ARIZONA STATE UNIVERSITY

CSE 434, SLN 70569 – Computer Networks – Fall 2021

Instructor: Dr. Violet R. Syrotiuk

Full Socket Project

Austin Spencer

10/16/2021

a. 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"
  }
```

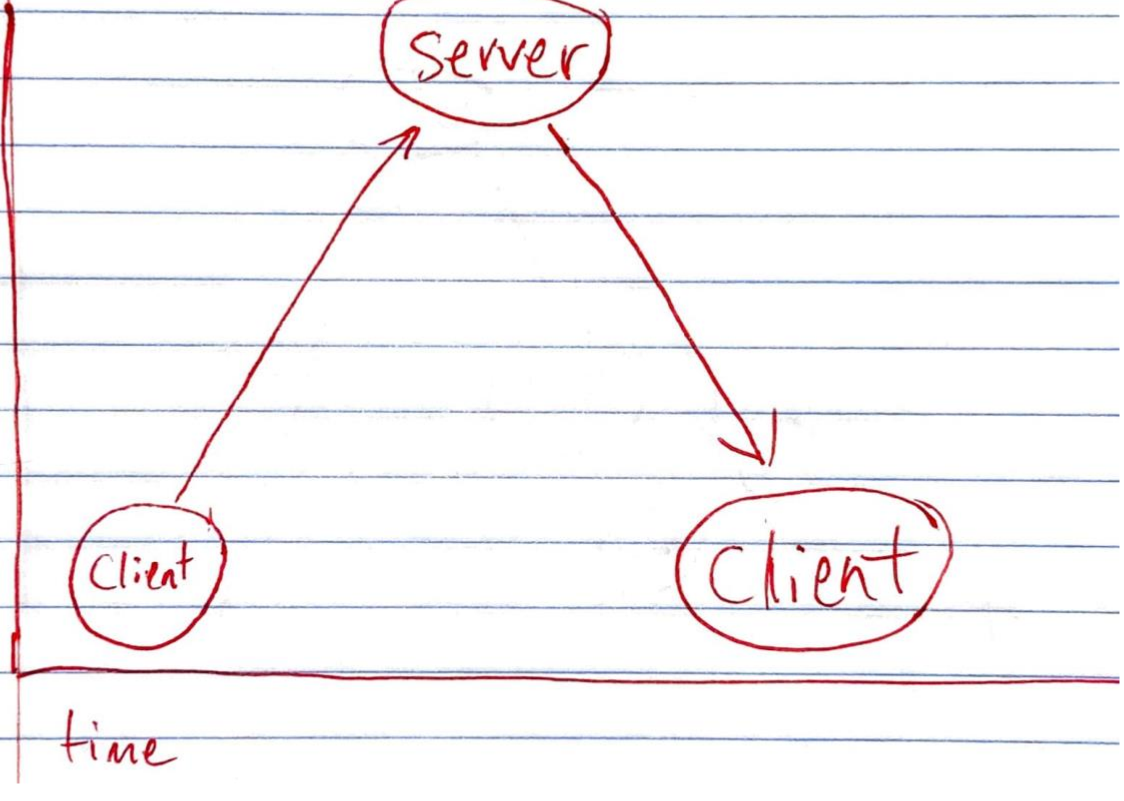
dht-rebuilt (unsuccessful) - {

