CMPUT 379 Assignment 2 Report

By Nathan Klapstein #1449872

Table of Contents

Contents

Objectives

a2sdn

The main objective of a2sdn was to create a linear software-defined network (SDN) that composed of one main controller and multiple switches (sw1, sw2, … sw7) that communicated between each other via named pipes (FIFOs) and a common traffic file.

Other objectives required by a2sdn are:

Controller

The controller for a2sdn is required to communicate between each connected switch via named pipes (FIFOs) and handling the creation and management of routing rules to be applied to activate switches. The controller is required to keep tabs on each connected switch such as noting each switch's accepted source and destination IPs and their neighboring switches.

Other objectives required by the controller are:

Switch

Each switch for a2sdn is required to communicate with the controller with a named pipe (FIFO) and is required to store, understand, and apply routing logic provided by the controller.

Other objectives required by the switch(es) are:

For a more detailed report on the objectives required by a2sdn please read:

- https://eclass.srv.ualberta.ca/mod/url/view.php?id=3160961

Design Overview

a2sdn

- C++ was used over C due to its added functionality
- Class-based structure was used to separate functionality
- Used a cmake file over a makefile for development (more modular/futureproof)
- Connection class was created to represent two connection FIFOs (one sending and one receiving) between a switch-switch or switch-controller
    - This kept things simple and helped construct an API to interact easily between switches and controllers.
- Packet class was created to keep transfers between switches and the controller uniform
- poll() was used due to its simplicity of handling multiple and dynamic file descriptors
- Majority of print statements have starting strings of ERROR: WARNING: INFO: and DEBUG: for ease of debugging
- Signals were handled using a signalfd (see signalfd.h) thus, allowing for poll to query them
    - This kept implementation uniform between polling stdin, the connection FIFOs, and signals

Controller

- Controller class was created to represent a controller
  - handled construction and argument parsing
  - start() method started the controller's poll() loop
  - Class functionality for responding to OPEN, QUERY Packets
  - Class functionality for sending ACK, and ADD Packets
  - list() provides statistics on all received Packet types for debugging usage

Switch

- Switch class was created to represent a switch
  - Handled construction and argument parsing
  - start() method started the switch's poll() loop
  - ==, <, and > operators were defined for deduping and sorting the list of switches used by the controller
  - Class functionality for responding to ACK, ADD, and RELAY Packets
  - Class functionality for sending OPEN, QUERY, and RELAY Packets
  - list() provides statistics on all received Packet types for debugging usage along with packets delivered to the switch.
  - The switch will error and exit if a controller is not active (this seemed acceptable as we don't want any dangling switches running for no reason)
- FlowEntry structure was used for modeling flow rules for the switches
- FlowTable (a vector of FlowEntrys) was used for modeling the rules list for the switch

Project Status

a2sdn

Status: Completed – Limited Testing

Code Source: https://github.com/nklapste/a2sdn

Some difficulties encountered in the design of a2sdn was in maintaining a uniform format. Multiple refactoring/restructuring efforts were needed to obtain to the current build of a2sdn. Additionally, eliciting the requirements from the assignment's documentation was difficult for a2sdn. There were lots of little details that ended up being missed on the first draft of a2sdn. There is also a considerable amount of duplication between the Controller and Switch as they do share a lot of common functionality (e.g. accepting signals, stdin commands, and polling FIFOs). Using a more easily implemented Object Orientated (OO) language such as Java would likely remedy this issue without any major reductions in speed. Note, c++ templates could be used to implement OO, but, they can become quickly complicated and were deemed overkill for this project.

Software such as a2sdn is better off written first in a high-level language (e.g. python/java) for prototyping structure and API concepts (i.e. defining standard packet format, control flows, etc…). Then

finally, if speed is a priority re-constructing the software in a lower language (e.g. c, c++, **rust**) for speedups.

Testing and Results

a2sdn

Testing for a2sdn consisted of simple runtime testing over the basic supported commands and normal use cases.

Some basic test traffic files were created for testing the general functionality of a2sdn. They can be seen bundled with the source code named tf1.txt, tf2.txt, and tf3.txt.

Note: the traffic files tf1.txt, tf2.txt, and tf3.txt are bundled within the submit.tar and are also located within the project's repository for convenience.

Test tf1.txt

The first test for a2sdn involved making a single switch and controller and using the traffic file tf1.txt.

