CS544 -- Computer Networks
Professor Brian Mitchell
Term Project Part #3 – Network Protocol Implementation
See Blackboard for DUE DATE

Introduction

Your final assignment is to implement **your protocol** (probably an application/session layer) by writing a server and a client that uses the stateful protocol you specified in the second part of this project. 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.

Note that I am not expecting a robust (complete) implementation of what you specified in your design; however, what you build should provide enough function so that it could be considered a solid foundation to continue to develop into a complete solution. Please discuss with me how much you plan to implement so that we are on the same page and there are no surprises with grading.

REQUIREMENTS

Language

Both the client and server can be written in (almost) any language, but C/C++, Go, Rust, JAVA, or Python is preferred. If use another language other than those, you must obtain approval from the Professor before starting. Note that preference will be given to using a systems programming language to develop your protocol (C/C++, Go or Rust). Stated differently, I would expect less features to be implemented in your final deliverable if you implement in a systems programming language versus a high-level language such as Java or Python. This increases the realistic nature of this project as real network protocols are not developed in higher level languages such as Java or Python.

Platform

To receive full credit, I must be able to compile and run your application. You need to specify how to compile and run your assignment via providing scripts or preferably a makefile. It must be able to be compiled and run on a Linux-based system such as Tux, MacOS, or the Windows Linux subsystem. You must provide all source code and verify your build process works without any external configuration.

Use of 3rd Party Libraries and Frameworks

You may (and really should) use 3rd Party Libraries and Frameworks in your implementation that help with the processing of your protocol messages and PDUs for both the client and server. You may not however use 3rd Party Libraries and Frameworks for the “kernel” of your protocol. I will provide some sample code illustrating how to work with some QUIC libraries in various programming languages. Thus, all areas that deal with serialization of your messages, PDU processing, and working with the network itself must be your code on top of the socket API. If there is any confusion or question, please feel free to discuss with the Professor.

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 VERY IMPORTANT. In other words, you must implement the DFA you created for your protocol design. NOTE: If your protocol DFA did not earn full credit, you must fix the DFA to earn full credit for its implementation.
- **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). The client should not hard code any of this information, configuration information should be provided via either a configuration file or a set of command line arguments. If using peer-to-peer, you must be able to locate any services that you need.
- **SERVER:** Like the client, any required configuration information should not be hard-coded, it should be provided via a configuration file or via command line parameters. These should be clearly documented in your documentation.
- **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. If there are any questions, please see me first.

Note that you will be required to provide a test suite to demonstrate the protocol functions that you implemented when you submit your implementation. This can be done either using a testing framework for the programming language that you selected, or a bash script that demonstrates the features of the protocol that you implemented.

Extra Credit Options

There will be multiple ways to get extra credit for this part of the assignment. Some ideas are:

- *Implementation Robustness:* I am expecting a workable prototype implementation. A more complete implementation will be awarded proportional extra credit
- *Concurrent or Asynchronous Server:* The minimal server implementation does not need to be concurrent (supporting more than one client at a time). Extra credit will be awarded to a server that is able to handle multiple clients (using a process/thread/coroutine model). Lots of help on how to do this can be found in any programming text, online or you could always discuss with me.
- *Design Excellence:* I am expecting good code quality and appropriate documentation as part of the base deliverable. Deliverables that exhibit high design quality that clearly show some thought towards maintainability and extension will be awarded extra credit.
- *Using a Systems Programming Language:* As I mentioned above, I am expecting a lesser implementation if you use a real systems programming language versus a high-level language such as Java or Python. However, doing slightly more than the minimum using a high-level language will result in extra credit. See implementation robustness above.
- Providing a summary of how focusing on the implementation increased learning and resulted in needing to update your protocol specification you handed in as Part 2 of this

assignment. Software engineering is a feedback-driven process, thus documenting what you learned and how it altered or invalidated some of your design decisions will result in extra credit.

- *Demo Skills:* Providing a 5-minute max video demo with a voice-over demonstrating your running protocol and describing its key features. This video must be made available on a cloud-based service such as YouTube, but the video does not need to be made public, you can either grant access to me, or use an obfuscated URL. The video can be deleted after the term ends.
- Working with a cloud-based git-based system such as GitHub, GitLab, or BitBucket to manage your project deliverables and to submit your final assignment. You can create this repo as a public or private repo. If you make it a private repo, please ensure that we (the Professor and TA have access way in advance of your final submission).
- Have the client program dynamically find the server (e.g. just execute *client* with no parameters or special configuration and the program finds the server listening on that port anywhere in the network).
- Include with your code automated testing code that shows the correctness of your implementation. This is another software engineering best practice. Ideally you would research and implement some degree of fuzzing here.
- Active help of your fellow classmates. I am highly encouraging students to use Discord to collaborate and help each other out with technical challenges during the implementation of your final deliverable. This is how tech companies operate via tools like Slack to drive collaboration. Outstanding collaborators will be rewarded extra credit.

What and How to Hand In

As mentioned earlier, your final deliverable should include all your code and instructions on how to how to compile and build your software on a Linux-based OS via automation. **Please put this information, along with any required configuration and/or command line flag descriptions in a readme.txt or a readme.md file.**

- You may turn in your final project as a single zip file on Blackboard, or submit a URL on blackboard to a cloud based git-system such as GitHub, GitLab or BitBucket. Remember the latter will be rewarded extra credit.
- All source code should be appropriately commented. I am not looking for comments on every line of code, just that all functions/methods are appropriately documented to clearly describe what they do. Any complex algorithms and data structures should also be well documented.
- Any extra credit submission content (from above) should clearly be documented in your readme file. For example, a link to a video demo.

GRADING RUBRIC

Successful run of your protocol	65 points
DFA validation	15 points
Well commented source code	10 points

README	10 points
Extra Credit	Variable
Total	100-125 points