Group 18: Thy Do, Brian Moore

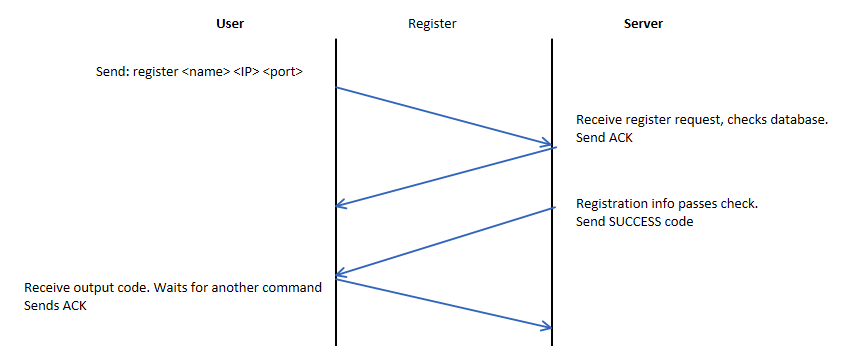
CSE 434

Dr. Syrotiuk

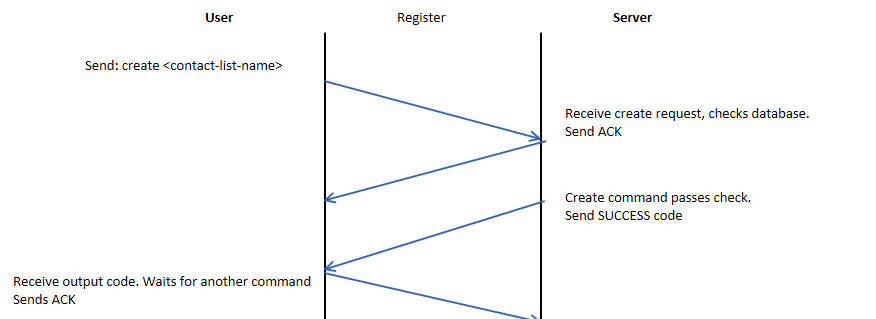
Socket Project Milestone Design Document

**Design**

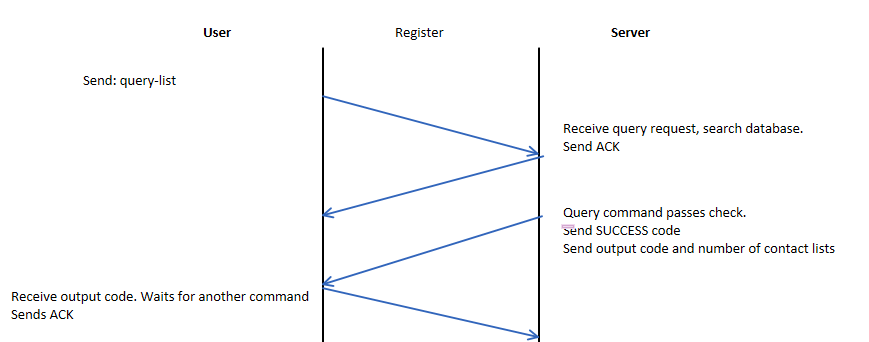
* This socket project is written in Python, currently compatible with python (not python3).
* First, the server is started up, printing its own IP address and waits for a client to send requests. As the client connects to the server, a welcome message is shown along the server IP address, and the user is prompted to enter a command. We also print the client’s IP address for easy registration. The server parses client’s message for the command keyword first, then passes it to the proper function.
* There are 3 global databases: a dictionary for each registered user, a dictionary for all created lists with joined users, and a dictionary for contact list and im-start contact name tuple. Dictionary of registered user is a nested dictionary; each name entry has its own dictionary of IP address and port number. This allows easy implementation, as usernames are keywords to be looked up in the dictionary, we can also query either for the user’s IP address or port number separately. The contact list is a triply nested dictionary. Each entry is the contact list name, they contain entries of joined user, defined similarly to the registered database. We chose this method so data can be looked up using keywords, allowing easy implementation. The databases are accessed using built-in algorithms and methods. The third dictionary is to verify the same person that initiated the im-start command is the same name that sends im-complete. The dictionary’s entries are keyed by the contact-list.
* The server currently supports 20 users and 10 contact lists.
* Clients are sent feedback messages reflecting the outcome of their request. Server end also has internal print messages, showing client’s IP addresses to differentiate sources of requests. Other internal messages are printed to reflect results of different functions to maintain developer’s sanity as we build this program.
* With the implementation of IM functionality, the client program is added a chat socket. When the user registers, the port registered is bound to the chat socket. The chat socket runs in a separate process to accept peer messages and process them.
* How we send P2P message: message, list name, contact list, number of contacts, wrapped in a list that gets encoded. Upon receiving, the list is decoded. The contact list itself is an encoded dictionary. We rotate the list until the current peer is on top and prints it. We rotate once more to extract the address and port of the next host. The counter is decremented for every peer the message is sent to, once the message reaches back to the original sender, it prints SUCCESS.
* For the final submission, the following commands are implemented:
  + Register: The command format is register <contact-name> <IP-address> <port>. The command will result in FAILURE under these following circumstances: insufficient parameter (it will also send back usage instruction), registered name already exists, invalid IP address, invalid port number, or maximum number of registered users already met. A short explanation follows. SUCCESS is sent back if user’s information is added to the database successfully. The client’s port that is entered is then bound to the chat socket to listen to incoming message. This process is running in a separate processing thread.