Controller

The initial output of running the controller is shown below:

```
$ ./a2sdn cont 1
DEBUG: creating controller: nSwitches: 1
INFO: Making Connection:
        src: 0 dst: 1 sendFIFO: fifo-0-1
DEBUG: making FIFO at: fifo-0-1
INFO: Making Connection:
        src: 0 dst: 1 receiveFIFO: fifo-1-0
DEBUG: making FIFO at: fifo-1-0
INFO: created controller: nSwitches: 1
DEBUG: opening receiveFIFO: fifo-1-0
```

Next, the switch sw1 was then started and connected to the controller. The controller's stdout response/logs output is show below:

```
DEBUG: pfds[1] has connection POLLIN event: fifo-1-0
DEBUG: parsed packet: OPEN: switchID:1 leftSwitchID:-1 rightSwitchID:-1
IPLow:100 IPHigh:110
DEBUG: parsed OPEN packet: switchID: 1 leftSwitchID: -1 rightSwitchID: -1
switchIPLow: 100 switchIPHigh: 110
DEBUG: adding new switch: switchID: 1 leftSwitchID: -1 rightSwitchID: -1
switchIPLow: 100 switchIPHigh: 110
INFO: sending ACK packet: connection: fifo-0-1 packet: ACK:
DEBUG: opening sendFIFO: fifo-0-1
```

As we can see the controller obtained an OPEN packet and parsed it. It then after parsing the packet sent a ACK packet back to the switch sw1.

Next, the list command was executed on the controller through stdin input. The controller's stdout response/logs output is show below:

```
list
Controller information:
[sw1] port1= -1, port2= -1, port3= 100-110
Packet Stats:
        Received:    OPEN:1, ACK:0, QUERY:0, ADDRULE:0, RELAYIN: 0
        Transmitted: OPEN:0, ACK:1, QUERY:0, ADDRULE:0, RELAYOUT:0
```

As we can see the list command is nice and verbose. And provides extra packet statistics for potential extension and/or debugging purposes.

Next, the SIGUSR1 signal was sent to the controller through the command kill -10 <controllers_pid>. The controller's stdout response/logs output is shown below:

```
DEBUG: received SIGUSR1 signal
Controller information:
[sw1] port1= -1, port2= -1, port3= 100-110
Packet Stats:
        Received:    OPEN:1, ACK:0, QUERY:0, ADDRULE:0, RELAYIN: 0
        Transmitted: OPEN:0, ACK:1, QUERY:0, ADDRULE:0, RELAYOUT:0
```

Lastly, the exit command was executed on the controller through stdin input. The controller's stdout response/logs output is shown below:

```
exit
Controller information:
[sw1] port1= -1, port2= -1, port3= 100-110
Packet Stats:
        Received:    OPEN:1, ACK:0, QUERY:0, ADDRULE:0, RELAYIN: 0
        Transmitted: OPEN:0, ACK:1, QUERY:0, ADDRULE:0, RELAYOUT:0
INFO: exit command received: terminating
```

BONUS: Starting the same switch twice (i.e. restarting sw1 with new config) provides the following stdout from the controller:

Starting switch sw1 with ./a2sdn sw1 tf1.txt null null 100-110 on a newly created controller provides the following stdout from the controller:

```
DEBUG: pfds[1] has connection POLLIN event: fifo-1-0
DEBUG: parsed packet: OPEN: switchID:1 leftSwitchID:-1 rightSwitchID:-1
IPLow:100 IPHigh:110
DEBUG: parsed OPEN packet: switchID: 1 leftSwitchID: -1 rightSwitchID: -1
switchIPLow: 100 switchIPHigh: 110
```

```
DEBUG: adding new switch: switchID: 1 leftSwitchID: -1 rightSwitchID: -1
switchIPLow: 100 switchIPHigh: 110
INFO: sending ACK packet: connection: fifo-0-1 packet: ACK:
DEBUG: opening sendFIFO: fifo-0-1
```

Next restarting switch sw1 with ./a2sdn sw1 tf1.txt null null 200-210 provides the following stdout from the controller:

```
DEBUG: pfds[1] has connection POLLIN event: fifo-1-0
DEBUG: parsed packet: OPEN: switchID:1 leftSwitchID:-1 rightSwitchID:-1
IPLow:200 IPHigh:210
DEBUG: parsed OPEN packet: switchID: 1 leftSwitchID: -1 rightSwitchID: -1
switchIPLow: 200 switchIPHigh: 210
DEBUG: updating existing switch: switchID: 1 leftSwitchID: -1 rightSwitchID:
-1 switchIPLow: 200 switchIPHigh: 210
INFO: sending ACK packet: connection: fifo-0-1 packet: ACK:
DEBUG: opening sendFIFO: fifo-0-1
```

Running the list command on the controller gives the following stdout:

```
list
Controller information:
[sw1] port1= -1, port2= -1, port3= 200-210
Packet Stats:
        Received:    OPEN:2, ACK:0, QUERY:3, ADDRULE:0, RELAYIN: 0
        Transmitted: OPEN:0, ACK:2, QUERY:0, ADDRULE:3, RELAYOUT:0
```

As we can see the controller's stored information on switch sw1 was updated.

