Objectives

The main objective for this assignment was to gain experience writing programs capable of peer-to-peer communication. In order to solve the SDN problem proposed by the specification, I had to improve my understanding of FIFOs, I/O multiplexing and non-blocking I/O. For me, being able to actually understand these concepts and apply them in a practical scenario was of way more interest than the actual SDN itself. By the end of the implementation of the SDN, I feel a lot more comfortable with these concepts.

Design Overview

When the user runs the program, they need to specify whether they want to invoke it as a controller, or invoke it as a switch. This is done via the command line argument as per the specification for this assignment. After tokenizing the user input, one of two main functions are called: executeController() and executeSwitch().

When run as a controller, the following steps are performed:
- The controller opens up the necessary FIFOs for communication based off of the number of switches specified when the program was run. Also, it opens up a FIFO for keyboard polling.
- The controller forks a child process. The purpose of this child process is to be put into an infinite loop where it's sole purpose is to read keyboard commands from the terminal. This allows the parent process to continue looping and polling through its program without being blocked by waiting for user input.
- When the child process receives input (either 'list' or 'exit') it packages the command (similar to the lab exercise on eClass) and sends it to the keyboard FIFO connected to the controller.
- The controller polls the keyboard FIFO and when there is a packaged message waiting for it, it extracts the user command from it and executes it ('list' prints all the details required from the specification. 'exit' command does the same as 'list' except it exits the program afterwards)
- After polling user input (if there was any), the controller then polls the FIFOs attached to the switches.
- When receiving data from the switches, it checks what type of packet it was (OPEN or QUERY) and processes it in accordance to the assignment specification.
- Controllers respond to QUERY packets by creating a new rule and returning it to the switch.
- Controllers respond to OPEN packets by adding the switch's details to its list of known connected switches. It then replies to the switch with an ACK packet.
- After polling the switch FIFOs it loops back around and polls for user input. All the polling is non-blocking.

When run as a switch, the following are performed:
- The switch opens up all necessary FIFOs. Up to total of 7 for neighbouring switches and controller and keyboard FIFO.
- Sends an OPEN packet to the FIFO the controller reads from. This blocks the program until the controller sends an ACK packet back to the FIFO it writes to and this switch reads from.
- Switch forks the process for the same reason as the controller (to handle keyboard polling)
- Switch reads a (relevant) line from specified trafficFile and processes it
- Sends QUERY to controller if no rule for the specified line from trafficFile

- If QUERY is sent, switch waits until controller polls and sends back a new rule via an ADD packet.
- If rule exists, the switch processes the line from the traffic file based on the rule.
- The processing of the lines obey the specifications for assignment 2. (Can drop packets if destination switch is undefined, else it forwards it to the destination switch. If the destination switch is not adjacent to the current switch, it forwards the packet towards the destination switch. The packet will continue to be processed down this linear chain until it reaches its destination)
- User input is polled similar to the controller above
- Ports 1 and 2 of the switch are polled to see if any adjacent switches forwarded a packet to the current switch. If yes, this switch processes it similar to the process above.
- The switch loops the processes above until all lines relevant to it are read, after this it continues to poll ports 1 and 2 and listening for user input.

The relaying of messages follow the same framework as the lab exercise provided by E. Elmallah (URL in acknowledgments).

Assumptions

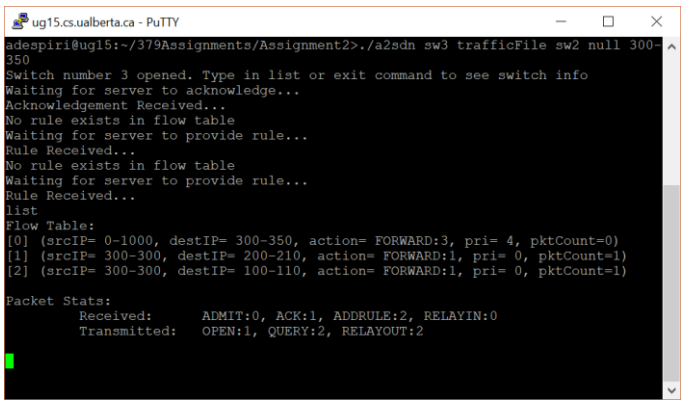
I am working under the assumption that all switch-to-switch or switch-to-controller FIFOs are provided in the current testing directory. As such I have not included them in my tar file. I did include the keyboard FIFOs for the switches and controller since I am not certain if they will be provided when tested.

Project Status

The project is functioning as intended. All specifications that were required are implemented (at least to my knowledge). Error handling could use a little bit of work, but it works for all the major cases (i.e user puts in improper command line arguments when invoking the program). However, it is not perfect and if the tester really wants to break they system, they probably can.

Testing and Results

I created my own test cases by modifying the trafficFile. I also created the necessary FIFOs in order to test the program. I followed the example in the specifications, and after verifying it worked for those cases, I also created another case where a packet had to be relayed multiple times until it reaches its destination. My program worked in all cases.



***Sample Screenshot of one of the switches**

Acknowledgments
Title: Reading Text Files
http://www.fredosaurus.com/notes-cpp/io/readtextfile.html
Author: Fred Swartz, 2003

Title: How to Tokenize a String
https://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm
Author: TutorialsPoint

Title**:** Iteration over std::vector: unsigned vs signed index variable
https://stackoverflow.com/questions/409348/iteration-over-stdvector-unsigned-vs-signed-index-variable
Answered By: Johannes Schaub, 2009

Title: Reading Next Line of a File
http://www.cplusplus.com/forum/beginner/11304/
Answered By: Duthomhas, 2009

Title: Experiments With Sending and Receiving Formatted Messages
http://webdocs.cs.ualberta.ca/~c379/F18/379only/lab-messages.html
Author: E. Elmallah (course instructor)

Title: Experiments With Tokenizing Strings
http://webdocs.cs.ualberta.ca/~c379/F18/379only/lab-tokenizing.html
Author: E.Elmallah (course instructor)

Use of FIFOs
Advanced Programming in the Unix Environment: Third Edition
Authors: W. Richard Stevens, Stephen A. Rago

Title: How Do I Use Poll()?
http://www.unixguide.net/unix/programming/2.1.2.shtml
Author: Not Stated

How to Flush a Formatted String
https://stackoverflow.com/questions/1716296/why-does-printf-not-flush-after-the-call-unless-a-newline-is-in-the-format-strin
Answered By: Rudd Zwolinski, 2009