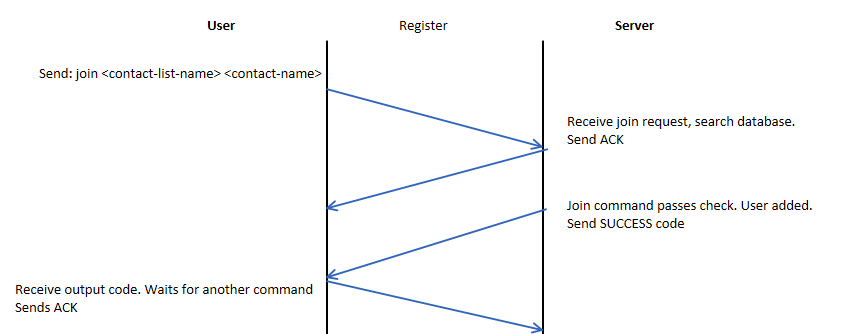
* + Create: Command format is create <contact-list-name>. The command sends back FAILURE if: insufficient parameter entered (sends usage instruction), list name already exists, or exceeded number of existing lists. Sends SUCCESS if list is successfully added to database.



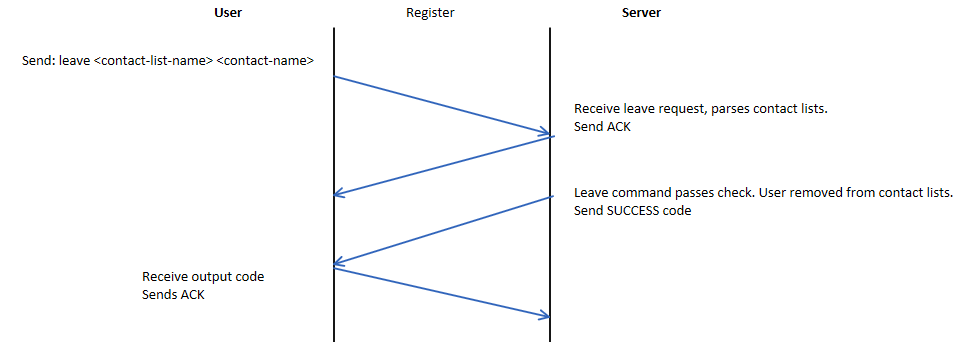
* + Query-lists: Command is query-lists. Sends back number of contact lists along with their names. If no list exits, 0 is sent.



