CMPUT 379 - Assignment #3 (10%)

Communicating Peer Processes using TCP Sockets and FIFOs (first draft)

Due: Monday, March 21, 2022, 09:00 PM
(electronic submission)

Objectives

This programming assignment is intended to give you experience in developing client-server programs that utilize TCP sockets for communication over the Internet, FIFOs for communication among peers, and I/O multiplexing (select or poll) for nonblocking I/O.

Details

In this assignment, you are asked to write a C/C++, called `a3w22`, that implements the transactions performed by the simplified hypothetical data center network described in Assignment 2. As in Assignment 2, each switch communicates with its neighbour switches using FIFOs. However, unlike Assignment 2, the packet switches communicate with the master switch over TCP sockets.

The program can be invoked as a TCP server (simulating a master switch), or a TCP client (simulating a packet switch) using:

```
% a3w22 master nSwitch portNumber
```

```
% a3w22 pswi dataFile (null|pswj) (null|pswk) IPlow-IPhigh serverAddress portNumber
```

where, in addition to the command line arguments in Assignment 2, the program uses the following arguments:

- `portNumber` specifies the TCP port used by the server to listen to incoming connections. (To minimize the chance of port number conflicts when running servers in the lab, use port number $9xxx$ where xxx is the last 3 digits of your student ID number.)
- `serverAddress` is the IP address of the server's host specified either as a symbolic name (e.g., `ui11.cs.ualberta.ca`, or `localhost`), or in dotted-decimal notation (e.g., `127.0.0.1`).

Features from Assignment 2

Your `a3w22` program should implement the following features (detailed in Assignment 2).

1. Data transmission among adjacent packet switches uses FIFOs. Each FIFO is named as `fifo-x-y` where $\texttt{x} \neq \texttt{y}$, and $\texttt{x}, \texttt{y} \in [1, \texttt{MAX\_NSW}]$ and $\texttt{MAX\_NSW} = 7$ switches. Such a FIFO carries packets in the direction from switch x to switch y. For simplicity, you may create the needed FIFOs before running the program.

2. Each switch handles data from hosts in the specified IP range `[IPlow-IPhigh]`, and each IP address is $\leq$ `MAXIP` $(= 1000)$.

3. Each row in a switch's forwarding table stores a *pattern-plus-action* rule composed of the following fields:

srcIP_lo	srcIP_hi	destIP_lo	destIP_hi	actionType	actionVal	pktCount

   where `actionType` can be either *forward* or *drop* a matched packet.

4. Communication between switches uses messages stored in formatted packets. Each packet has a type, and carries a message (except `HELLO_ACK` packets). Your program should support the packet types described in Assignment 2: `HELLO`, `HELLO_ACK`, `ASK`, `ADD`, and `RELAY`.

5. When `a3w22` is invoked to simulate a device (a packet switch or a master switch), the program uses I/O multiplexing (e.g., `select()` or `poll()`) to handle I/O from the keyboard, and the attached devices in a nonblocking manner. Each iteration of the main loop of a device polls the keyboard for the **info** or **exit** commands, and polls other attached devices. Executing the `info` and `exit` commands are done as in Assignment 2.

6. When `a3w22` is invoked to simulate a packet switch, the program installs an initial rule in its forwarding table:

srcIP_lo $= 0$	srcIP_hi $=$ MAXIP	destIP_lo $=$ IPlow	destIP_hi $=$ IPhigh	actionType $=$ FORWARD	actionVal $= 3$	pktCount $= 0$

7. Upon receiving signal `SIGUSR1`, a switch displays the information specified in the `info` command.

8. **Data File:** Each iteration of the main loop of a switch reads and processes a single line from the data file (if the EOF has not been reached yet). The switch ignores empty lines, comment lines, and lines specifying other handling switches. A packet header in the input file is considered *admitted* if the line specifies the current switch. A delay line of the form `"pswi delay interval"` where `interval` is an integer in milliseconds, specifies that packet switch i should delay reading (and processing) the remaining part of the data file for the specified time interval.

   Note 1: During a delay period, the switch should continue monitoring and processing keyboard commands and packets received from the attached devices.

   Note 2: After reading all lines in the data file, the program continues to monitor and process keyboard commands, and the incoming packets from neighbouring devices.

New Features

In addition to implementing the above features from Assignment 2, your program should implement the following new features.

1. The master switch should detect if an attached packet switch closes its end of the socket (e.g., because a switch has terminated unexpectedly). When such an event occurs, the master switch should write a suitable message, e.g., `"lost connection to psw3"`.

2. Unexpected termination of a packet switch should not cause any other device to stop functioning properly in the new configuration.

3. **Output:** In addition to printing the output specified in Assignment 2 when the `info`, or `exit` commands are issued to a switch, each device should print all received and transmitted packets, as shown in Example output 3 of Assignment 2. Also as shown in Example 3, a switch entering a delay interval (as may be specified by a line in the data file) should print a message to this effect, e.g., `"Entering a delay period for 300 millisec"`.

More Details

1. This is an individual assignment. Do not work in groups.

2. Only standard include files and libraries provided when you compile the program using `gcc` or `g++` should be used.

3. Although many details about this assignment are given in this description, there are many other design decisions that are left for you to make. In such cases, you should make reasonable design decisions that do not contradict what we have said and do not significantly change the purpose of the assignment. Document such design decisions in your source code, and discuss them in your report. Of course, you may ask questions about this assignment (e.g., in the Discussion Forum) and we may choose to provide more information or provide some clarification. However, the basic requirements of this assignment will not change.

4. When developing and testing your program, **make sure you clean up all processes before you logout of a workstation.** Marks will be deducted for processes left on workstations.

Deliverables

1. All programs should compile and run on the lab machines (e.g., ug[00 to 34].cs.ualberta.ca) using only standard libraries (e.g., standard I/O library, math, and pthread libraries are allowed).

2. Make sure your programs compile and run in a fresh directory.

3. Your work (including a Makefile) should be combined into a single tar archive 'submit.tar', or 'submit.tar.gz'.

   (a) Executing 'make' or 'make `a3w22`' should produce the `a3w22` executable file.

   (b) Executing 'make clean' should remove unneeded files produced in compilation.

   (c) Executing 'make tar' should produce the tar archive.

(d) Your code should include suitable internal documentation of the key functions. If you use code from the textbooks, or code posted on eclass, acknowledge the use of the code in the internal documentation. Make sure to place such acknowledgments in close proximity of the code used.

4. Typeset a project report (e.g., one to three pages either in HTML or PDF) with the following (minimal set of) sections:

   – **Objectives:** state the project objectives and value from your point of view (which may be different from the one mentioned above)

   – **Design Overview:** highlight in point-form the important features of your design

   – **Project Status:** describe the status of your project; mention difficulties encountered in the implementation

   – **Testing and Results:** comment on how you tested your implementation

   – **Acknowledgments:** acknowledge sources of assistance

5. Upload your tar archive using the **Assignment #3 submission/feedback** link on the course's web page. Late submission is available for 24 hours for a penalty of 10% of the points assigned to the phase.

6. It is strongly suggested that you **submit early and submit often**. Only your **last successful submission** will be used for grading.

Marking

Roughly speaking, the breakdown of marks is as follows:

20% : successful compilation of reasonably complete program that is: modular, logically organized, easy to read and understand, includes error checking after important function calls, and acknowledges code used from the textbooks or the posted lab material

05% : ease of managing the project using the makefile

65% : correctness of implementing the master switch and the switch modules and displaying the specified outputs.

10% : quality of the information provided in the project report

---