```
    'res': FAILURE,
    'type': rebuilt-error,

    'data': {reason}

  }
```

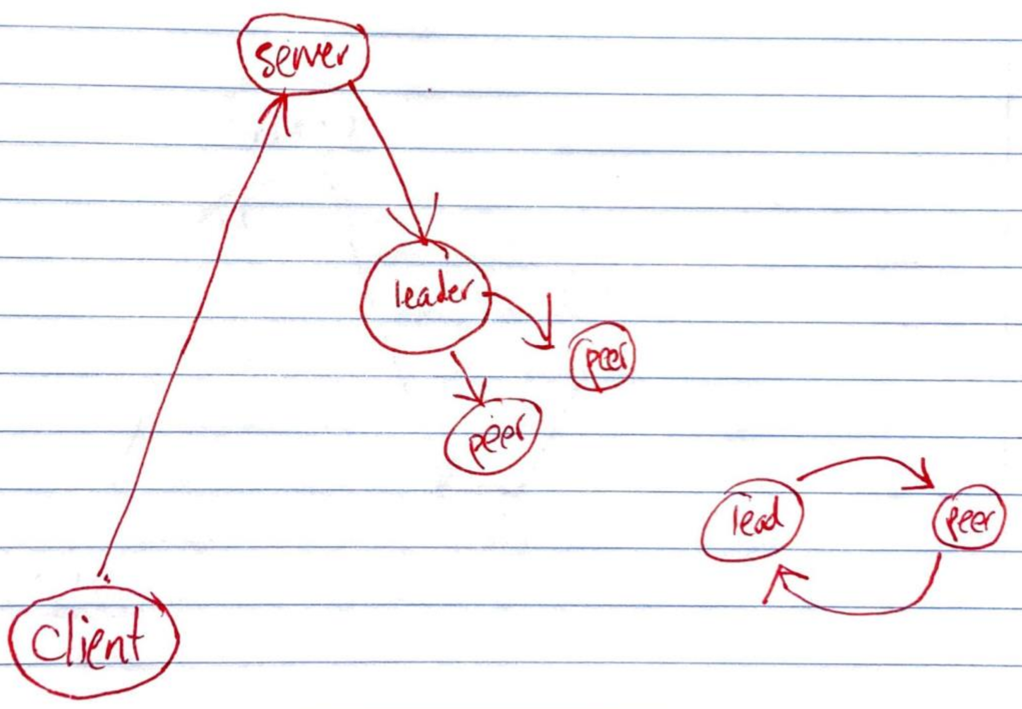
```
teardown-dht (successful) - {

    "res": "SUCCESS",

        "type": "join-response",

        "data": “Initiating teardown of the DHT”

