CS594—Project proposal

John O Brickley

February 15, 2024

Unless you suggest otherwise, I have chosen to implement an IRC client and server for my project in this class. Alternative projects I considered include: contributing to the development of the *Cult of the dead cow*'s "Veilid" protocol[1], likely through work on its "VeilidChat" application[2]; developing a "Matrix"–"Zulip" bridge using the framework setup by the developers of the "Beeper" chat application[3]; contributing to the development of one of the "Secure Scuttlebutt" implementations[4], or to its newer, actively developed derivative "tinySSB"[5]; or to developing an extension to HTTP's "last updated" functionality, adding automatic link content hashing and archival. Each of those ideas seems interesting and worthwhile to me, however: I presently have little experience programming in "Rust"; the "Matrix" bridges to other protocols linked by the "Beeper" project average about 10,000 lines of code[6], which places development of such an extension far outside the scope of this class; "Secure Scuttlebutt" in its original form does not appear to be under active development, and I am not yet competent to judge claims about flaws in its protocol; and, finally, developing an extension to the HTTP protocol that makes it easier to automatically distribute the task of archiving linked pages is a project that is, again, outside the scope of a single class.

Since I am learning about sockets and network programming concurrently with this course, I plan to use the code I will be developing over the next week in my operating systems class as the foundation of the IRC client and server. The code developed under that class's rubric will be written in C and will comprise a client, a server, and a secondary client that the server starts as a command runner. Those three programs will communicate to execute a set of commands for querying and modifying a UNIX filesystem. As such, the only overlap between that project and the project for this class will be the client-server architecture with simple interprocess communication—everything else will be rewritten in accordance with the IRC protocol's specification. Upon

[1] https://veilid.com/
[2] https://gitlab.com/veilid/veilidchat
[3] https://github.com/beeper
[4] https://github.com/ssbc
[5] https://github.com/ssbc/tinySSB
[6] When counted using `cloc *.<target language>`.

your request, I will provide you with the assignment specification from the operating systems class so that you can verify that the overlap between the two projects is reasonable. The grading rubric for the IRC project is duplicated below.

Grading sheet (IRC)

Feature	Max	Actual
RFC Document	20	
Server Process	3	
Client can connect to a server	3	
Client can create a room	3	
Client can list all rooms	3	
Client can join a room	3	
Client can leave a room	2	
Client can list members of a room	3	
Multiple clients can connect to a server	5	
Client can send messages to a room	5	
Client can join multiple (selected) rooms	10	
Client can send distinct messages to multiple (selected) rooms	10	
Client can disconnect from a server	5	
Server can disconnect from clients	5	
Server can gracefully handle client crashes	5	
Client can gracefully handle server crashes	5	
Programming Style	10	
Private or Ephemeral Messaging	5	
Secure messaging	5	
File transfer	5	
Cloud connected server	5	
Total	120	