CONTENTS:

1 FASTCHAT **1**

1.1 client module 1

1.2 database module 2

1.3 mainserver module 4

1.4 server module 5

2 Indices and tables **7**

Python Module Index **9**

Index **11**

CHAPTER

ONE

FASTCHAT

1.1 client module

client.**Receive**(*servernumber: int*, *decode_type=''*, *size: int = 1024*)
This function receives the message from the server

Parameters

- **servernumber** – The denotes server number which is going to send message
- **size** – Denotes size of message

Paramdecode_type Type of decoding

client.**Receive_img**(*servernumber: int*, *size: int = 1024*)
This function is used to receive image from server

Parameters

- **servernumber** – It denotes server number which is going to send message
- **size** – size of message

client.**Send**(*message: str*, *servernumber: int*, *encod_type: str = ''*)
This function sends message from client to server

Parameters

- **message** – Message to be sent
- **servernumber** – It denotes server number to which message to be sent
- **encod_type** – Type of encoding

client.**Send_msg**(*message: str*, *pubkey2: str*, *servernumber: int*)
This function sends rsa encrypted messages to the server

Parameters

- **message** – Message to be sent
- **pubkey2** – Public key of receiver
- **servernumber** – It denotes server number to which message to be sent

client.**decrypt_image**(*data*, *key*, *iv*, *filename*)
This function converts encrypted data back into image

Parameters

- **data** – The encrypted data of image

- **key** – The key with encryption is done
- **iv** – The iv with which encryption is done
- **filename** – The final filename of the image

client.**encrypt_image**(*filename*, *key*, *iv*)
This function converts image into encrypted data using key and iv

Parameters

- **filename** – The name of the image file
- **key** – key for encryption
- **iv** – iv for encryption

client.**getIV**(*blocksize*)
This function generates random number which we can use as encryption iv in AES

Parameters **blocksize** – It denotes the size of iv required

client.**getKey**(*keysize*)
This function generates random number which we can use as encryption key in AES

Parameters **keysize** – It denotes the size of key required

client.**handle_server_instruction**(*message*, *r*)
It handles the commands given by user on terminal

Parameters

- **message** – It denotes message which contain command
- **r** – It denotes the less load having server number

client.**handler**(*signum*, *frame*)
This function exits the user when ctrl+C is pressed on terminal

client.**receive**(*r*)
Pool messages from server (runs in separate thread)

Parameters **r** – It denotes the server number which is going to send message to client

client.**receive2**(*r*)
Fuction which receives from server (used as a helper function)

client.**receive3**(*r*)
Function which receives from server and decypt it

client.**smalltime**()
Helper function to get the server number which has least load

client.**write**()
Hold open input for sending messages (runs in separate thread)

1.2 database module

database.**add_participants_to_grp**(*grpname*, *admin*, *new_participant*, *publickey*)
This function adds groupname, admin name, username and publickey of username in table grp_modified

Parameters

- **groupname** – It denotes group name in which admin wants to add new participant

- **admin** – It denotes the admin of the group
- **new_participant** – It denotes the username
- **publickey** – It denotes the public key of admin

database.**all_members**(*groupname*, *name*)
This function returns all members present in a group

Parameters

- **groupname** – It denotes the groupname
- **name** – It denotes the user name who want this information

database.**change_admin**(*grpname*, *admin*, *new_admin*)
This function helps the admin to change another user to to admin

Parameters

- **grpname** – It denotes the group name for which admin to be changed
- **admin** – It denotes the username of existing admin
- **new_admin** – It denotes the username of new admin

database.**delete_group**(*grpname*, *admin*)
This function deletes all rows containing group name

Parameters

- **grpname** – It denotes the group name which admin wants to delete
- **admin** – It denotes the group admin

database.**delete_participants_from_grp**(*grpname*, *admin*, *member*)
This function deletes the username from the group if fuction is called by admin