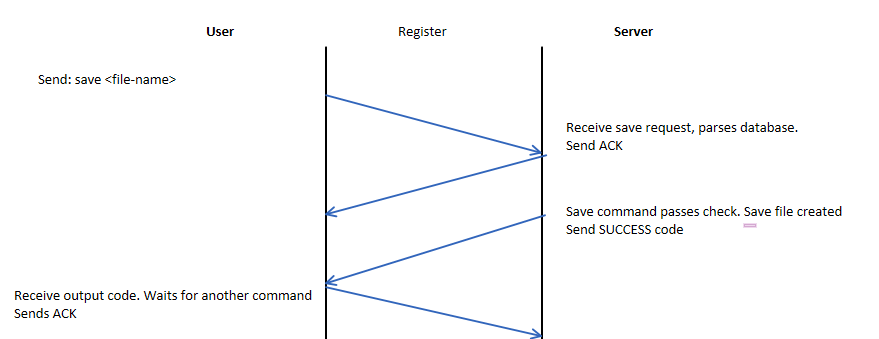
* + Join: Command format is join <contact-list-name> <contact-name>. FAILURE is sent if: contact list does not exist, contact name does not exists in registered database, or contact name already exists in the requested list, or insufficient parameter sent for this command. It will also now fail if there is an IM process is taking place with this list. Otherwise SUCCESS is sent.



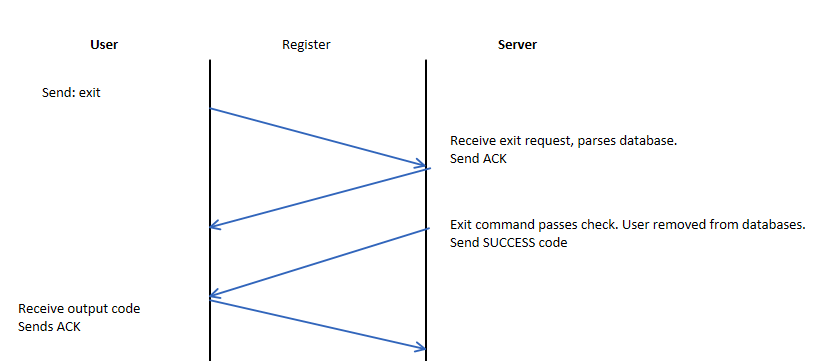
* + Leave: Command format is leave <contact-list-name> <contact-name> . FAILURE is sent if user sends insufficient parameters, IM is in process, or username is not registered, not in the list or the list itself does not exist. SUCCESS is sent otherwise.



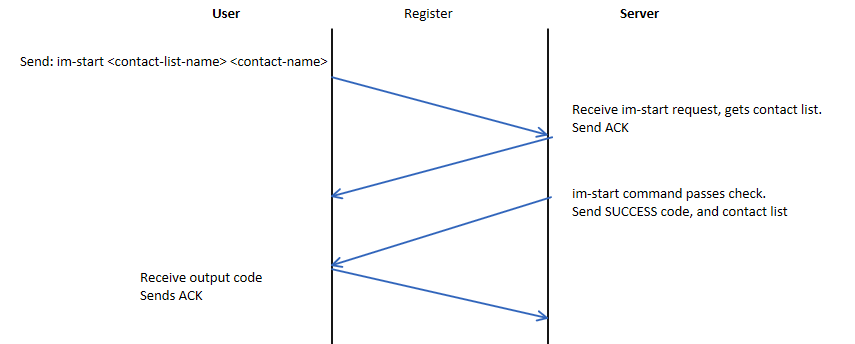
* + Save: Command format is save <file-name> and a text file is save in the format of file-name.txt. FAILURE is sent if command’s parameters are not provided. Otherwise SUCCESS is sent.

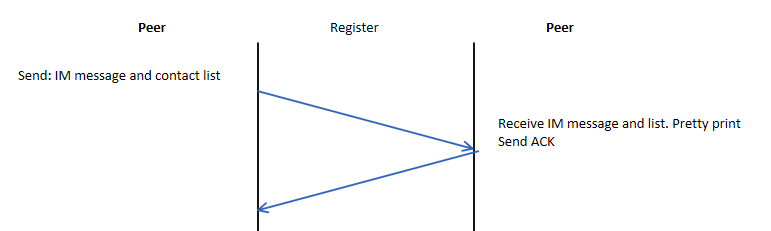


* + Exit: Format of command is exit <contact-name>. Currently, the command will remove the contact-name from the registered database and all joined lists. FAILURE code is sent if the user doesn’t provide sufficient parameters, or name does not exist in registered database or contact lists. SUCCESS is sent otherwise. In addition, it will fail if the user is a part of the IM process.