Switch (sw1)

The initial output of running the switch sw1 (with the controller already started) is shown below:

```
$ ./a2sdn sw1 tf1.txt null null 100-110
INFO: Making Connection:
        src: 1 dst: 0 sendFIFO: fifo-1-0
DEBUG: making FIFO at: fifo-1-0
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 1 dst: 0 receiveFIFO: fifo-0-1
DEBUG: making FIFO at: fifo-0-1
WARNING: error creating FIFO connection: File exists
INFO: created switch: sw1 trafficFile: tf1.txt swj: null swk: null IPLow: 100
IPHigh: 110
DEBUG: opening receiveFIFO: fifo-0-1
INFO: sending OPEN packet: connection: fifo-1-0 packet: OPEN: switchID:1
leftSwitchID:-1 rightSwitchID:-1 IPLow:100 IPHigh:110
DEBUG: opening sendFIFO: fifo-1-0
DEBUG: read traffic file line: # traffic file for a2sdn
```

```
DEBUG: ignoring comment line
DEBUG: read traffic file line: #       a2sdn cont 1
DEBUG: ignoring comment line
DEBUG: read traffic file line: #       a2sdn sw1 tf1.txt null null 100-110
DEBUG: ignoring comment line
DEBUG: read traffic file line: sw1 100 102
DEBUG: found line specifying self: sw1 100 102
DEBUG: parsed trafficFileItem: switchID: 1 srcIP: 100 dst: 102
DEBUG: searching for FlowEntry for: srcIP: 100 dstIP: 102
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3
pri= 4 pktCount= 0)
DEBUG: pfds[1] has connection POLLIN event: fifo-0-1
DEBUG: Parsed packet: ACK:
INFO: ACK packet received:
DEBUG: read traffic file line: sw2 200 300
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw3 200 300
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw1 100 103
DEBUG: found line specifying self: sw1 100 103
DEBUG: parsed trafficFileItem: switchID: 1 srcIP: 100 dst: 103
DEBUG: searching for FlowEntry for: srcIP: 100 dstIP: 103
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3
pri= 4 pktCount= 1)
DEBUG: read traffic file line: sw1 100 104
DEBUG: found line specifying self: sw1 100 104
DEBUG: parsed trafficFileItem: switchID: 1 srcIP: 100 dst: 104
DEBUG: searching for FlowEntry for: srcIP: 100 dstIP: 104
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3
pri= 4 pktCount= 2)
DEBUG: finished reading traffic file
```

As we can see a lot has gone on within the start-up of the switch such as sending the OPEN , parsing the traffic file, and accepting pollin events from the connection from the controller. Also, a warning was raised noting that the FIFOs fifo-0-1 and fifo-1-0 already exist and were not created. This is not concerning as the controller made these. But, this still acts as a warning to notify the user of the potential of another switch sw1 running (in this test case, however, this is not the case).

Next, the list command was executed on the switch through stdin input. The switch's stdout response/logs output is show below:

```
list
sw1 FlowTable:
[0] (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3 pri= 4 pktCount= 3)
Packet Stats:
        Received:    OPEN:0, ACK:1, QUERY:0, ADDRULE:0, RELAYIN: 0, ADMIT:3
        Transmitted: OPEN:1, ACK:0, QUERY:0, ADDRULE:0, RELAYOUT:0
```

As we can see the list command is nice and verbose. And provides extra packet statistics for potential extension and/or debugging purposes.

