CS544 -- Computer Networks
Spring 2017-2018
Professor Mike Kain

Network Protocol Implementation

Due Date: Friday, June 8th, 2018 at 11:59pm EDT (although it can be turned in early – to allow you time to study for the final).

Your final assignment is to implement **your protocol** (probably an application/session layer) by writing a concurrent server (one that handles multiple clients simultaneously) and a client that use your stateful protocol. The goal of this assignment is to complete the process of designing a network protocol by implementing it, and showing that there is feedback in the design process through implementation. Depending on your protocol, it may be peer-to-peer or client/server.

REQUIREMENTS

Language
Both the client and server can be written in (almost) any language, but C/C++, JAVA, or Python is preferred. If use another language other than those, you must obtain approval from the Professor before starting.

Platform
To receive full credit, we must be able to run your application. You need to specify how to compile and run your assignment on what operating systems (Linux or Windows preferred). You must provide all source code as well as executables. You must provide instructions on how to compile and run your implementation including but not limited to: CLASSPATH (or other path), command line arguments, and any other details.

Protocol Requirements
The protocol must display the following features:

- **STATEFUL**: Both the client and server must implement and check & validate the statefulness of your protocol. THIS IS IMPORTANT. In other words, you must implement the DFA you created for your protocol design. If your protocol DFA did not earn full credit, you must fix the DFA in order to earn full credit for its implementation.

- **CONCURRENT**: The server must be able to handle multiple clients (either using a process/thread model or select) – both can be found in any programming text or see Ehsan or myself.

- **SERVICE**: The server must bind to a hardcoded port number (you pick this value) and the client defaults to that port number. This must be documented in your protocol design.

- **CLIENT**: The client must be able to specify the hostname or IP address of the server (be able to specify either). If using peer-to-peer, you must be able to locate any services that you need.

- **UI**: The user interface of the client isn't too important here – a command line UI is acceptable. It will be up to your protocol, but the protocol logic should be inside the client, not visible through the UI only the client and server should know the actual protocol commands). In other words, the user should not have to know the protocol commands to get it to work.

You are required to provide documentation describing how your project satisfies the Protocol Requirements. You must label areas of your code which address each of these requirements. With your submission, include a file that details what files, classes, etc. satisfy each Protocol Requirement. Within each file, you must identify the segments of code that satisfy the requirement. For instance, the code that implements your DFA should begin with a comment that says "STATEFUL". In the document, you must also indicate whether or not you implemented the extra credit and how.

NOTES:

- Porting of the code for the socket algorithm(s) (the shell of the client and server) is acceptable – but every procedure MUST be accompanied by a reference in the header of where you took the routine (like a term paper). You MUST write your protocol implementation from scratch.
- **Any use of open source frameworks or toolkits must be approved by the Professor before use.**

Turn in (zipped together as first letter last name.zip) through Drexel Learn:

- An updated complete proposal document including any changes or updates we suggested and a section describing any differences since the original submission (called out as the differences in the second version of the document). The implementation must agree with the protocol design. Also include any performance implications that you find during your coding and testing. This document should be titled Updated_first letter last name.<ext>.
- All (well-commented) source code. My basic rule is that all code should be commented in a way that others can pick up the code and instantly understand it – write comments the way you would want to see them if you picked up code and had to learn it (basically a lot of comments – each block should have some level of comments). The top of each file should contain the class name, date, and purpose of the file.

  Every function should have a comment block about it's purpose and every file should have a comment block about who wrote the code and it's purpose.
- A sample run of your protocol in action – this could be a script or a video (about 3 to 5 minutes in length). **This can be turned in before the last class (we will be looking at the class protocols during the last class of the semester).**
- A document describing which files satisfy each of the specified requirements. This document should be titled Requirements_Firstletter_lastname<x>.<ext>
- A README containing the following information
  - How to compile and run your programs (including any usercodes and passwords needed).

- Any analysis about how robust your assignment is – do you think that it's tough to crack through fuzzing? Explain how you tested your implementation's robustness.
- Indicate if you implemented the extra credit.

EXTRA CREDIT:
Have the client program dynamically find the server (e.g. just execute *client* with no parameters and the program finds the server listening on that port anywhere in the network). This is all I'll say about the extra credit -- the research is up to you. It is worth up to 10 extra points, depending on how creative you get (the more you use networking, the more points you get). Be sure to mention under what circumstances your approach will or will not work. If you have any questions, please contact Ehsan or myself. We will be around as often as possible to answer questions, and be available via e-mail and Drexel Learn, as well as before and after class.

GRADING RUBRIC

Successful run of your protocol	40 points
DFA validation	15 points
Well commented source code	10 points
Updated design document	10 points
Requirements document	10 points
Video of your protocol in action	5 points
README	10 points
Extra Credit	10 points
Total	110 points