CMPUT379 – Assign 3
Project Report

Brady Pomerleau

Objectives:

To get an introduction to using sockets and fifos for interprocess communication, as well as an introduction to Software Defined Networks (SDN).

Design Overview:

- a3sdn
  - program can be executed in 2 modes:
    - controller: the network manager. Invoked from the command line using form:
      a3sdn const <nSwitch> <port>
      where nSwitch is the number of switches to be connected (upper limit is 7), port is the tcp port to communicate on
    - Switch: router in the network. Invoked from the command line using form:
      a3sdn sw[id] trafficFile sw<port1>/null sw<port2>/null <range> <controllerAddress> <port>
      where id is the number of the switch, port 1 is the switch on the left (null if there is none), port2 is the switch on the right(null if there is none), range is the set of addresses serviced by the switch, controllerAddress is the network address of the controller (specified as domain name or in dotted decimal notation) and port is the port number of the controller.
  - Switches are arranged in a line from left to right, starting at 1.
  - A switch starts with one rule (to route its packets to its own serviceable network), and can query additional rules from the controller when it doesn't know what to do with a packet.
  - Switches read from the trafficfile. Comments, blank lines, and lines designated for other switches are ignored. There are two types of admitted lines, a delay message and a normal packet. Delay messages specify a time in milliseconds that the switch will defer reading from the trafficfile again (though all other activities should continue). Normal packets are handled either with an existing rule, or with a rule obtained after querying the controller.
  - Packets are passed along the network to the destination if the packet is serviceable by one of the switches in the network.
  - Typing list in either execution mode prints information about the traffic run through the controller/switch.

- Typing exit prints the same information and then exits the program.
- If a switch disconnects the controller should recognize this and print an appropriate message indicating the event.

Project Status:

All functionality is implemented as described.

Notes:

Controller waits until all switches have connected. This results in no serviceable packets being dropped.

Testing and Results:

The following tests were conducted on university lab computers:

Test	Description	Result
t1.dat	Run t1.dat. Obeserve that handshake occurs and correct stats reported after t1.dat has been parsed.	success
t2.dat	Run t2.dat. Obeserve that queries are responded to with correct rules, packets are not dropped, and correct stats reported after t2.dat has been parsed.	success
t3.dat	Run t3.dat. Obeserve that packets are passed along several routers and correct stats reported after t1.dat has been parsed.	success
List and exit	List and exit perform as intended	success
t4.dat	Run t4.dat. Observe that sw1 prints the appropriate delay message, stops reading trafficfile, still handles relay from sw2, and then finishes processing trafficfile after delay time has elapsed.	success
Disconnect switch	Controller should detect and print message	~success - Detects when read or write fails, does nothing otherwise. That means inactive switch disconnecting will not be detected, but active switch disconnections will be reported

Acknowledgements:

To complete this assignment I used the following resources:

- Course textbooks:
  - Operating Systems Concepts ver.9
  - Advanced Programming in the Unix Environment ver.3
- various C programming errors troubleshooting accomplished using Google search and forums (primarily stackoverflow.com)
- socket connection code from https://beej.us/guide/bgnet/html/multi/getaddrinfoman.html