Parameters

- **grpname** – It denotes group name from which admin has to delete participant
- **admin** – It denotes admin of the group
- **member** – It denotes the member to which admin wants to kick out

database.**deletion_of_old_msgs**()
This function deletes all messages which are in database for more than 120 seconds time

database.**exit_user**(*username*)
This function changes online status of user to offline while leaving

Parameters **username** – It denotes the username who is exiting (Going offline)

database.**get_public_key**(*name*)
This function gives public key of the given username

Parameters **name** – It denotes username

database.**group**(*groupname*, *admin*, *publickey*)
This function adds groupname, admin name and publickey of admin in table grp_modified

Parameters

- **groupname** – It denotes group name of the new group which user wants to create
- **admin** – It denotes the user who is creating the group
- **publickey** – It denotes the public key of admin

database.**is_online**(*name*)
This function returns the online/offline status of the user

Parameters **name** – It denotes the username of client

database.**leave_grp**(*grpname*, *name*)
This function helps user to leave a specific group.

Parameters

- **grpname** – It denotes the group which user wants to leave
- **name** – It denotes the username of user

database.**msg_delete**(*username*)
This fuction return one message of a user when he/she comes online and delete then from the table

Param It denotes the username who came online

database.**msg_store**(*username*, *msg*, *is_image*)
This function stores the messages when receiver are offline

Parameters

- **username** – It denotes offline user's name
- **msg** – It denotes the message which will be saved in database
- **is_image** – It denotes whether given message is an image or not

database.**no_old_msgs**(*username*)
This functions returns number of undelivered messages when the user comes onlne

Parameters **username** – It denotes the username of client

database.**open_database**()
This function creates database and all required tables. This function is called when main server is initiated

database.**sign_in_up**(*name*, *passw*)
This function takes care of authentication by storing username and encrypted password in table 'clients' and also keeps track of user is online or not along with public key of the user.

Parameters

- **name** – It denotes username
- **passw** – It denotes password of the respective username

database.**update_pubkey**(*name*, *publicKey*)
This function adds public key to the respective username in the table 'clients'

Parameters

- **name** – It denotes username
- **publicKey** – It denotes public key of the respective username

1.3 mainserver module

mainserver.**connections**()
Add client_socket to the list of participant

mainserver.**del_old_msgs**()
This fuction deletes the undelivered messages from the database after specified amount of time (Here 120 seconds)

mainserver.**optserver**()
This function is used to server which has less work. It is base for the load balancing in the program

mainserver.**recieve**(*clientsocket*)
This is the main receive function of mainserver. This provides server which has less load, checks sign in and sign up

1.4 server module

class server.**Participant**(*username: str*, *client_socket*, *thread*, *publickey*)
Bases: object

This class is used for storing data for each client like username and socket but not password

Parameters

- **username** (*string*) – user name of the client which is unique
- **client_socket** (*socket*) – client socket to which server should communicate to the client
- **thread** (*Thread*) – receiving thread for requests sent by client
- **publickey** (*string*) – RSA public key of client but saved as a string

server.**add_member**(*groupname*, *participant_name: str*, *admin_name: str*)
This function add participants in the group if admin gives the command

Parameters

- **groupname** (*string*) – Name of the group in which admin wants to add memeber
- **participant_name** (*string*) – username of the participant to be added in the group
- **admin_name** (*string*) – Username of the admin

server.**handle**(*participant: server.Participant*)
This function sends command to participant is participant is available

Parameters **participant** (*Participant*) – particpant object to which message is to be sent

server.**handle_command**(*participant*, *command*)
This function handles all the commands from the user

