DHT Application Design:

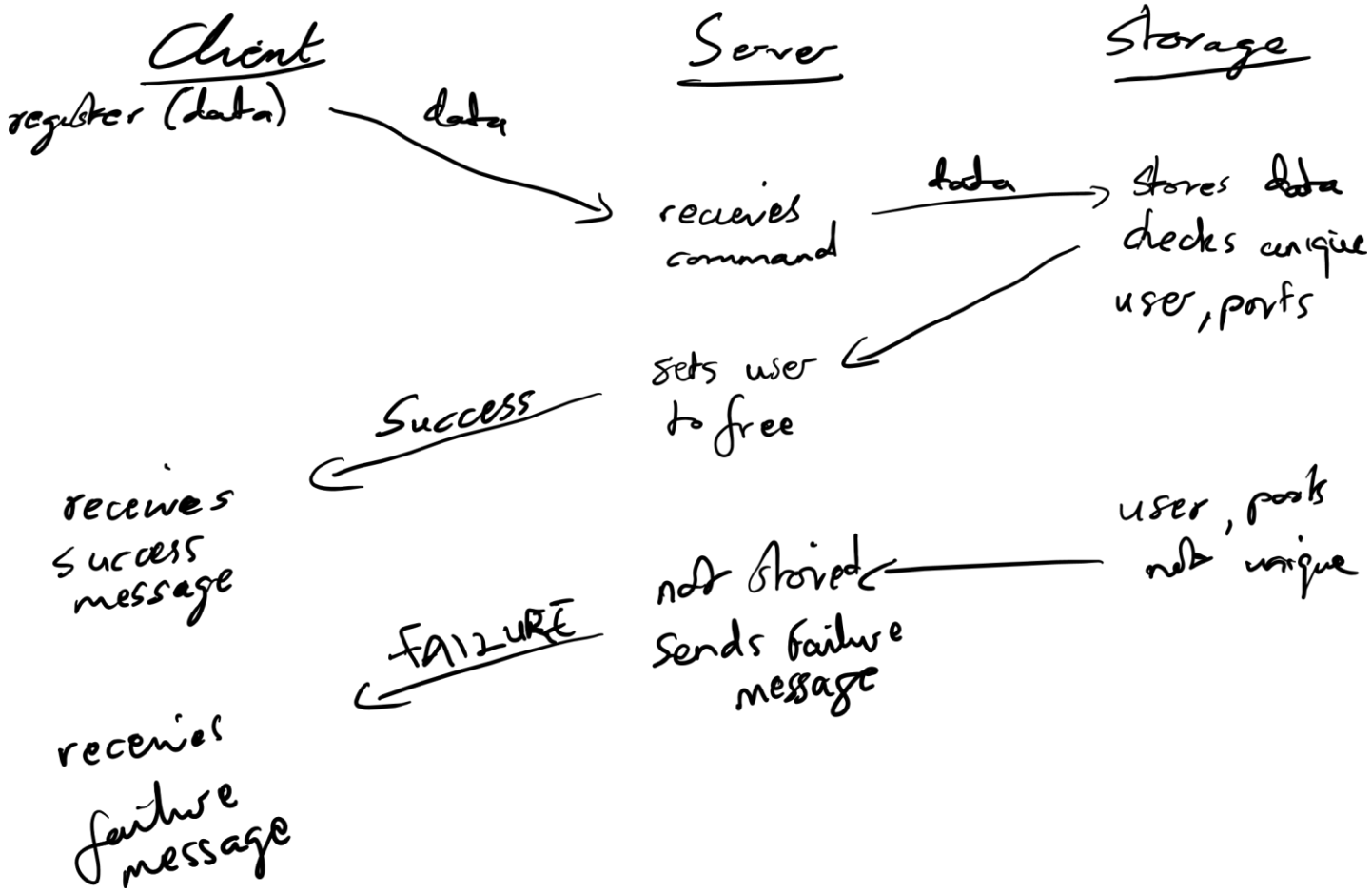
The server is in an infinite listening mode, awaiting commands from the client. When the server receives a command from the client, it decodes the command and maps it to a specific function using the controller function i.e. the controller function acts like a switch. After the execution of the command the server sends a message of completion back to the client, which elaborates whether the command was successful or unsuccessful. The program uses Json to send tuples across communicating devices (peers/clients/server).

