**Assignment 2: Transport Protocols (Kathryn Lepine 30044629)**

Need to have five separate terminals running. Login to each terminal on ssh [username@136.159.5.27](mailto:username@136.159.5.27). Follow the commands to compile and run each file in the order presented.

Terminal 1: “gcc -o voting voting.c” 🡪 “./voting”

Terminal 2: “gcc -o currency currency.c” 🡪 “./currency”

Terminal 3: “gcc -o translator translator.c” 🡪 “./translator”

Terminal 4: “gcc -o indirection indirection.c” 🡪 “./indirection”

Terminal 5: “gcc -o client client.c” 🡪 “./client”

If you would like to connect multiple clients, you will need to run the client.c program on other terminals. You will only interact and type commands into the terminal running client.c. Once client is connected to the indirection server you can use the microservices. You may choose the services “1” translator, “2” currency, “3” voting, “4” quit client, and “0” will take you to the main menu with all the microservices.

Client to indirection is a TCP connection, indirection to the microservices is UDP.

If the UDP connections gets lost when using currency or translator, you will be prompted to re send the message. If a connection gets lost while using voting you will be taken to the main client menu and can choose a microservices.

Voting only allows you to see the results once you have voted. A client can vote as many times as they would like. They key is used for encryption is random int between 0 and 9.

Translator supports five words: hello, goodbye, no, yes, thank you.

Currency supports CAN to BIT, US, EUR, GBP all using predefined exchanged values.

Testing was done on my MacBook Air on my home wifi while shh-ing into computer science server 136.159.5.27 which is csa3. I only tested with all terminals connected to the same csa3, to simulate a broken UDP connection I would quit the microservices with command control-c then try using the service on client.