Parameters

- **participant** (*Participant*) – particpant object to which message is to be sent
- **command** (*string*) – unique strings for different work

server.**main**()
This function is exclusively used for load balancing. Mainserver pings the server via this function

server.**receive**()
Main loop. Add new clients, old clients in the chat room

server.**receive_from**(*participant*, *encod_type: str = ''*, *size: int = 1024*)
This function is used to receive a message only once from a client when we have object and to decrypt using utf-8 or just receiving without decryption

server.**receive_from2**(*participant_name*, *encod_type: str = ''*, *size: int = 1024*)
This function is used to receive a message only once from a client when we have participant name and it calls another function receive_from by passing socket

server.**remove_member**(*groupname*, *participant_name: str*, *admin_name: str*)
This function removes the given participant from the group

Parameters

- **groupname** (*string*) – Name of the group
- **participant_name** (*string*) – username of the participant to be removed from the group
- **admin_name** (*string*) – Username of the admin

server.**send**(*participant_name: str*, *message: str*, *encd_type=''*)
This function is used to send messages to client when we don't have object when we have only participant name, this function checks in the list of participants and finds the client socket and then sends a message.

Parameters

- **participant_name** (*string*) – name of the particpant to which message is to be sent
- **message** (*string*) – message to be sent to client
- **encd_type** (*string*) – type of encodin(utf-8)

server.**send_group**(*groupname*, *participant_name: str*)
This function sends messages in the group

Parameters

- **groupname** (*string*) – It denotes groupname in which message to be sent
- **participant_name** (*string*) – User name of the user who send message in the group

server.**send_to**(*participant*, *message: str*, *encd_type*)
This function is used to send messages to client when we have participant object along with encrypting using utf-8 or without any encryption (already sha256 or rsa encrypted)

Parameters

- **participant** (*Participant*) – participant object to which message is to be sent
- **message** (*string*) – message to be sent to client
- **encd_type** (*string*) – type of encoding(utf-8)

server.**timestamp**()
Used in function main()

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

c

client, 1

d

database, 2

m

mainserver, 4

s

server, 5

INDEX

A

add_member() (*in module server*), 5
add_participants_to_grp() (*in module database*), 2
all_members() (*in module database*), 3

C

change_admin() (*in module database*), 3
client (*module*), 1
connections() (*in module mainserver*), 4

D

database (*module*), 2
decrypt_image() (*in module client*), 1
del_old_msgs() (*in module mainserver*), 4
delete_group() (*in module database*), 3
delete_participants_from_grp() (*in module database*), 3
deletion_of_old_msgs() (*in module database*), 3

E

encrypt_image() (*in module client*), 2
exit_user() (*in module database*), 3

G

get_public_key() (*in module database*), 3
getIV() (*in module client*), 2
getKey() (*in module client*), 2
group() (*in module database*), 3

H

handle() (*in module server*), 5
handle_command() (*in module server*), 5
handle_server_instruction() (*in module client*), 2
handler() (*in module client*), 2

I

is_online() (*in module database*), 3

L

leave_grp() (*in module database*), 4

M

main() (*in module server*), 5
mainserver (*module*), 4
msg_delete() (*in module database*), 4
msg_store() (*in module database*), 4

N

no_old_msgs() (*in module database*), 4

O

open_database() (*in module database*), 4
optserver() (*in module mainserver*), 5

P

Participant (*class in server*), 5

R

Receive() (*in module client*), 1
receive() (*in module client*), 2
receive() (*in module server*), 5
receive2() (*in module client*), 2
receive3() (*in module client*), 2
receive_from() (*in module server*), 5
receive_from2() (*in module server*), 5
Receive_img() (*in module client*), 1
recieve() (*in module mainserver*), 5
remove_member() (*in module server*), 6

S

Send() (*in module client*), 1
send() (*in module server*), 6
send_group() (*in module server*), 6
Send_msg() (*in module client*), 1
send_to() (*in module server*), 6
server (*module*), 5
sign_in_up() (*in module database*), 4
smalltime() (*in module client*), 2

T

timestamp() (*in module server*), 6

U

update_pubkey() (*in module database*), 4

W

write() (*in module client*), 2