Next, the SIGUSR1 signal was sent to the switch through the command kill -10 <switch_pid>. The switch's stdout response/logs output is shown below:

```
DEBUG: received SIGUSR1
sw1 FlowTable:
[0] (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3 pri= 4 pktCount= 3)
Packet Stats:
        Received:    OPEN:0, ACK:1, QUERY:0, ADDRULE:0, RELAYIN: 0, ADMIT:3
        Transmitted: OPEN:1, ACK:0, QUERY:0, ADDRULE:0, RELAYOUT:0
```

Lastly, the exit command was executed on the switch through stdin input. The switch's stdout response/logs output is shown below:

```
exit
sw1 FlowTable:
[0] (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3 pri= 4 pktCount= 3)
Packet Stats:
        Received:    OPEN:0, ACK:1, QUERY:0, ADDRULE:0, RELAYIN: 0, ADMIT:3
        Transmitted: OPEN:1, ACK:0, QUERY:0, ADDRULE:0, RELAYOUT:0
INFO: exit command received: terminating
```

BONUS: Starting the switch without a running controller provides the following stdout from the switch:

```
$ ./a2sdn sw1 tf1.txt null null 100-110
INFO: Making Connection:
        src: 1 dst: 0 sendFIFO: fifo-1-0
DEBUG: making FIFO at: fifo-1-0
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 1 dst: 0 receiveFIFO: fifo-0-1
DEBUG: making FIFO at: fifo-0-1
WARNING: error creating FIFO connection: File exists
INFO: created switch: sw1 trafficFile: tf1.txt swj: null swk: null IPLow: 100
IPHigh: 110
DEBUG: opening receiveFIFO: fifo-0-1
INFO: sending OPEN packet: connection: fifo-1-0 packet: OPEN: switchID:1
leftSwitchID:-1 rightSwitchID:-1 IPLow:100 IPHigh:110
DEBUG: opening sendFIFO: fifo-1-0
ERROR: opening sendFIFO: No such device or address
```

This seems to be acceptable behavior as we do not want a switch to be running without an active controller to manage it.

Test tf2.txt

A test run with a2sdn following a normal use case using the traffic file tf2.txt is shown below:

Controller

The initial output of running the controller is shown below:

```
$ ./a2sdn cont 2
DEBUG: creating controller: nSwitches: 2
INFO: Making Connection:
        src: 0 dst: 1 sendFIFO: fifo-0-1
DEBUG: making FIFO at: fifo-0-1
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 0 dst: 1 receiveFIFO: fifo-1-0
DEBUG: making FIFO at: fifo-1-0
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 0 dst: 2 sendFIFO: fifo-0-2
DEBUG: making FIFO at: fifo-0-2
INFO: Making Connection:
        src: 0 dst: 2 receiveFIFO: fifo-2-0
DEBUG: making FIFO at: fifo-2-0
INFO: created controller: nSwitches: 2
DEBUG: opening receiveFIFO: fifo-1-0
DEBUG: opening receiveFIFO: fifo-2-0
```

Next, the switch sw1 was started and connected to the controller. The controller's stdout response/logs output is shown below:

```
DEBUG: pfds[1] has connection POLLIN event: fifo-1-0
DEBUG: parsed packet: OPEN: switchID:1 leftSwitchID:-1 rightSwitchID:2
IPLow:100 IPHigh:110
DEBUG: parsed OPEN packet: switchID: 1 leftSwitchID: -1 rightSwitchID: 2
switchIPLow: 100 switchIPHigh: 110
DEBUG: adding new switch: switchID: 1 leftSwitchID: 2 rightSwitchID: -1
switchIPLow: 100 switchIPHigh: 110
INFO: sending ACK packet: connection: fifo-0-1 packet: ACK:
DEBUG: opening sendFIFO: fifo-0-1
DEBUG: pfds[1] has connection POLLIN event: fifo-1-0
DEBUG: parsed packet: QUERY: switchID:1 srcIP:100 dstIP:200
DEBUG: parsed QUERY packet: switchID: 1 srcIP: 100 dstIP: 200
DEBUG: making FlowEntry for: switchID: 1 srcIP: 100 dstIP: 200
DEBUG: found matching switch: 1
DEBUG: valid srcIP: 100
DEBUG: invalid dstIP: 200 checking if neighboring switches are valid
DEBUG: checking leftSwitchID: -1
DEBUG: checking rightSwitchID: 2
DEBUG: left and right switch invalid creating DROP FlowEntry
```

```
INFO: sending ADD packet: connection: fifo-0-1 packet: ADD: srcIPLow:0
srcIPHigh:1000 dstIPLow:200 dstIPHigh:200 actionType:2 actionVal:0 pri:4
pktCount:0
DEBUG: opening sendFIFO: fifo-0-1
DEBUG: pfds[1] has connection POLLIN event: fifo-1-0
DEBUG: parsed packet: QUERY: switchID:1 srcIP:100 dstIP:300
DEBUG: parsed QUERY packet: switchID: 1 srcIP: 100 dstIP: 300
DEBUG: making FlowEntry for: switchID: 1 srcIP: 100 dstIP: 300
DEBUG: found matching switch: 1
DEBUG: valid srcIP: 100
DEBUG: invalid dstIP: 300 checking if neighboring switches are valid
DEBUG: checking leftSwitchID: -1
DEBUG: checking rightSwitchID: 2
DEBUG: left and right switch invalid creating DROP FlowEntry
INFO: sending ADD packet: connection: fifo-0-1 packet: ADD: srcIPLow:0
srcIPHigh:1000 dstIPLow:300 dstIPHigh:300 actionType:2 actionVal:0 pri:4
pktCount:0
```