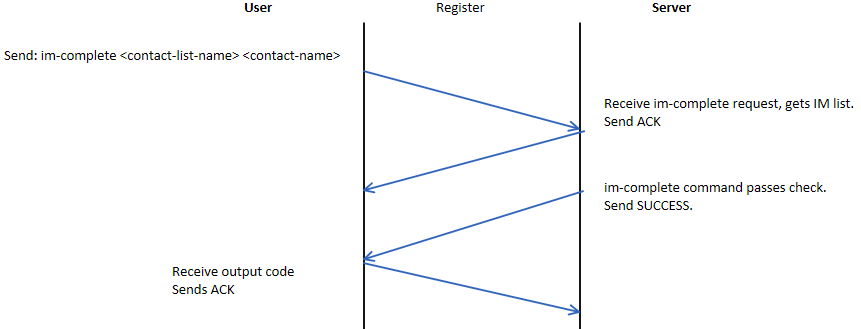


* + im-start: Format of command is im-start <contact-list-name> <contact-name>. Server checks if command parameters are met, sends FAILURE if not met. Also checks if contact list exists, and not currently under an IM process. Also, if contact name is not a part of the contact list, send FAILURE. Otherwise, server sends the client the contact list, and saves the contact list/ client name tuple in a list for verification later. Client program then prints the current client information on top, including name, IP address and port number. Then prompts that client for a message to send to their peers. The message and list will then be sent and printed through the contact circle, each time with the current peer information on top. As all peers receive the message, SUCCESS is printed to the initiator.

