FastChat Documentation

Telepound

Nov 25, 2022

CONTENTS:

CHAPTER

ONE

TELEPOUND

1.1 clientHelper module

class clientHelper.**Client**(*balancer_*)
Bases: object

This class contains all the functionalities of a client.

__init__(*balancer_*)
Constructor Method. This method initialises member variables of class and also initialised signup/login process.

Parameters **balancer** (*socket.socket*) – This is the socket of the load balancer we are connected to.

attemptLogin()
This method completes login process. It completes the login process including:

1. Takes username and password inputs and verifies if correct or not.
2. Retrieves the public and private of the client and stores them.
3. Sends login details to server.

attemptSignup()
This method completes Signup process. It does the following:

1. Checks if the given username is valid.
2. Confirms password.
3. Generate rsa encryption keys for this client.
4. Sends new user information to server to be stored.

createGroup()
This method handles new Group creation.

It takes groupname as user input, checks if it is valid, and sends group info to server for group to be created.

leaveGroup()
This method handles leaving a group.

It takes groupname from user and if client is in the group, it sends informaiton to server for changes to be made.

modifyGroup(*msg*)
This method handles modification of group, adding of removing members.

If takes required groupname and member name from user, checks if they are valid, and sends information to server for changs to be made.

Parameters **msg** (*string*) – This is the type of modification requested.

packJSON(*type*, *fromUser*, *toUser*, *msg*, *filename*, *groupMsg*)
This method packs a json object with given information, mainly for sending messages to another user.

This method, while reading and attaching files, encrypts them.

Parameters

- **type** (*string*) – This is the type of message we are sending.
- **fromUser** (*string*) – Username of this client, the user sending this message.
- **toUser** (*string*) – Username of client we want to send message to.
- **msg** (*string*) – THe message to be sent (encrypted).
- **filename** (*string*) – The name/path of the file to be attached.
- **groupMsg** (*bool*) – If the message is the group message or individual.

packJSONlogin(*username*)
This method packs a json object for the login info to send to server.

Parameters **username** (*string*) – The username of the client that logged in.

packJSONsignup(*username*, *passwd*, *public_key*, *private_d*, *private_p*, *private_q*, *saltPrivate*)
This method packs a json object for the signup info to send to server.

Parameters

- **username** (*string*) – The username of the client that logged in.
- **passwrd** (*string*) – The password of this client (encrypted).
- **public_key** (*rsa.PublicKey*) – The public rsa key of this client, contains n and e values.
- **private_d** (*string*) – The d value of private key of this client (encrypted).
- **private_p** (*string*) – The p value of private key of this client (encrypted).
- **private_q** (*string*) – The q value of private key of this client (encrypted).
- **saltPrivate** (*string*) – The salt used to encrypt the private keys.

recvMessage(*msgLength*, *inputSocket*)
This method receives a msg is any connection/server tries to send one and handles all cases for types of messages.

Parameters

- **msgLength** (*int*) – The length of the message to be received.
- **inputSocket** (*socket.socket*) – The socket which is trying to send message.

recvServers()
This function is used to receive active servers information from the load balancer.

If it receives a singal assigned server, it stores it's information.

If it receives all the servers online, it stores them and connects to all.

sendMessage(*inputSocket*)
This method handles everything that can happen if The user enters some instruction.

It initializes group creation/modification if appropriate command is entered.

Else it sends specified message to the specified user/group.

Parameters **inputSocket** (*socket.socket*) – The socket in which the user is giving input.

unpackJSON(*packString*)
This message unpacks any json strings and returns the json object.

Parameters **packString** (*string*) – The string to be unpacked.

clientHelper.**groupExist**(*groupname*)
This function finds if a group of given groupname exists or not.

Parameters **groupname** (*string*) – The groupname we want to find.

clientHelper.**userExist**(*username*)
This function finds if a user of given username exists or not.

Parameters **username** (*string*) – The username we want to find.

1.2 balancerHelper module

balancerHelper.**newClientConn**(*msgJson*, *client*)
This method handles all the changes to be made if a new Client logins or signs up.

Parameters

- **msgJson** (*dict*) – The msg received as login/signup request from user.
- **client** (*socket.socket*) – The client where login/signup occured.

balancerHelper.**newServerConn**(*msgJson*, *server*, *addr*)
This method handles events if a new server is made online.

Parameters

- **msgJson** (*dict*) – The msg received from the new server when it came online.
- **server** (*socket.socket*) – The server which came online.
- **addr** (*tuple*) – The addr of the new server

balancerHelper.**packJSONConnClient**(*type*, *to*, *serverIP*, *serverPort*)
This method is to create and pack a Json object with the given information, speciaized for use when a client connects.

Parameters

- **type** (*string*) – The type of the message, the key to distinguish different kind of messages.
- **to** (*string*) – Username of client this message is to be sent to.
- **serverIP** (*string, list*) – the IP details of the server(s) we want to send.
- **serverPort** (*int, list*) – the port details of the server(s) we want to send.

balancerHelper.**serverAssign**(*username*, *client*)
This method assigns the sever, with least load factor, to the given cient.

Parameters

- **username** (*string*) – The username of client which requested to assign server.
- **client** (*socket.socket*) – The client to assign server to.

balancerHelper.**unpackJSON**(*packString*)
This message unpacks any json strings and returns the json object.

Parameters **packString** (*string*) – The string to be unpacked.

1.3 serverHelper module

serverHelper.**addMember**(*msgJson*, *client*)
This method handles events when client sends a request to add a member to a group.

Parameters

- **msgJson** (*dict*) – The request message send by client
- **client** (*socket.socket*) – The client who send the request

serverHelper.**newConnection**(*msgJson*, *client*)
This method handles events if new client connects to this server.

Parameters

- **msgJson** (*dict*) – This is the message sent by client for new connection request.
- **client** (*socket.socket*) – This is the client who established new connection.

serverHelper.**newGroup**(*msgJson*, *client*)
This method handles events when client sends a request to create a new group.

Parameters

- **msgJson** (*dict*) – The request message send by client
- **client** (*socket.socket*) – The client who send the request

serverHelper.**removeMember**(*msgJson*, *client*)
This method handles events when client sends a request to remove a member from a group, or leave a group.

Parameters

- **msgJson** (*dict*) – The request message send by client
- **client** (*socket.socket*) – The client who send the request

serverHelper.**sendMsg**(*msgJson*, *client*, *msg*)
This method handles events when client sends a message to another user or group and the user is online.

Parameters

- **msgJson** (*dict*) – The unpacked message send by client
- **client** (*socket.socket*) – The client who send the request
- **msg** (*bit string*) – The packed/complete message sent by the client

serverHelper.**storeMsg**(*msgJson*, *client*, *msg*)
This method handles events when client sends a message to another user or group and the user is offline.

Parameters

- **msgJson** (*dict*) – The unpacked message send by client

- **client** (*socket.socket*) – The client who send the request
- **msg** (*bit string*) – The packed/complete message sent by the client

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

c

s

INDEX

Symbols

A

B

C

G

L

M

N

P

R

S

U