On top of obtaining info on the switch sw1 from it's OPEN packet and sending an ACK back the controller responded to two QUERY packets made by switch sw1. For both QUERY packets the controller had no valid matching switch to match to the QUERY too so for both QUERY packets a DROP FlowEntry rule was created and sent to the switch sw1 through an ADD packet.

Next, the switch sw2 was started and connected to the controller. The controller's stdout response/logs output is shown below:

```
DEBUG: pfds[2] has connection POLLIN event: fifo-2-0
DEBUG: parsed packet: OPEN: switchID:2 leftSwitchID:1 rightSwitchID:-1
IPLow:200 IPHigh:210
DEBUG: parsed OPEN packet: switchID: 2 leftSwitchID: 1 rightSwitchID: -1
switchIPLow: 200 switchIPHigh: 210
DEBUG: adding new switch: switchID: 2 leftSwitchID: -1 rightSwitchID: 1
switchIPLow: 200 switchIPHigh: 210
INFO: sending ACK packet: connection: fifo-0-2 packet: ACK:
DEBUG: opening sendFIFO: fifo-0-2
DEBUG: pfds[2] has connection POLLIN event: fifo-2-0
DEBUG: parsed packet: QUERY: switchID:2 srcIP:200 dstIP:300
DEBUG: parsed QUERY packet: switchID: 2 srcIP: 200 dstIP: 300
DEBUG: making FlowEntry for: switchID: 2 srcIP: 200 dstIP: 300
DEBUG: found matching switch: 2
DEBUG: valid srcIP: 200
DEBUG: invalid dstIP: 300 checking if neighboring switches are valid
DEBUG: checking leftSwitchID: 1
DEBUG: checking rightSwitchID: -1
DEBUG: left and right switch invalid creating DROP FlowEntry
INFO: sending ADD packet: connection: fifo-0-2 packet: ADD: srcIPLow:0
srcIPHigh:1000 dstIPLow:300 dstIPHigh:300 actionType:2 actionVal:0 pri:4
pktCount:0
```

```
DEBUG: opening sendFIFO: fifo-0-2
DEBUG: pfds[2] has connection POLLIN event: fifo-2-0
DEBUG: parsed packet: QUERY: switchID:2 srcIP:200 dstIP:100
DEBUG: parsed QUERY packet: switchID: 2 srcIP: 200 dstIP: 100
DEBUG: making FlowEntry for: switchID: 2 srcIP: 200 dstIP: 100
DEBUG: found matching switch: 2
DEBUG: valid srcIP: 200
DEBUG: invalid dstIP: 100 checking if neighboring switches are valid
DEBUG: checking leftSwitchID: 1
DEBUG: left switch valid creating FORWARD FlowEntry
INFO: sending ADD packet: connection: fifo-0-2 packet: ADD: srcIPLow:0
srcIPHigh:1000 dstIPLow:100 dstIPHigh:110 actionType:1 actionVal:1 pri:4
pktCount:0
DEBUG: opening sendFIFO: fifo-0-2
```

Next, the list command was executed on the controller through stdin input. The controller's stdout response/logs output is shown below:

```
list
Controller information:
[sw1] port1= -1, port2= 2, port3= 100-110
[sw2] port1= 1, port2= -1, port3= 200-210
Packet Stats:
        Received:    OPEN:2, ACK:0, QUERY:4, ADDRULE:0, RELAYIN: 0
        Transmitted: OPEN:0, ACK:2, QUERY:0, ADDRULE:4, RELAYOUT:0
```

As we can see after starting both switches sw1 and sw2 controller has knowledge on both them.