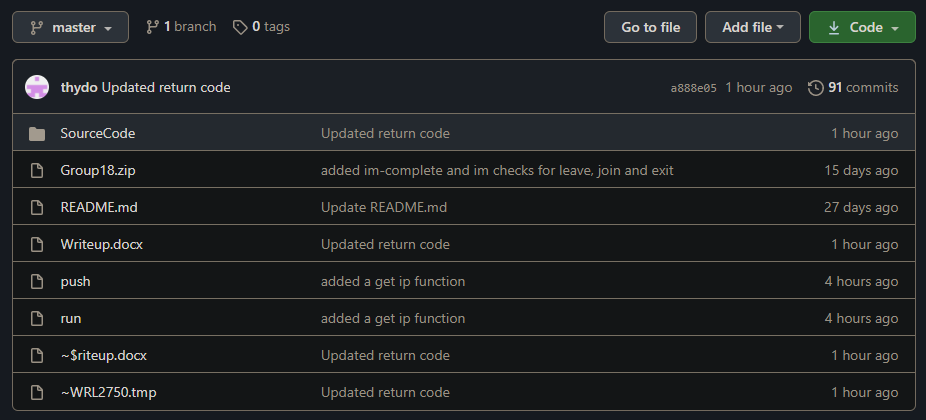


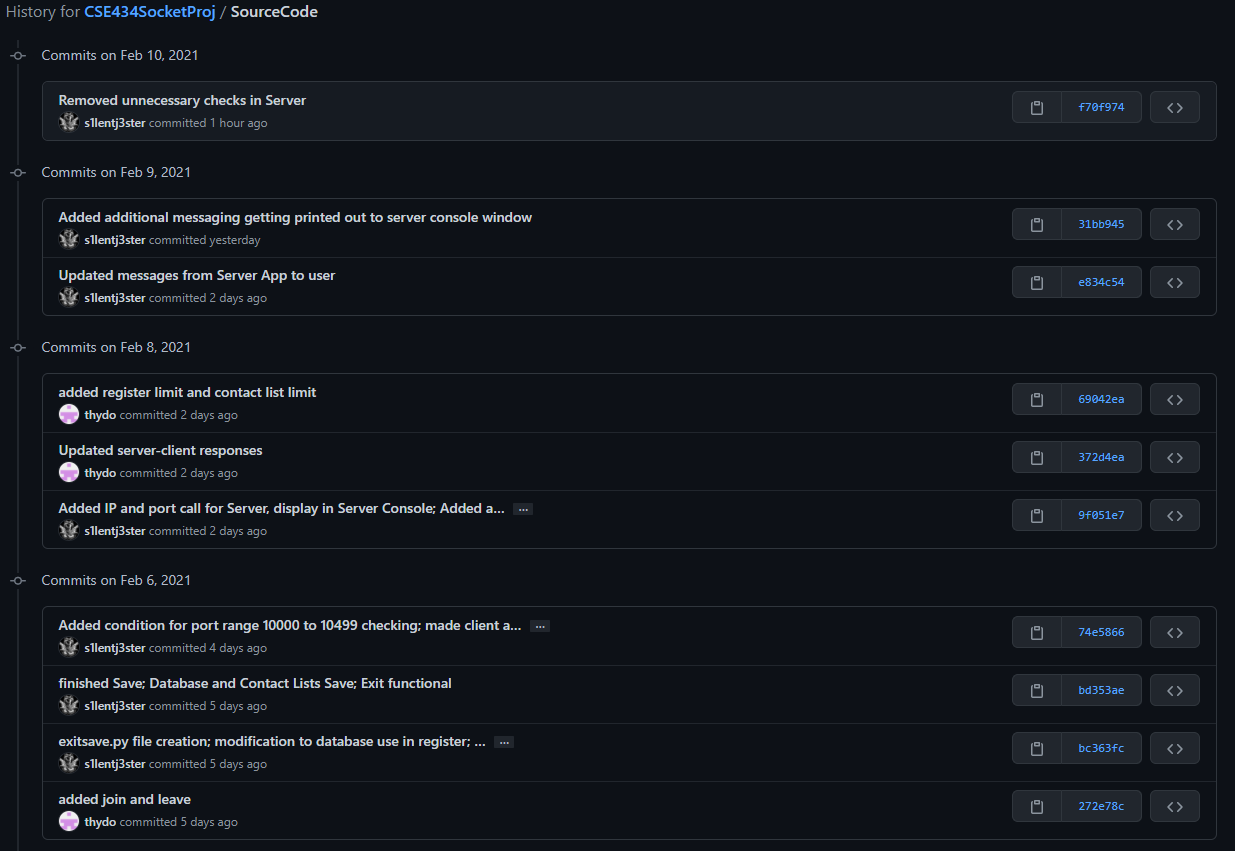


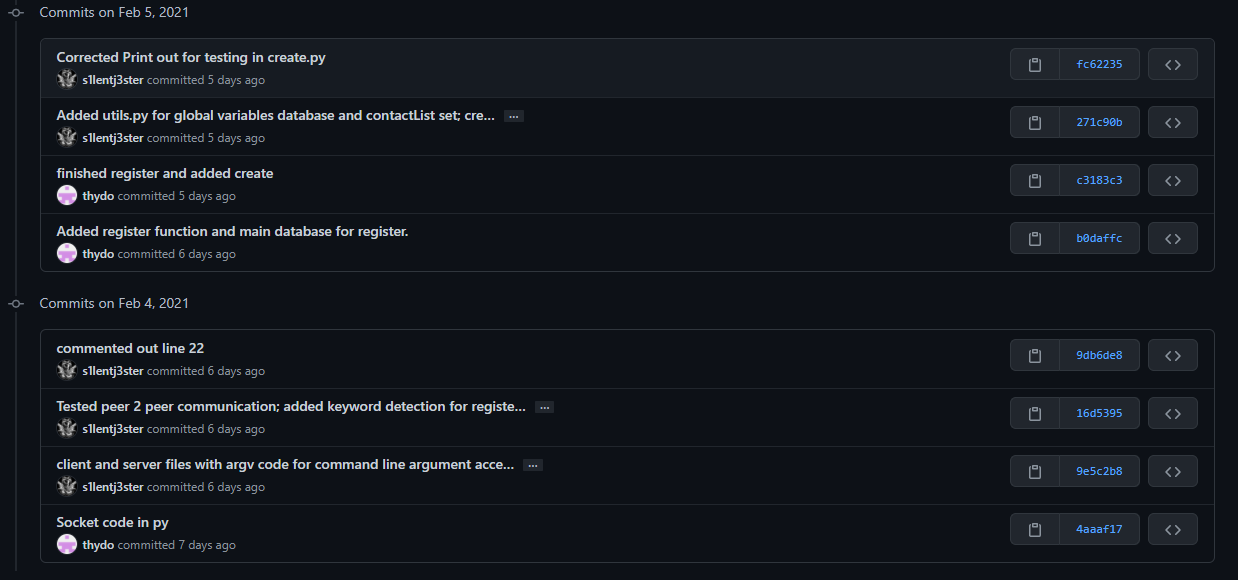
* + im-complete: Format of command is im-complete <contact-list-name> <contact-name>. The user that initiated im-start sends im-complete to the server. Server checks if it’s the correct contact name / contact list tuple. Sends FAILURE if the list or contact name doesn’t exist, or contact name is not in the contact list, or it’s not the contact name that started the IM process, or command parameters are not met. Sends SUCCESS otherwise.