Data Structures and Algorithms:

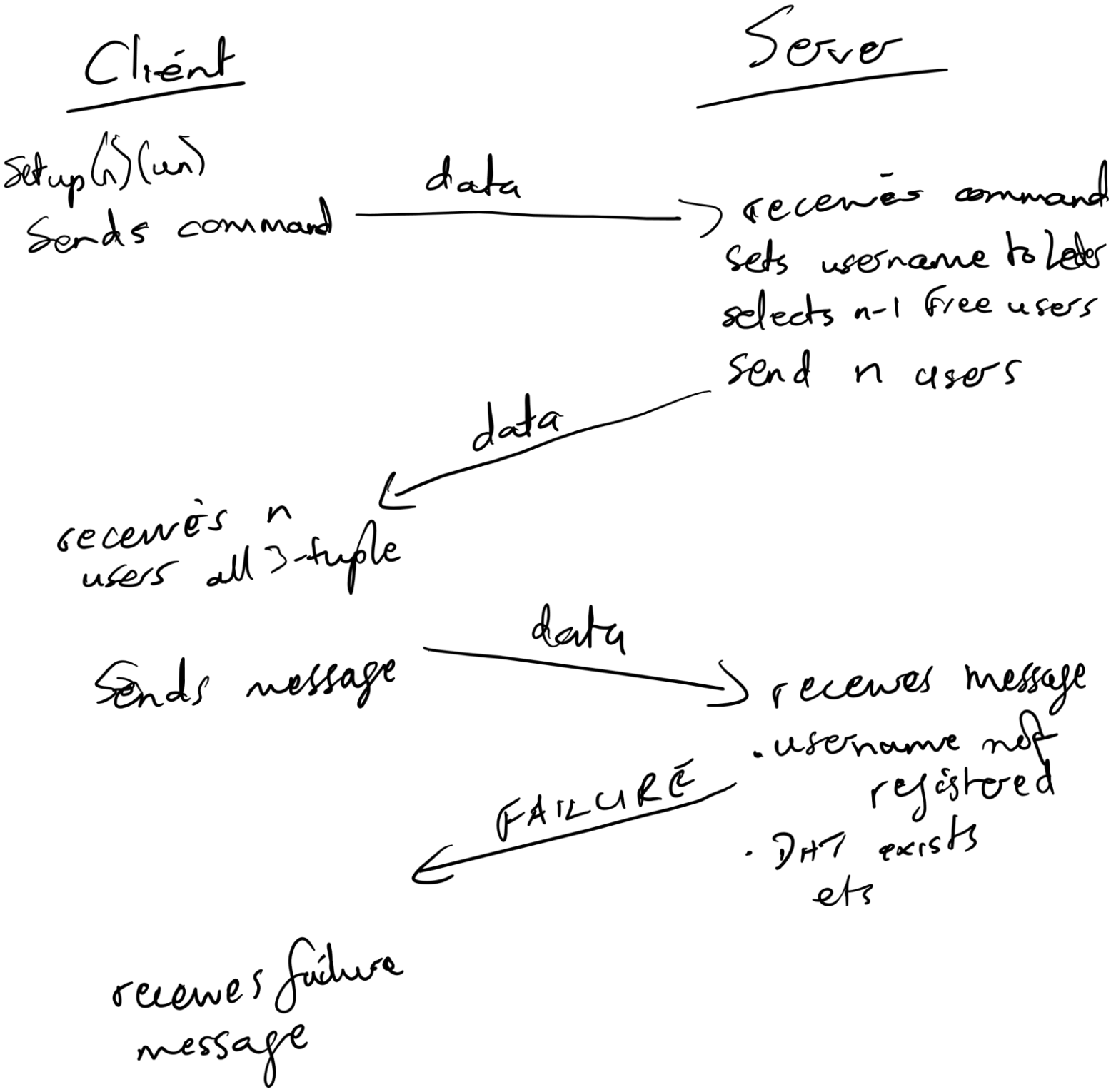
This project uses a Distributed Hash Table, which is a unique instance of a hash table that is distributed among participating nodes in a network. The reason for the use of the DHT is that nodes can be added and removed from the network with minimal disruption to the overall functionality of the network. The data, being distributed to different nodes allows lookup of information to be fast and readily available if a key is given or known for the data to be queried. The runtime complexity of the DHT is very small, and hence the overall overhead is reduced. In a system like this, a DHT provides a high level of fault tolerance, especially given that this implementation is over UDP.