Switch (sw1)

The initial output of running the switch sw1 (with the controller already started) is shown below:

As shown switch sw1 created and sent two QUERY packets and was given two drop rules as a response from the controller.

```
./a2sdn sw1 tf2.txt null sw2 100-110
INFO: Making Connection:
        src: 1 dst: 0 sendFIFO: fifo-1-0
DEBUG: making FIFO at: fifo-1-0
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 1 dst: 0 receiveFIFO: fifo-0-1
DEBUG: making FIFO at: fifo-0-1
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 1 dst: 2 sendFIFO: fifo-1-2
DEBUG: making FIFO at: fifo-1-2
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
```

```
        src: 1 dst: 2 receiveFIFO: fifo-2-1
DEBUG: making FIFO at: fifo-2-1
WARNING: error creating FIFO connection: File exists
INFO: created switch: sw1 trafficFile: tf2.txt swj: null swk: sw2 IPLow: 100
IPHigh: 110
DEBUG: opening receiveFIFO: fifo-0-1
DEBUG: opening receiveFIFO: fifo-2-1
INFO: sending OPEN packet: connection: fifo-1-0 packet: OPEN: switchID:1
leftSwitchID:-1 rightSwitchID:2 IPLow:100 IPHigh:110
DEBUG: opening sendFIFO: fifo-1-0
DEBUG: read traffic file line: # traffic file for a2sdn
DEBUG: ignoring comment line
DEBUG: pfds[1] has connection POLLIN event: fifo-0-1
DEBUG: Parsed packet: ACK:  :
INFO: ACK packet received:
DEBUG: read traffic file line: #       a2sdn cont 1
DEBUG: ignoring comment line
DEBUG: read traffic file line: #       a2sdn sw1 tf2.txt null sw2 100-110
DEBUG: ignoring comment line
DEBUG: read traffic file line: #       a2sdn sw2 tf2.txt sw1 null 200-210
DEBUG: ignoring comment line
DEBUG: read traffic file line: sw1 100 102
DEBUG: found line specifying self: sw1 100 102
DEBUG: parsed trafficFileItem: switchID: 1 srcIP: 100 dst: 102
DEBUG: searching for FlowEntry for: srcIP: 100 dstIP: 102
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3
pri= 4 pktCount= 0)
DEBUG: read traffic file line: sw1 100 200
DEBUG: found line specifying self: sw1 100 200
DEBUG: parsed trafficFileItem: switchID: 1 srcIP: 100 dst: 200
DEBUG: searching for FlowEntry for: srcIP: 100 dstIP: 200
INFO: sending QUERY packet: connection: fifo-1-0 packet: QUERY: switchID:1
srcIP:100 dstIP:200
DEBUG: opening sendFIFO: fifo-1-0
DEBUG: pfds[1] has connection POLLIN event: fifo-0-1
DEBUG:   Parsed   packet:   ADD:   srcIPLow:0   srcIPHigh:1000   dstIPLow:200
dstIPHigh:200 actionType:2 actionVal:0 pri:4 pktCount:0
INFO: parsed ADD packet:
        Adding new flowTable rule:
                srcIP_lo:  0  srcIPHigh:  1000  dstIPLow:  200  dstIPHigh:  200
actionType: 2 actionVal: 0 pri: 4 pktCount: 0 INFO: parsed RELAY packet:
        switchID: 1 srcIP: 100 dstIP: 200DEBUG: searching for FlowEntry for:
srcIP: 100 dstIP: 200
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 200-200 action=DROP:0
pri= 4 pktCount= 0)
DEBUG: read traffic file line: sw1 100 300
DEBUG: found line specifying self: sw1 100 300
DEBUG: parsed trafficFileItem: switchID: 1 srcIP: 100 dst: 300
DEBUG: searching for FlowEntry for: srcIP: 100 dstIP: 300
```

```
INFO: sending QUERY packet: connection: fifo-1-0 packet: QUERY: switchID:1
srcIP:100 dstIP:300
DEBUG: opening sendFIFO: fifo-1-0
DEBUG: pfds[1] has connection POLLIN event: fifo-0-1
DEBUG:   Parsed   packet:   ADD:   srcIPLow:0   srcIPHigh:1000   dstIPLow:300
dstIPHigh:300 actionType:2 actionVal:0 pri:4 pktCount:0
INFO: parsed ADD packet:
        Adding new flowTable rule:
                srcIP_lo:  0  srcIPHigh:  1000  dstIPLow:  300  dstIPHigh:  300
actionType: 2 actionVal: 0 pri: 4 pktCount: 0 INFO: parsed RELAY packet:
        switchID: 1 srcIP: 100 dstIP: 300DEBUG: searching for FlowEntry for:
srcIP: 100 dstIP: 300
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 300-300 action=DROP:0
pri= 4 pktCount= 0)
DEBUG: read traffic file line: sw2 200 300
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw3 200 300
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw2 200 100
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw2 200 100
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw1 100 103
DEBUG: found line specifying self: sw1 100 103
DEBUG: parsed trafficFileItem: switchID: 1 srcIP: 100 dst: 103
DEBUG: searching for FlowEntry for: srcIP: 100 dstIP: 103
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3
pri= 4 pktCount= 1)
DEBUG: read traffic file line: sw1 100 104
DEBUG: found line specifying self: sw1 100 104
DEBUG: parsed trafficFileItem: switchID: 1 srcIP: 100 dst: 104
DEBUG: searching for FlowEntry for: srcIP: 100 dstIP: 104
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3
pri= 4 pktCount= 2)
DEBUG: finished reading traffic file
```