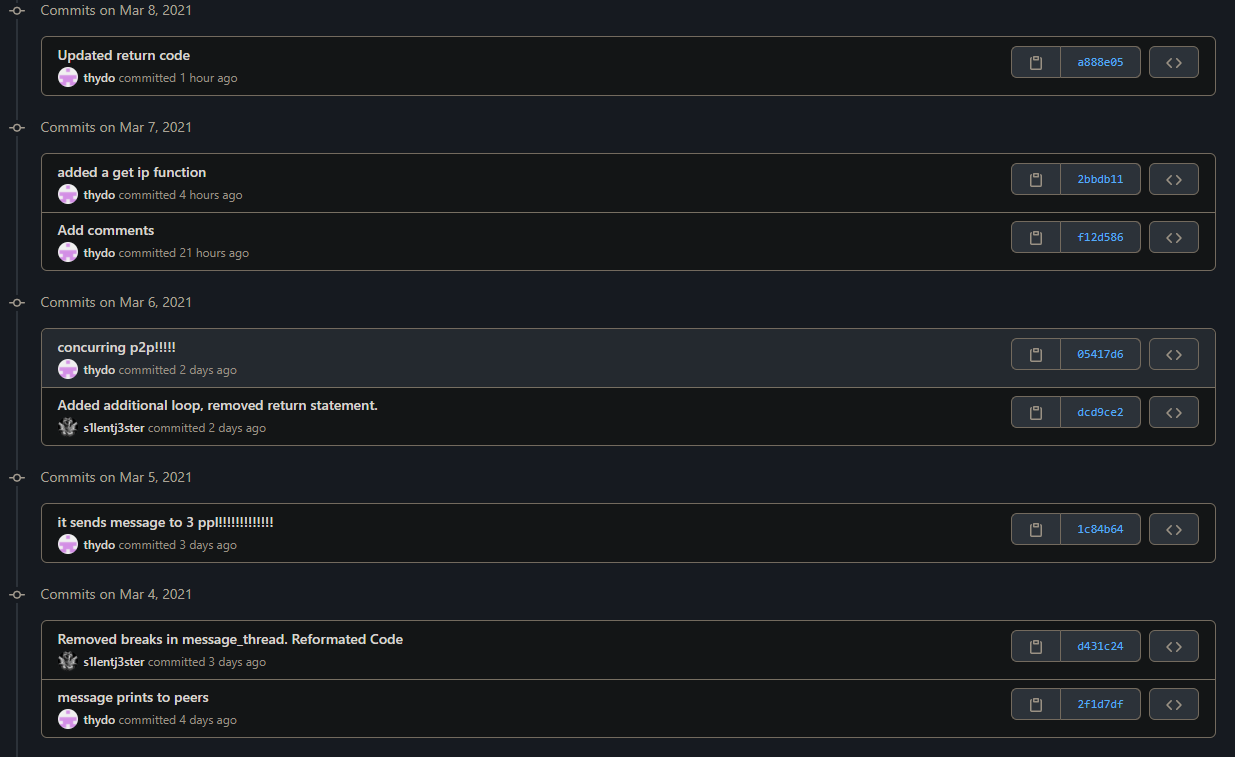
****

**Git history**

****







**Link to demo video:** <https://youtu.be/W7vGow48-sI>