ARIZONA STATE UNIVERSITY

**CSE 434, SLN 70569 – Computer Networks – Fall 2021**

Instructor: Dr. Violet R. Syrotiuk

**Full Socket Project**

Austin Spencer

10/16/2021

1. **Message Format**

register (successful) - {

'res': SUCCESS,

'type': register,

'data': "{username} added to state table successfully"

}

register (unsuccessful) - {

'res': FAILURE,

'type': register-error,

'data': "{reason}"

}

deregister (successful) - {

'res': SUCCESS,

'type': deregister,

'data': None

}

deregister (unsuccessful) - {

'res': FAILURE,

'type': error,

'data': {reason}

}

setup-dht (successful) - {

'res': SUCCESS,

'type': DHT,

'data': [(‘username’, ‘ip’, ‘port’, ‘port’, ‘port’),(…)]

}

setup-dht (unsuccessful) - {

'res': FAILURE,

'type': setup-dht,

'data': “{reason}”

}

dht-complete (successful) - {

'res': SUCCESS,

'type': dht-setup,

'data': None

}

dht-complete (unsuccessful) - {

'res': FAILURE,

'type': dht-setup-error,

'data': {reason}

}

query-dht (successful) - {

'res': SUCCESS,

'type': query-response,

'data': (‘username’, ‘ip’, ‘port’, ‘port’, ‘port’)

}

query-dht (unsuccessful) - {

'res': FAILURE,

'type': query-error,

'data': {reason}

}

leave-dht (successful) - {

'res': SUCCESS,

'type': query-response,

'data': “Removing {username} from DHT”

}

leave-dht (unsuccessful) - {

'res': FAILURE,

'type': leave-error,

'data': {reason}

}

join-dht (successful) - {

"res": "SUCCESS",

"type": "join-response",

"data": {

"username": "randy",

"leader": [

[

"leader ip",

Left port

],

[

"leader ip",

Right port

]

]

}

}

}

join-dht (unsuccessful) - {

'res': FAILURE,

'type': join-error,

'data': {reason}

}

dht-rebuilt (successful) - {

"res": "SUCCESS",

"type": "rebuilt-response",

"data": "DHT has been successfully rebuilt"

}

dht-rebuilt (unsuccessful) - {

'res': FAILURE,

'type': rebuilt-error,

'data': {reason}

}

teardown-dht (successful) - {

"res": "SUCCESS",

"type": "join-response",

"data": “Initiating teardown of the DHT”