  }
teardown-dht (unsuccessful) - {

    'res': FAILURE,

    'type': teardown-error,

    'data': {reason}

  }
```

b.Time-space diagrams

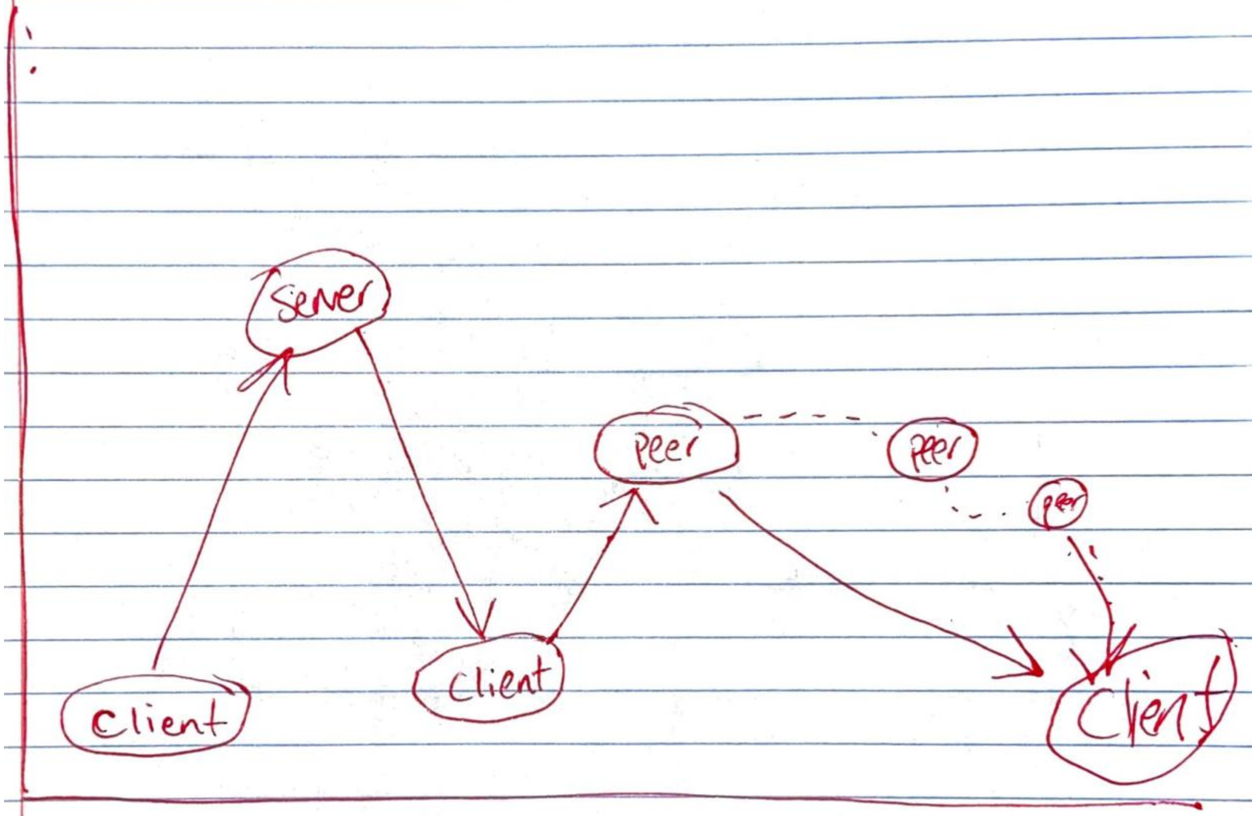
Register / Deregister



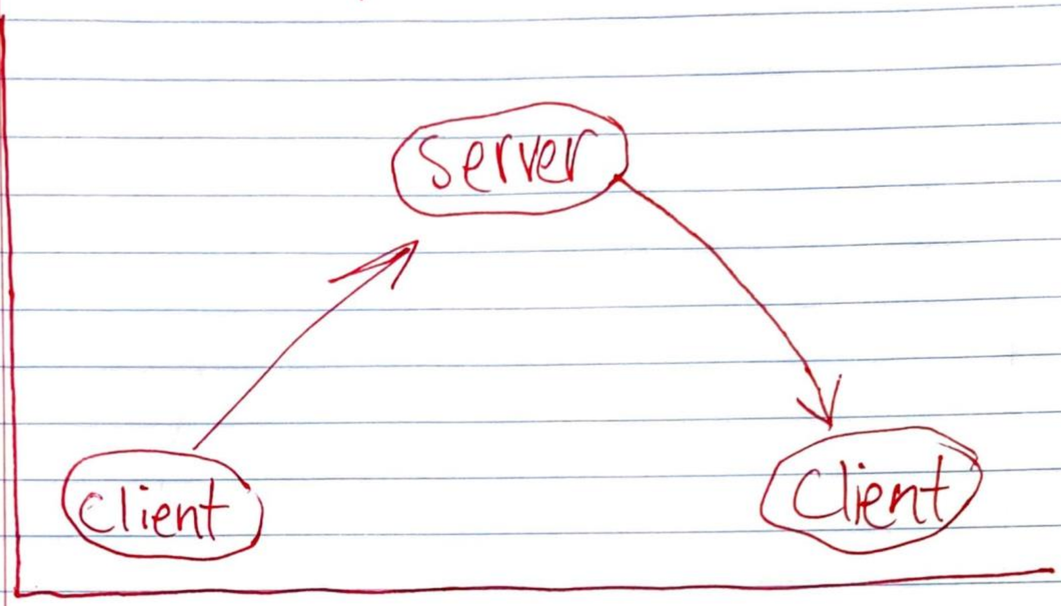
Setup-dht



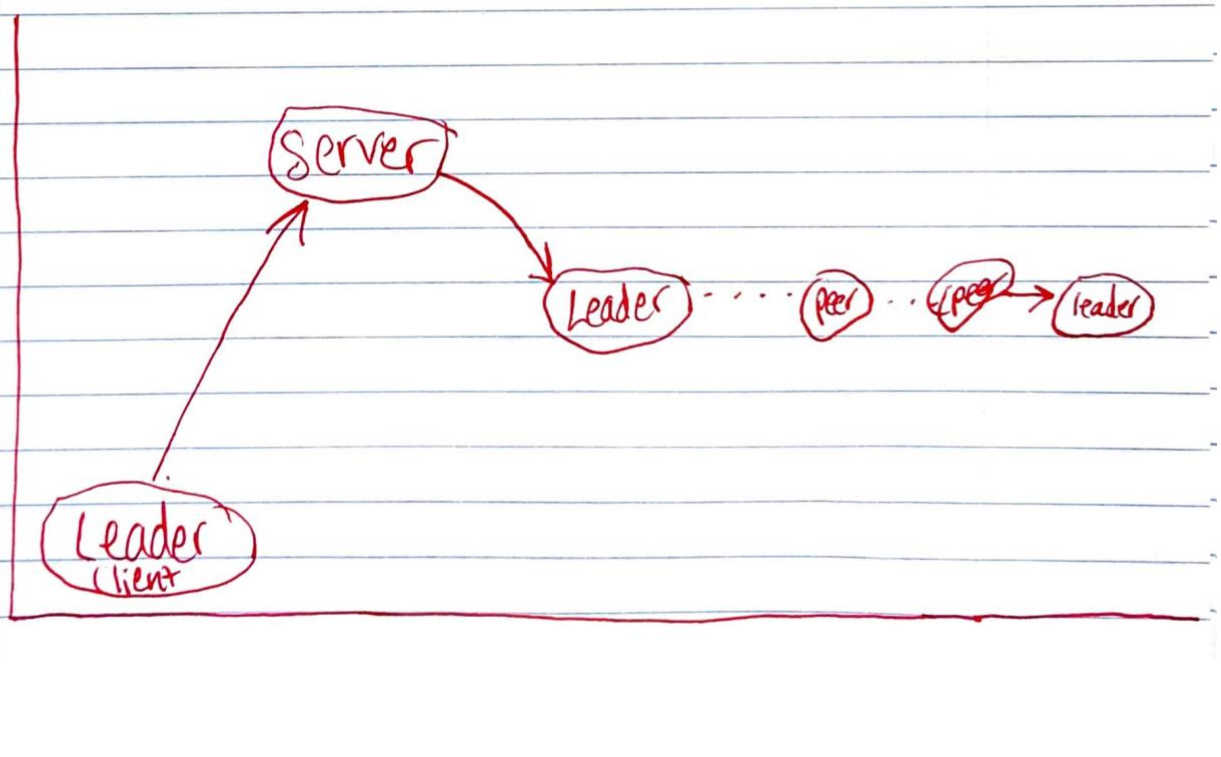
Query -dht

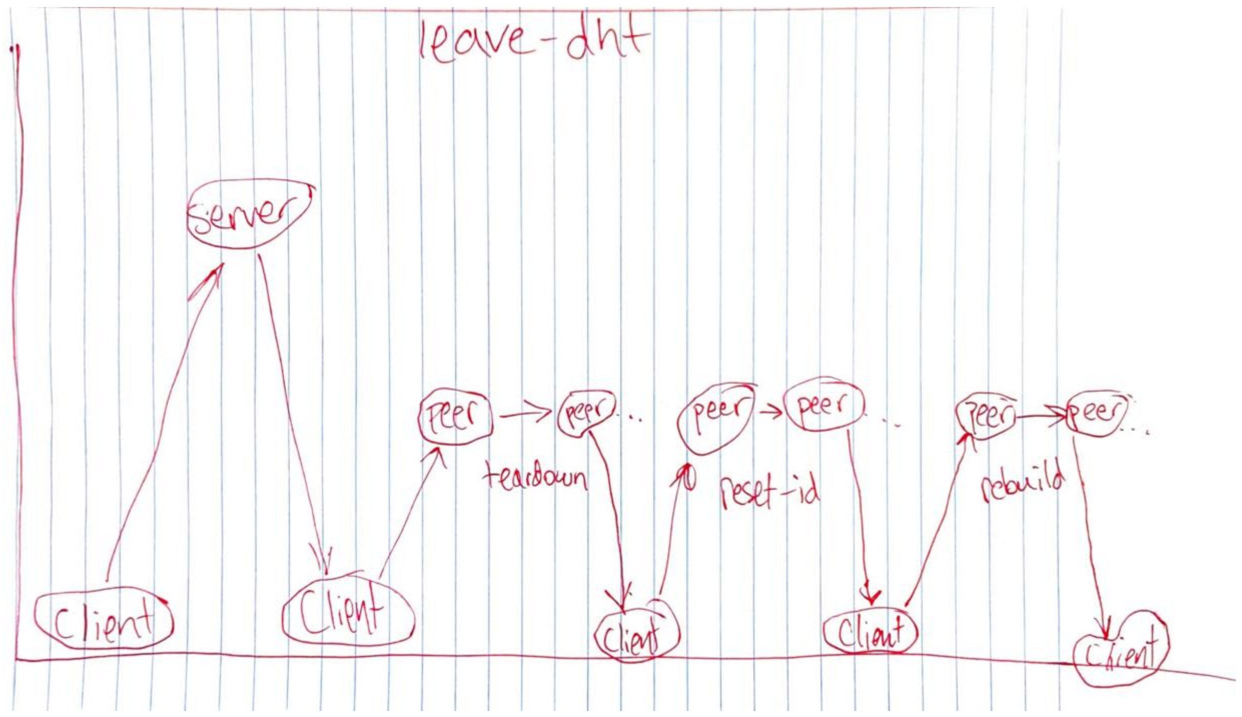


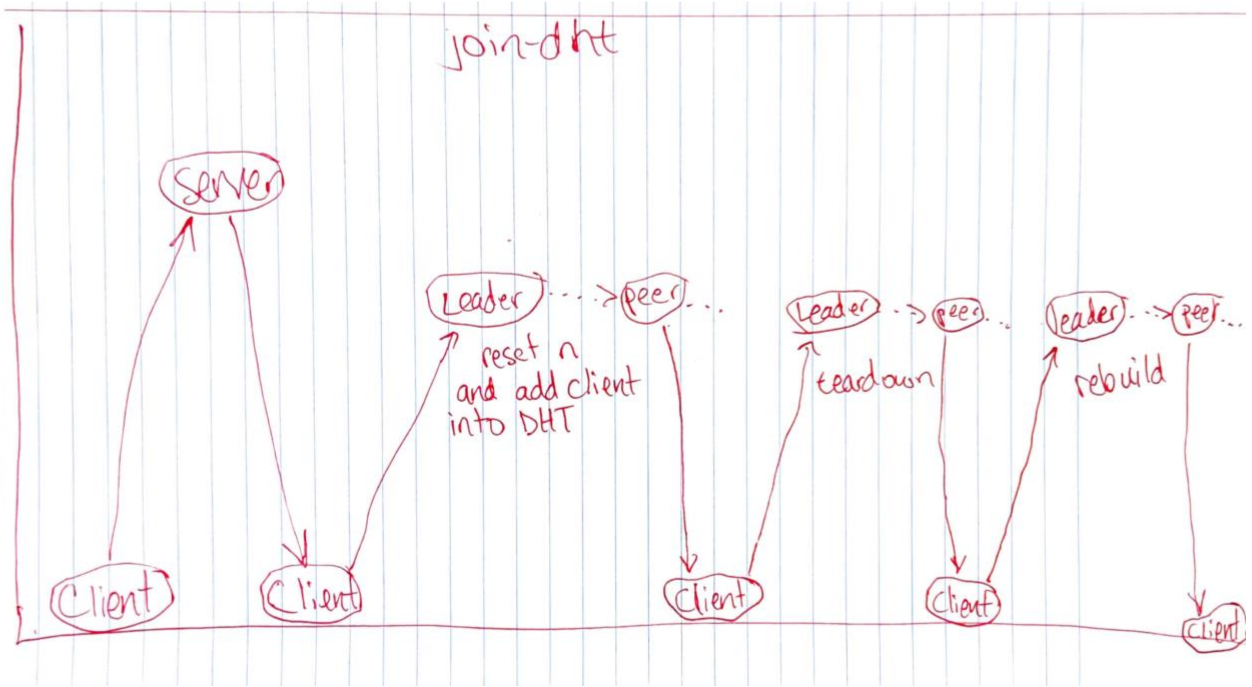
dht-complete / dht-rebuilt



teardown-dht







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.

c. 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.

2. Users – Dictionary