Next the switch sw2 was started. Switch sw1 responds with the following stdout:

```
DEBUG: pfds[3] has connection POLLIN event: fifo-2-1
DEBUG: Parsed packet: RELAY: switchID:2 srcIP:200 dstIP:100
INFO: parsed RELAY packet:
        switchID: 2 srcIP: 200 dstIP: 100DEBUG: searching for FlowEntry for:
srcIP: 200 dstIP: 100
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3
pri= 4 pktCount= 3)
DEBUG: pfds[3] has connection POLLIN event: fifo-2-1
DEBUG: Parsed packet: RELAY: switchID:2 srcIP:200 dstIP:100
INFO: parsed RELAY packet:
```

```
        switchID: 2 srcIP: 200 dstIP: 100DEBUG: searching for FlowEntry for:
srcIP: 200 dstIP: 100
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3
pri= 4 pktCount= 4)
```

Next, the list command was executed on the switch through stdin input. The switch's stdout response/logs output is shown below:

```
list
sw1 FlowTable:
[0] (srcIP= 0-1000 dstIP 100-110 action=DELIVER:3 pri= 4 pktCount= 5)
[1] (srcIP= 0-1000 dstIP 200-200 action=DROP:0 pri= 4 pktCount= 1)
[2] (srcIP= 0-1000 dstIP 300-300 action=DROP:0 pri= 4 pktCount= 1)
Packet Stats:
        Received:    OPEN:0, ACK:1, QUERY:0, ADDRULE:2, RELAYIN: 2, ADMIT:5
        Transmitted: OPEN:1, ACK:0, QUERY:2, ADDRULE:0, RELAYOUT:0
```

Switch (sw2)