}

teardown-dht (unsuccessful) - {

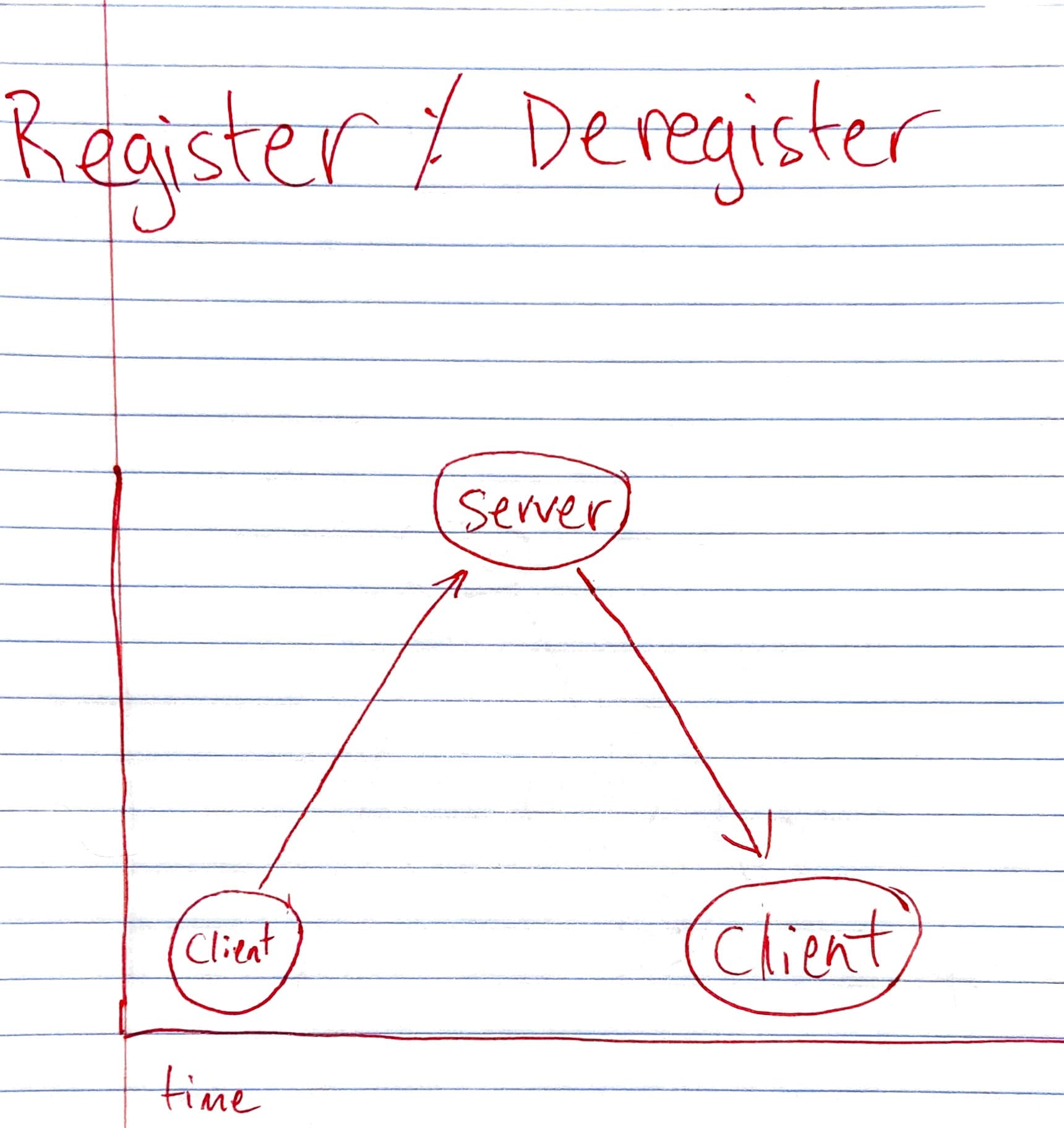
'res': FAILURE,

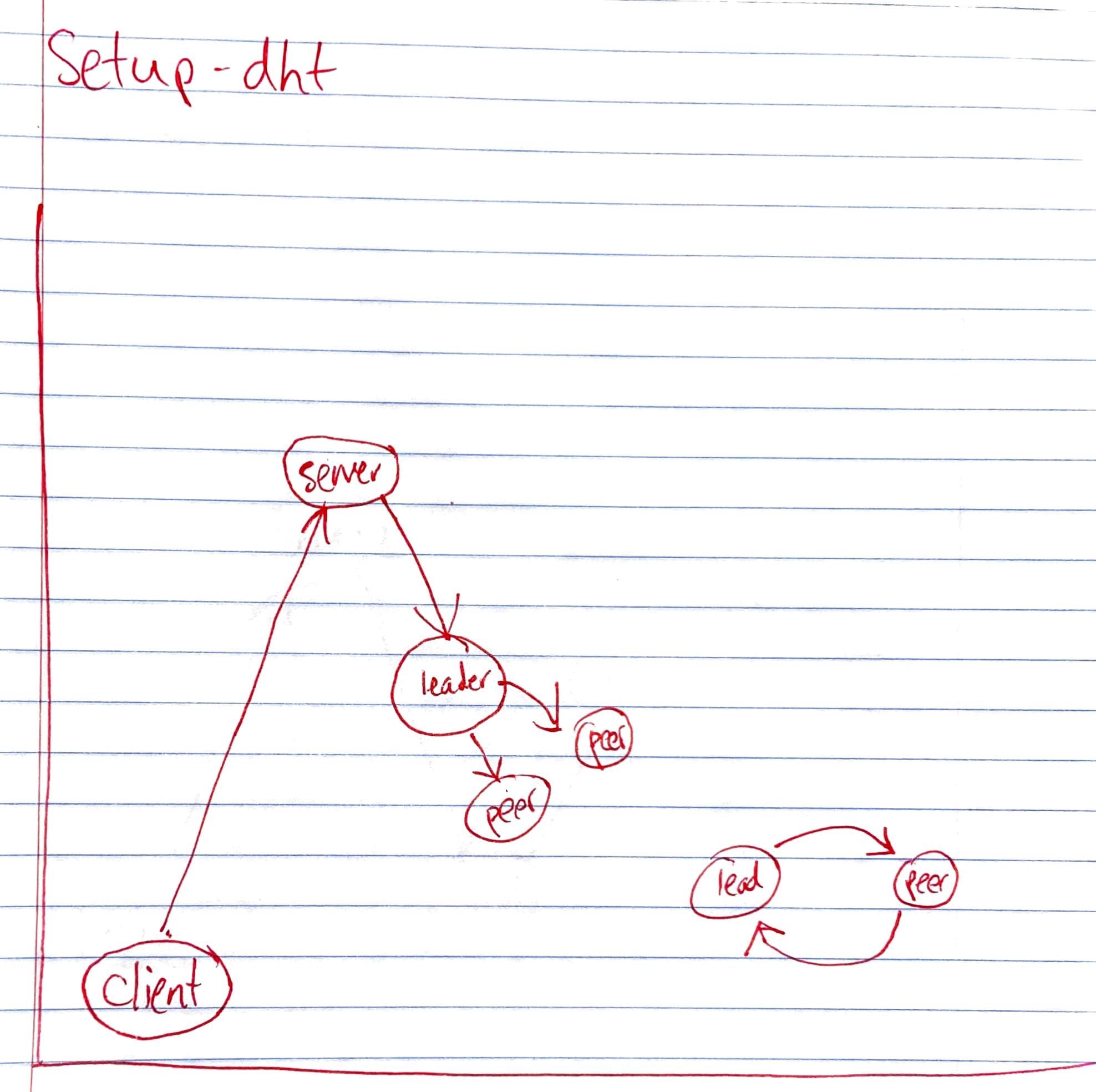
'type': teardown-error,

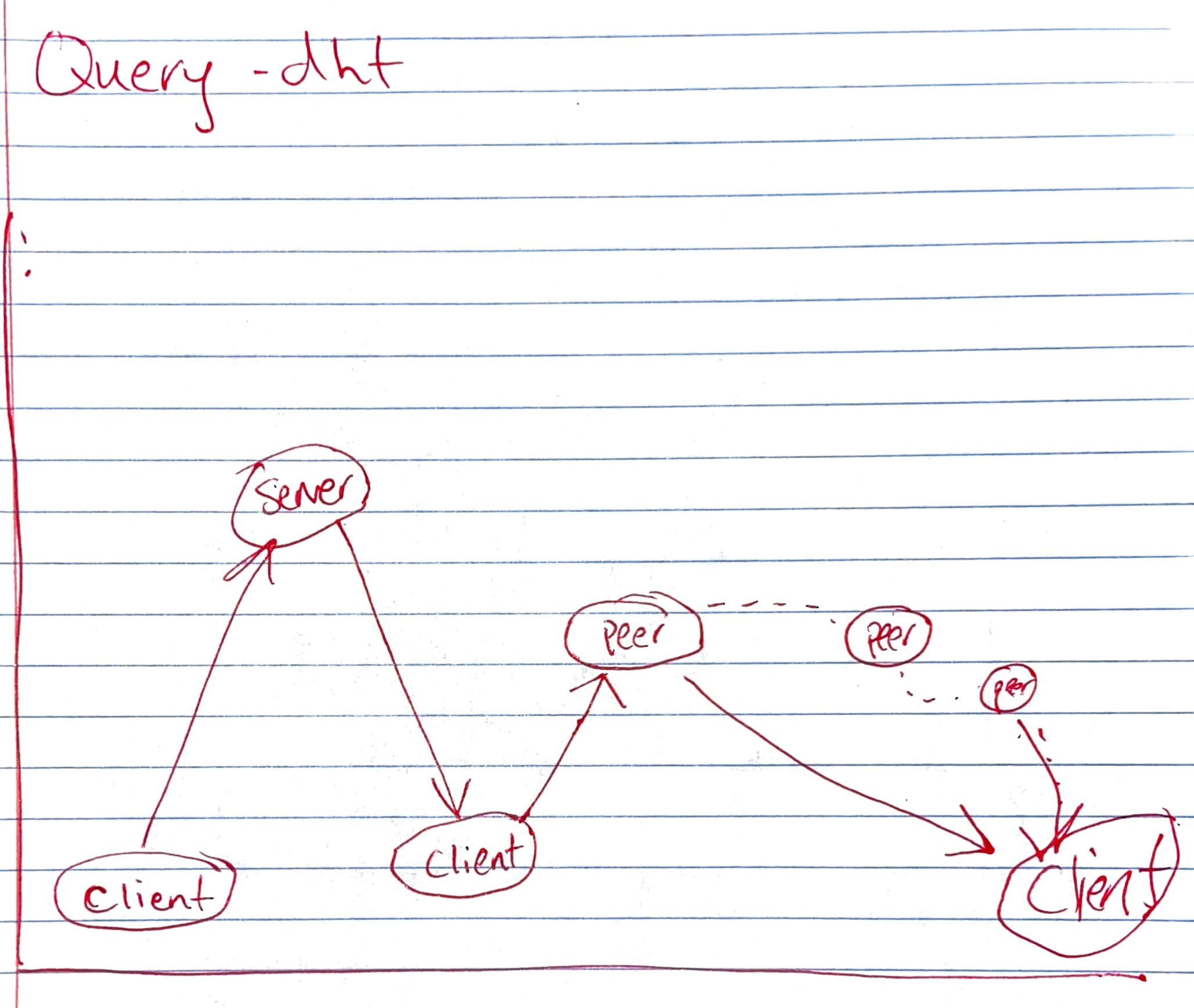
'data': {reason}

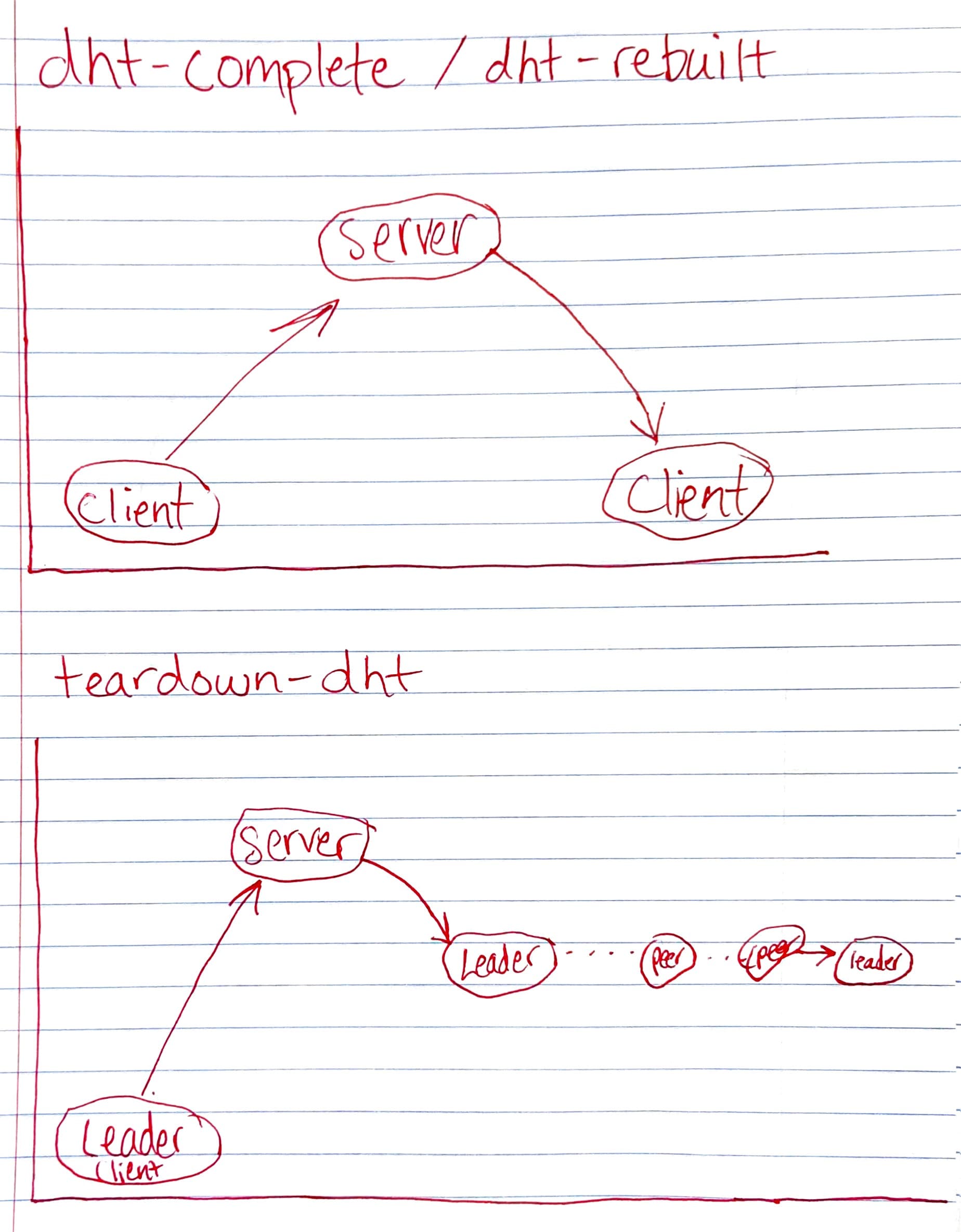
}