   Python dictionaries are very handy for key – value pairs

3. 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.

4. 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

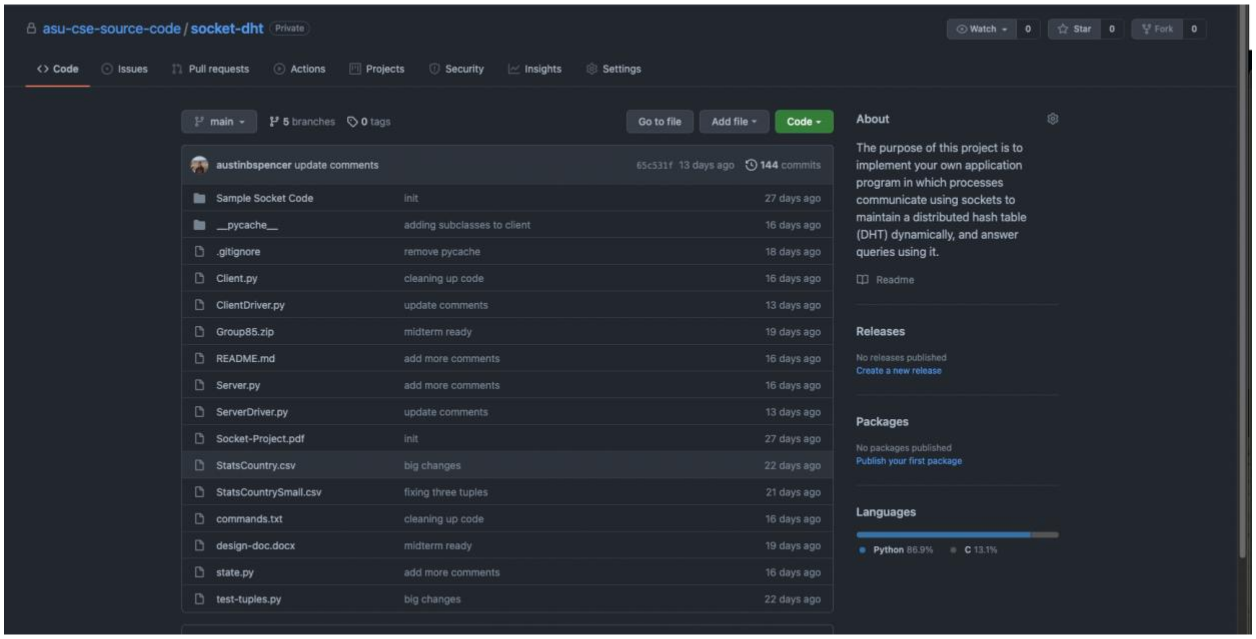
   This is a big object that holds everything the client needs to persist. Tried to make the client code simpler by doing this.

2. 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.

3. 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.

d.Snapshot showing commits in GitHub


asu-cse-source-code / socket-dht Private

Watch 0 Star 0 Fork 0

Code Issues Pull requests Actions Projects Security Insights Settings

main 5 branches 0 tags Go to file Add file Code

austinbspencer update comments 65c531f 13 days ago 144 commits

Sample Socket Code	init	27 days ago
__pycache__	adding subclasses to client	16 days ago
.gitignore	remove pycache	18 days ago
Client.py	cleaning up code	16 days ago
ClientDriver.py	update comments	13 days ago
Group85.zip	midterm ready	19 days ago
README.md	add more comments	16 days ago
Server.py	add more comments	16 days ago
ServerDriver.py	update comments	13 days ago
Socket-Project.pdf	init	27 days ago
StatsCountry.csv	big changes	22 days ago
StatsCountrySmall.csv	fixing three tuples	21 days ago
commands.txt	cleaning up code	16 days ago
design-doc.docx	midterm ready	19 days ago
state.py	add more comments	16 days ago
test-tuples.py	big changes	22 days ago

About

The purpose of this project is to implement your own application program in which processes communicate using sockets to maintain a distributed hash table (DHT) dynamically, and answer queries using it.

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 86.9% C 13.1%

e. Video link

https://youtu.be/TQGvZwVt50g