The initial output of running the switch sw2 (with the controller already started) is shown below:

```
$ ./a2sdn sw2 tf2.txt sw1 null 200-210
INFO: Making Connection:
        src: 2 dst: 0 sendFIFO: fifo-2-0
DEBUG: making FIFO at: fifo-2-0
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 2 dst: 0 receiveFIFO: fifo-0-2
DEBUG: making FIFO at: fifo-0-2
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 2 dst: 1 sendFIFO: fifo-2-1
DEBUG: making FIFO at: fifo-2-1
WARNING: error creating FIFO connection: File exists
INFO: Making Connection:
        src: 2 dst: 1 receiveFIFO: fifo-1-2
DEBUG: making FIFO at: fifo-1-2
WARNING: error creating FIFO connection: File exists
INFO: created switch: sw2 trafficFile: tf2.txt swj: sw1 swk: null IPLow: 200
IPHigh: 210
DEBUG: opening receiveFIFO: fifo-0-2
DEBUG: opening receiveFIFO: fifo-1-2
INFO: sending OPEN packet: connection: fifo-2-0 packet: OPEN: switchID:2
leftSwitchID:1 rightSwitchID:-1 IPLow:200 IPHigh:210
DEBUG: opening sendFIFO: fifo-2-0
DEBUG: read traffic file line: # traffic file for a2sdn
DEBUG: ignoring comment line
DEBUG: pfds[1] has connection POLLIN event: fifo-0-2
DEBUG: Parsed packet: ACK:  :
INFO: ACK packet received:
```

```
DEBUG: read traffic file line: #       a2sdn cont 1
DEBUG: ignoring comment line
DEBUG: read traffic file line: #       a2sdn sw1 tf2.txt null sw2 100-110
DEBUG: ignoring comment line
DEBUG: read traffic file line: #       a2sdn sw2 tf2.txt sw1 null 200-210
DEBUG: ignoring comment line
DEBUG: read traffic file line: sw1 100 102
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw1 100 200
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw1 100 300
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw2 200 300
DEBUG: found line specifying self: sw2 200 300
DEBUG: parsed trafficFileItem: switchID: 2 srcIP: 200 dst: 300
DEBUG: searching for FlowEntry for: srcIP: 200 dstIP: 300
INFO: sending QUERY packet: connection: fifo-2-0 packet: QUERY: switchID:2
srcIP:200 dstIP:300
DEBUG: opening sendFIFO: fifo-2-0
DEBUG: pfds[1] has connection POLLIN event: fifo-0-2
DEBUG:   Parsed   packet:   ADD:   srcIPLow:0   srcIPHigh:1000   dstIPLow:300
dstIPHigh:300 actionType:2 actionVal:0 pri:4 pktCount:0
INFO: parsed ADD packet:
        Adding new flowTable rule:
                srcIP_lo:  0  srcIPHigh:  1000  dstIPLow:  300  dstIPHigh:  300
actionType: 2 actionVal: 0 pri: 4 pktCount: 0 INFO: resolving packet:
        srcIP: 200 dstIP: 300DEBUG: searching for FlowEntry for: srcIP: 200
dstIP: 300
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 300-300 action=DROP:0
pri= 4 pktCount= 0)
DEBUG: read traffic file line: sw3 200 300
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw2 200 100
DEBUG: found line specifying self: sw2 200 100
DEBUG: parsed trafficFileItem: switchID: 2 srcIP: 200 dst: 100
DEBUG: searching for FlowEntry for: srcIP: 200 dstIP: 100
INFO: sending QUERY packet: connection: fifo-2-0 packet: QUERY: switchID:2
srcIP:200 dstIP:100
DEBUG: opening sendFIFO: fifo-2-0
DEBUG: pfds[1] has connection POLLIN event: fifo-0-2
DEBUG:   Parsed   packet:   ADD:   srcIPLow:0   srcIPHigh:1000   dstIPLow:100
dstIPHigh:110 actionType:1 actionVal:1 pri:4 pktCount:0
INFO: parsed ADD packet:
        Adding new flowTable rule:
                srcIP_lo:  0  srcIPHigh:  1000  dstIPLow:  100  dstIPHigh:  110
actionType: 1 actionVal: 1 pri: 4 pktCount: 0 INFO: resolving packet:
        srcIP: 200 dstIP: 100DEBUG: searching for FlowEntry for: srcIP: 200
dstIP: 100
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=FORWARD:1
pri= 4 pktCount= 0)
```

```
INFO: sending RELAY packet: connection: fifo-2-1 packet: RELAY: switchID:2
srcIP:200 dstIP:100
DEBUG: opening sendFIFO: fifo-2-1
DEBUG: read traffic file line: sw2 200 100
DEBUG: found line specifying self: sw2 200 100
DEBUG: parsed trafficFileItem: switchID: 2 srcIP: 200 dst: 100
DEBUG: searching for FlowEntry for: srcIP: 200 dstIP: 100
DEBUG: found matching FlowEntry: (srcIP= 0-1000 dstIP 100-110 action=FORWARD:1
pri= 4 pktCount= 1)
INFO: sending RELAY packet: connection: fifo-2-1 packet: RELAY: switchID:2
srcIP:200 dstIP:100
DEBUG: opening sendFIFO: fifo-2-1
DEBUG: read traffic file line: sw1 100 103
DEBUG: ignoring line specifying another switch
DEBUG: read traffic file line: sw1 100 104
DEBUG: ignoring line specifying another switch
DEBUG: finished reading traffic file
```

Next, the list command was executed on the switch through stdin input. The switch's stdout response/logs output is shown below:

```
list
sw2 FlowTable:
[0] (srcIP= 0-1000 dstIP 100-110 action=FORWARD:1 pri= 4 pktCount= 2)
[1] (srcIP= 0-1000 dstIP 200-210 action=DELIVER:3 pri= 4 pktCount= 0)
[2] (srcIP= 0-1000 dstIP 300-300 action=DROP:0 pri= 4 pktCount= 1)
Packet Stats:
        Received:    OPEN:0, ACK:1, QUERY:0, ADDRULE:2, RELAYIN: 0, ADMIT:3
        Transmitted: OPEN:1, ACK:0, QUERY:2, ADDRULE:0, RELAYOUT:2
```

Acknowledgments

cplusplus.com for C++ documentation:

http://www.cplusplus.com/

CLion IDE for C++ refactoring and development tools:

https://www.jetbrains.com/clion/

Tutorial on Cmake:

https://www.cs.swarthmore.edu/~adanner/tips/cmake.php