AMAN ANAND

Student ID – 1481975

Assignment #3

CMPUT-379

Objectives:

The objective of the assignment was to familiarize us with developing peer-to-peer programs that utilize sockets for communication between controller and switches, FIFO's for communication between the switches, I/O multiplexing for nonblocking I/O for read/write messages, and signals for examining the progress of the running processes. The assignment treated as a method to make us comfortable coding in UNIX environment and get familiarize with the low-level language such as C/C++ to manage communication using sockets and file descriptors. The two main parts of the assignment were controller and switch(s) (1-7). The controller acted as a server and helped the switches (clients) to communicate with each other using packets. The main packets used were OPEN, ACK, QUERY, RELAY & ADD.

Design Overview:

The important feature of the design are as follows:

a3sdn

- Main Function deciding to run as server (controller) or client (switch)
- Parsed the input from command line and called the Controller () / Switch ()

Controller

- Server of the program
- Communicates with all switches to send Rules (either Drop, Forward packet)

- Detects if a switch connected, terminates unexpectedly and prints a relevant message to screen
- Catches the SIGUSR1 signal and handler prints the Switches connected and Packet Stats

Switch

- Client of the program
- Communicates with controller and the switches to send and receive packets
- Converts the user input server address to IP address and communicates to the controller through the port number provided by the user
- Catches SIGUSR1 signal and handler prints the Switches connected and Packet Stats

Project Status:

The project is working according to the requirement specifications. All tests provided in the specification (Example 1,2,3) under Asssignment#2 and (Example 2) under Assignment#3 work accordingly.

The difficulties faced during the project were:

- Making Recursion to check whether a port is reachable from left or right side of another port (The algorithm is working correctly but I ended up not using it because the specification only required sw1-7 with connected in ascending order). But the algorithms can be used in a generic client server application.
- Closing file descriptors at the wrong time were causing errors in reading at another end of FIFO.

Overall, the project was fun, and I learnt a lot, specially using socket (), poll () and FIFOs.

Testing and Results:

The testing of the program was done by the following steps:

1. Running the program as server (controller) on one terminal
2. Running the program as client (switch 1) in another terminal
3. Running the program as client (switch 2) in another terminal. and so on till switch 7
4. Running kill -s SIGUSR1 commands in another terminal, to test the signal handler

The program was tested with quite a lot of traffic files:

- Example 1,2,3 provided
- Made my own traffic file and connected 7 switches to controller

All the example tests given the specification ran and gave correct results as mentioned in the requirements.

Acknowledgments:

Most assistance were taken from the following links:

1. Documentation of C++ language - https://www.tutorialspoint.com/cplusplus/
2. Advanced Programming In UNIX environment - https://proquest-safaribooksonline-com.login.ezproxy.library.ualberta.ca/book/programming/unix/9780321638014