The algorithm basically ensures that each node can be used to query or provide the required data and can access the server, and it’s 2 adjacent nodes.

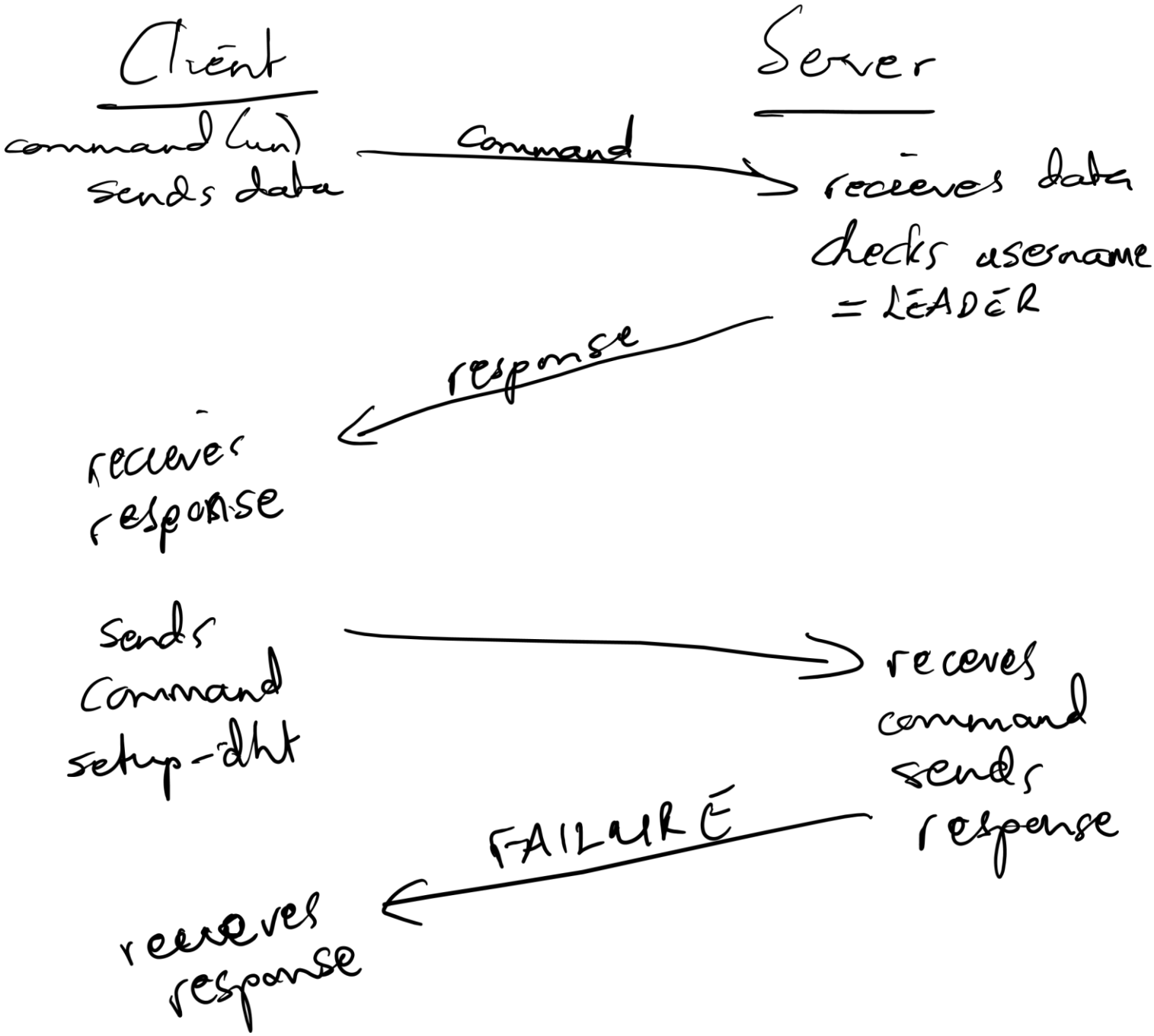
register
Format: <username> <IPv4-address> <port>



setup-dht
Format: <n> <username>



dht-complete
Format: <username>



query-dht

Format: <username>