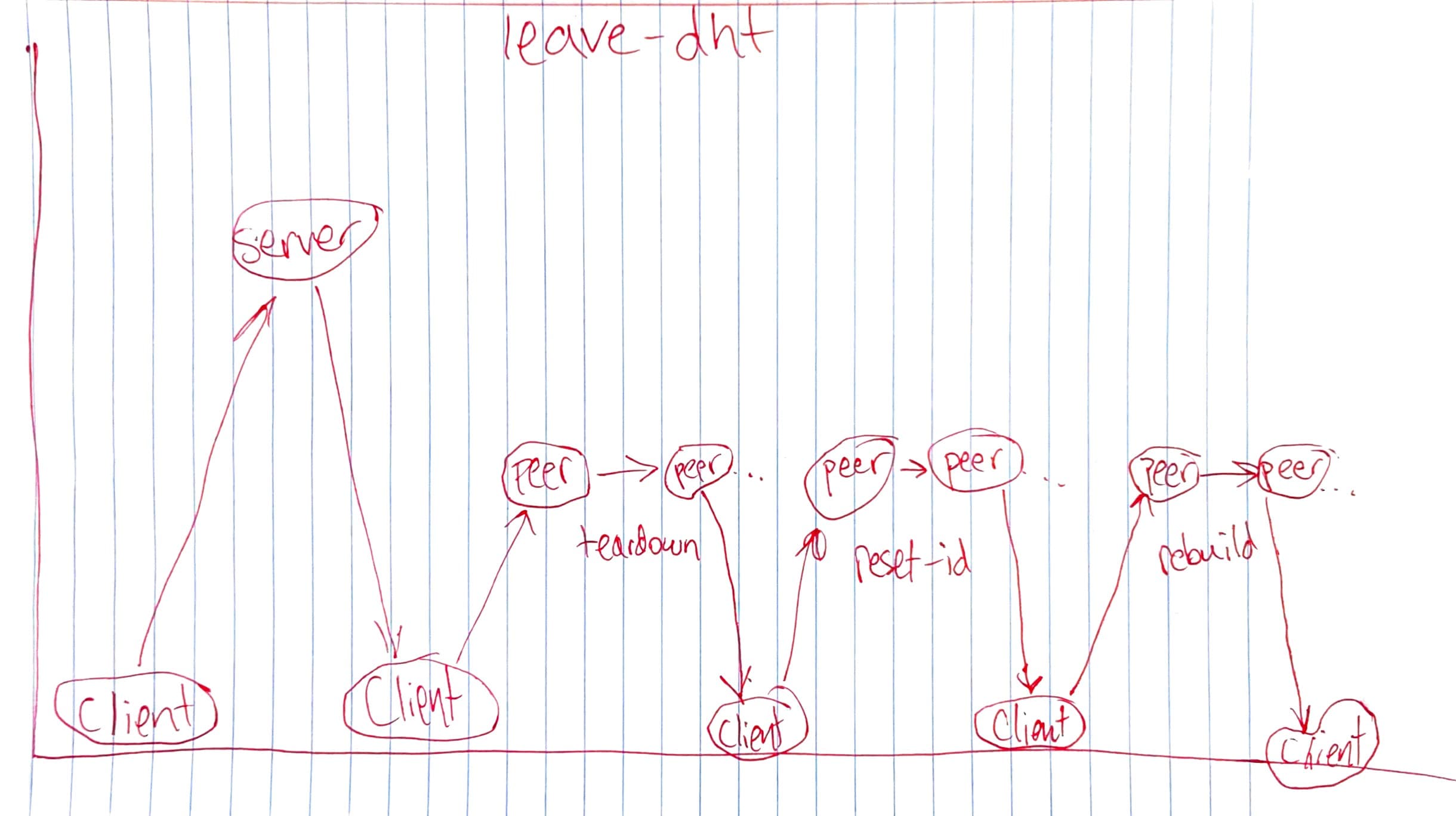
1. **Time-space diagrams**

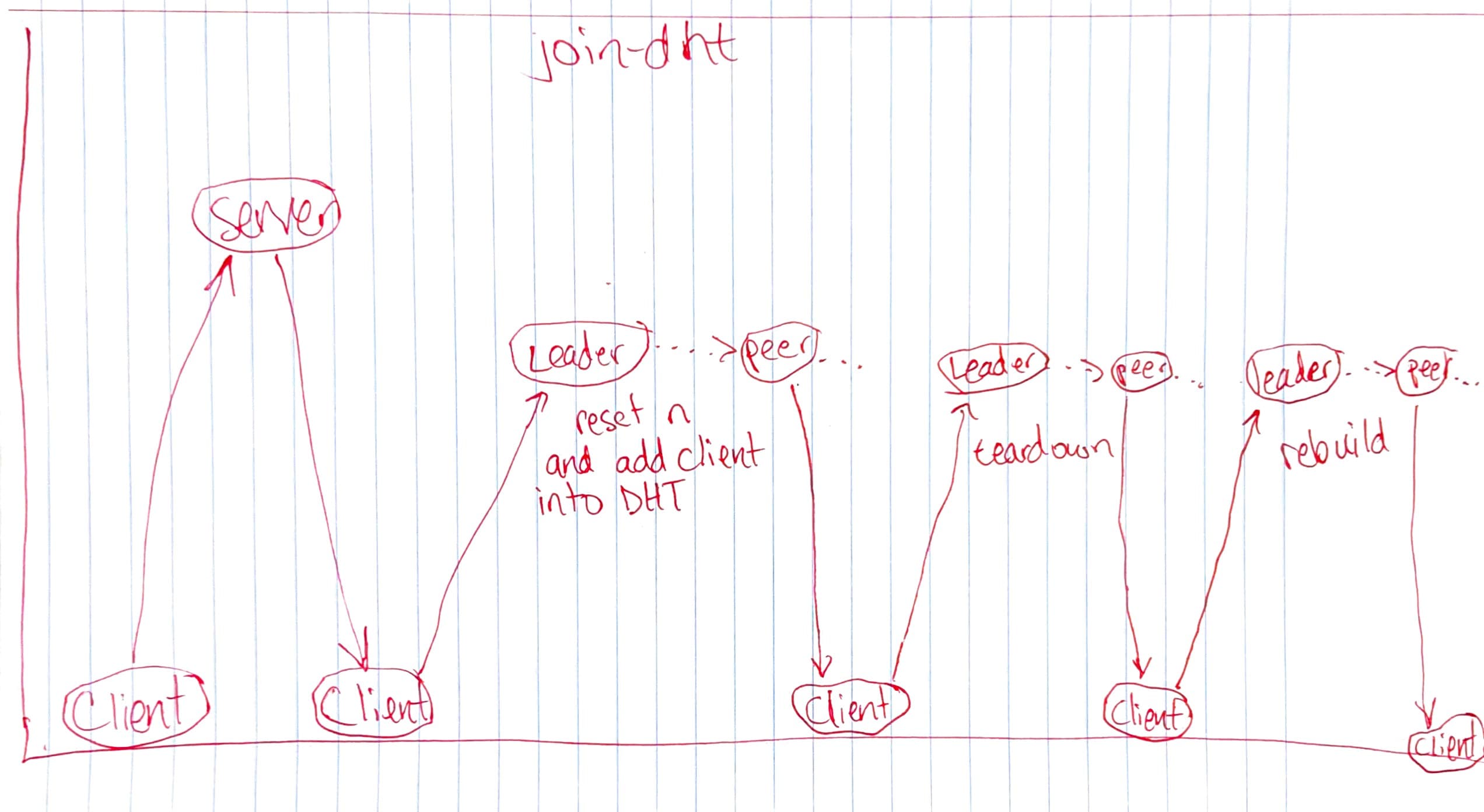












For the join-dht process I made it very similar to the leave-dht process. One thing that I did to make the process simplified is that when a registered client joins the dht they join as the last user in the dht and connect to the leader. This allows me to not need to reset the id’s of each node, rather I can just make the new node’s id the length of the dht – 1. After that, I call teardown and rebuild the same way that is called in leave-dht.

1. **Data structures and algorithms used**

Server side:

1. User object – Python Class object

I am going to try to use more classes as I go on this project. For the milestone I was more focused on functionality but hope to be able to get the code prettier for the final due date.

1. Users – Dictionary

Python dictionaries are very handy for key – value pairs

1. DHT – List of dictionaries

I prefer dictionaries to any other thing besides Class objects in Python. If it were easier to send class objects in the packets I would probably use those over dictionaries.

1. Three\_tuples – tuple of dictionaries

Again – I prefer using Python dictionaries over lists. To me it makes it much easier to visualize the data I am using and easier to debug.

Client side:

