first line of the file has 3 integers indicating that there are

N hosts, M switches and L links in the network. The following are L lines of tuples in the form of $< dev1, dev2 >$ that describe the links, meaning that $dev1$ and $dev2$ are connected via a bi-directional link at both directions. A partial input file for the network in Figure 2 looks like follows:

8 5 12

h1,s1

h2,s1

h3,s1

h4,s2

...

Exercise 2 - Traditional Learning Switch (8 Marks)

In this section, you will need to implement a traditional learning switch on ONOS. With your "learningSwitch" application installed in ONOS, any two random hosts should be able to ping each other.

Task 1: Learning Switch (5 Marks)

In order for the hosts to communicate with each other, the switches need to know how to correctly route a packet to its destination. In this part, your task is to implement the application of self-learning switches using the ONOS controller, which enables the switches to learn how to route packets without any prior knowledge of the network.

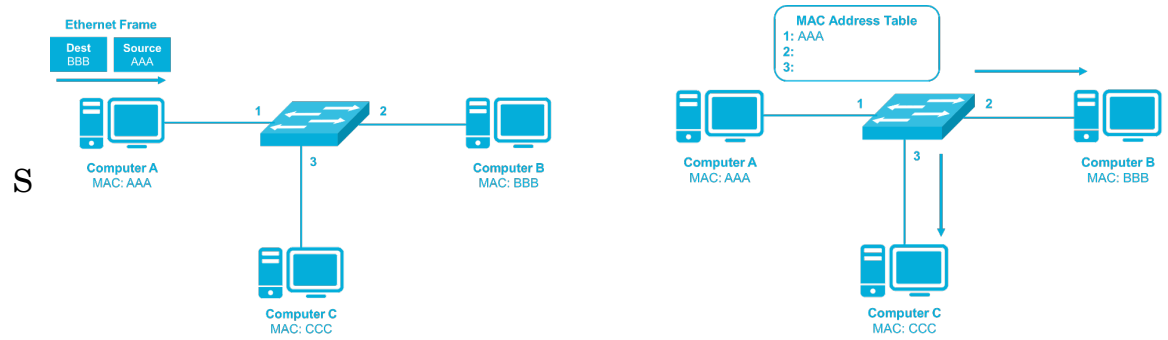
The key idea is to record the MAC address and the corresponding port number when receiving a packet that does not match any entry in the forwarding table. Thus, the switches can gradually learn the port number for reaching each host in the network. In the following part, we provide a simple example for illustration.

Initially, the switch has no knowledge of the three computers A, B and C that connect to it as is shown in Figure 2a. When Computer A sends a frame to Computer B, the switch does not know how to route the frame. It then floods this frame to all the ports, hoping that Computer B could receive it and reply.

Although now the switch still knows nothing about Computer B, it has learnt that Computer A can be reached by port 1. It then adds this entry into its forwarding table as is shown in Figure 2b, so that a frame destined for Computer A will be routed to port 1 in the future. Similarly, when the reply from Computer B comes back, the switch can subsequently learn that port 2 corresponds to Computer B.

Task 2: Firewall (3 Marks)

In this section, you will implement a layer-3 firewall application using the ONOS controller. The application needs to block traffic flows: the TCP traffic sent to a certain host on a certain port. You will be using the test case in the ONOS walkthrough document to



S

(a) Initial state.

(b) Learning MAC address of A.

Figure 2: Example of a self-learning switch.

verify it. In particular, the test case specifies a firewall rule to block the TCP traffic from h1 to h4 on port 4001.

The basic idea is that we need to implement a ONOS CLI called `add-firewall-on-port` to manually install the rules on the ONOS controller.

Directory tree

```
.
├── app # ONOS application folder
│   ├── learningSwitch  # Learning switch framework
│   │   ├── src/main/java/org/onosproject/learningSwitch
│   │   │   ├── cli
│   │   │   │   └── AddFirewallOnPortCommand.java
│   │   │   ├── package-info.java
│   │   │   ├── macTableEntry.java # Helper class for MAC table
│   │   │   └── LayerTwoManager.java # Main learning switch logic
│   │   └── pom.xml
│   └── pom.xml # Maven configuration file
└── topos # Mininet topology folder
    ├── star.py # Load star topology into mininet
    └── star.in # Input topology file of star structure
```

Todos

1. Implement star topology by filling “star.py”. (3 Marks)
2. Implement “LayerTwoManager” application in “learningSwitch/”. (5 Marks)
3. Implement “AddFirewallOnPortCommand” command line in “learningSwitch/”. (3 Marks)

Exercise 3 - Intent-based Learning Switch (7 Marks)

In this section, you will need to implement a simple reactive forwarding app that uses intent service. With your "intentForward" application installed in ONOS, any two random hosts can be connected via high-level, user-defined "Intent".

In this exercise, we are going to implement packet forwarding in ONOS way. ONOS, as a networking operating system, has two distinct advantage comparing to tradition network: high-level application intention and global network topology view. This exercise is designed to elaborate these two aspects.

Task 3: Fault tolerant learning switch (3 Marks)

You will need to implement Task 1 using intent-based abstraction in ONOS.You will need to evaluate the fault-tolerance functionality for it. In reality, links could be down due to hardware or software failures. In the Mininet environment, we will emulate such dynamic scenarios during runtime by bringing down/up links (see for more information). The learning switch should be able to tolerant this kind of situations.

The fault-tolerance functionality of learning switch can be hard to implement in other SDN controllers such as POX. However, it can be natively supported in ONOS controller. In POX, the controller need to attach a time-to-live or TTL value to entries in the routing tables. Any outdated entry, i.e., an entry for which the duration between the current time and the creation time exceeds the TTL value, will be removed from the routing tables. After that, the switch can broadcast to construct a new routing table entry even if the network topology has changed. In contrast, ONOS intent-based controller can actively handle the link down/up events, and will update forwarding tables automatically.

Report and Demo Video (4 Marks)

You are expected to write a 2 pages report to summarize your programming experiences on implementing SDN controllers on ONOS. In addition, you need to submit a demo video that showcases your output for the test cases.

The report name should be <Student Number>.pdf. The main content should cover:

- Compare "learningSwitch" and "intentForward", state in your opinion, how are they different from each other? Do they behavior differently with varied topology?
- Difficulties faced during learning/programming on ONOS and how do you solved them?

The demo video length should be within 10 minutes and cover all test cases listed in the ONOS walkthrough document. The video name should be <Student Number>.mp4, where the video type needs to be .mp4.

Directory tree

```
.
├── app # ONOS application folder
```

```
    ├── intentForward  # Intented fowarding framework
    │   ├── src/main/java/org/onosproject/intentForward
    │   │   ├── package-info.java
    │   │   └── IntentReactiveForwarding.java # Main intented forwarding logic
    │   └── pom.xml
    └── pom.xml # Maven configuration file
└── topos # Mininet topology folder
    ├── ring.py # Load ring topology into mininet
    └── ring.in # Input topology file of ring structure
```

Todos

1. Implement ring topology by filling “ring.py”. (Similar to star.py 2 Marks)
2. Implement “IntentReactiveForwarding” application in “intentForward/”. (3 Marks)
3. Complete a 2 pages report. (2 Marks)
3. Demo video to show all test cases. (2 Marks)

THE END