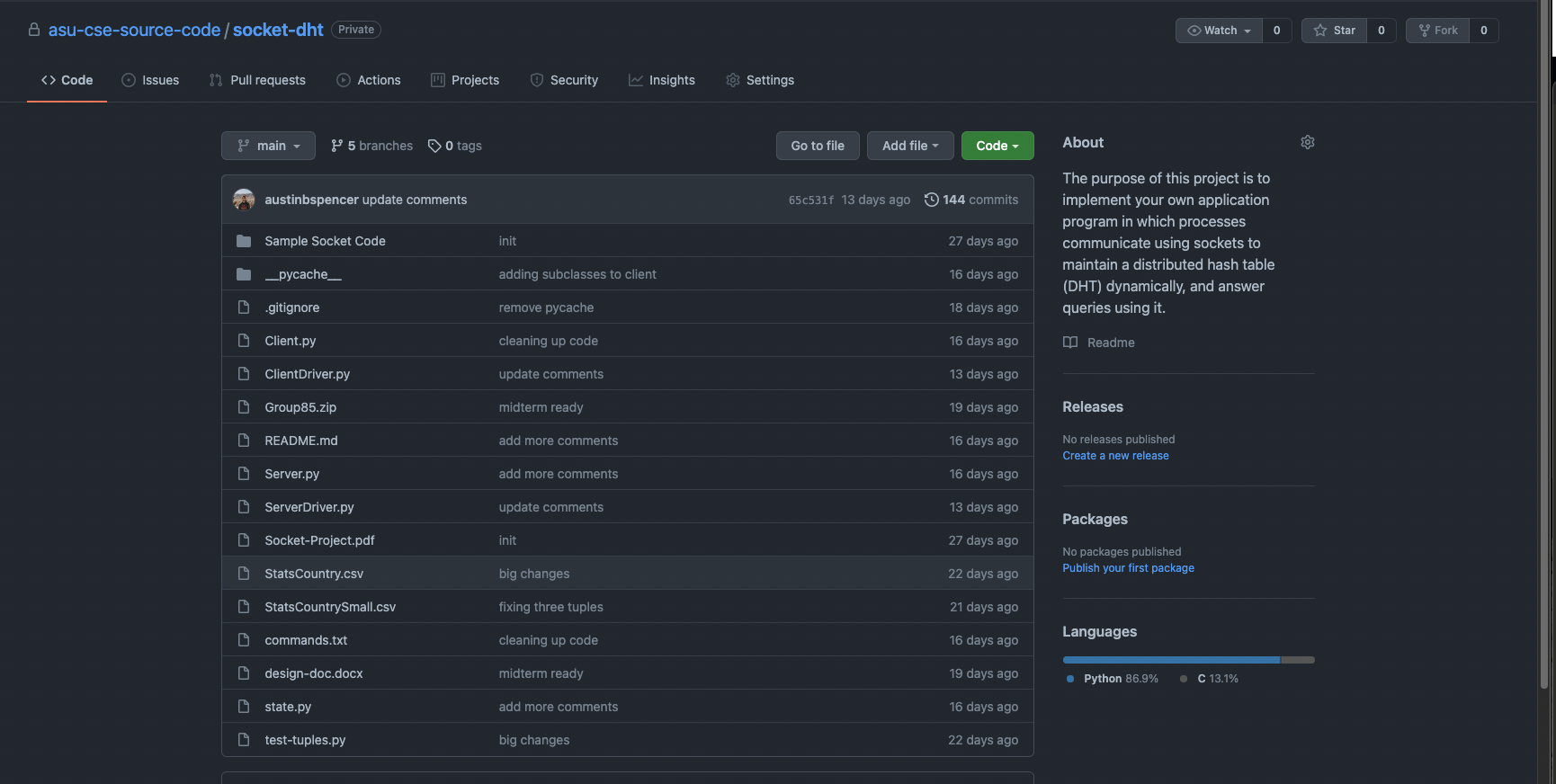
1. Client object – Python Class object

This is a big object that holds everything the client needs to persist. Tried to make the client code simpler by doing this.

1. Local hash table – List of lists of dictionaries

This is where the records are stored on each peer in the client program. It is a list of lists of dictionaries. I think the dictionary is key so that you don’t lose what each value means in the record. To handle collisions in the hash table I used chaining since each index of the hash table is a list.

1. Join-dht command: For the join-dht process I made it very similar to the leave-dht process. One thing that I did to make the process simplified is that when a registered client joins the dht they join as the last user in the dht and connect to the leader. This allows me to not need to reset the id’s of each node, rather I can just make the new node’s id the length of the dht – 1.
2. **Snapshot showing commits in GitHub**



1. [Video link](https://youtu.be/TQGvZwVt50g)

https://youtu.be/TQGvZwVt50g