INDIAN INSTITUTE OF TECHNOLOGY, DELHI

REPORT

Assignment 2

ARNAV TULI | ENTRY NO. 2019CS10424

Course - COL334

Prof. Huzur Saran, Prof. Abhijnan Chakraborty

Due September 17, 2021

Chapter 1

Design Decisions

Here, I mention my design decisions that are over and above the assignment's specifications.

1. **Programming Language used:** Python
2. **Server IP Address:** `localhost (127.0.0.1)`
   Server Port Number: `8000`
3. Server-side script (`server.py`) takes no arguments and runs on the above-mentioned IP address and port number by default. **Command:** `python server.py`
4. Client-side script (`client.py`) takes two arguments: `username` and `server IP address`. **Command:** `python client.py [username] [server_addr]`
5. The script first registers the client sockets for sending and receiving messages respectively, in that order. The client-side requires both sockets to be registered before chatting can take place. However, server does not place any such restrictions. Hence, it is possible to modify only `client.py` so that the user registers for only one of the two services.
6. Any errors during registration (ERROR 100 and ERROR 101) lead to termination of client-side script, and the user is expected to start a new process with a *valid* username and server address. This is done to prevent malicious users from spamming the server with irrelevant messages (pre-registration).
7. At server, if there are any parsing errors or header inconsistencies in the packet received from the sender, an error `ERROR 103` is sent back as specified and the socket is closed (both sending/receiving). However, if the same error is raised by the recipient, but not the server, the server sends back the error `ERROR 104` to the sender and does not close the socket. The sender's socket is not closed in this case, as the server was not able to spot any errors in the message that it received from the original sender. The error, ERROR 104 is also sent to sender if the server is unable to forward the message to recipient (but the recipient exists) and/or it receives an unexpected reply from the recipient (not specified in the protocol).
   New Error introduced: `ERROR 104 Unexpected\n\n`.
8. **Broadcast:** Server creates n threads for forwarding sender's message, if there are n total recipients. If any thread results in an error (thread creation, error from receiver), an error (`ERROR 105`) message is sent back to the sender. Otherwise, regular `SENT` message is sent.
   New Error introduced: `ERROR 105 Broadcasting Error\n\n`.

9. The maximum number of bytes received in a single `socket.recv()` call is `2048` (bytes). I have decided this maximum size assuming that the one-line messages will not exceed 1024 characters (bytes), which is a reasonable bound and can always be explicitly coded into the script. Also, the usernames although should not be very long, but the assignment does not place any upper bound on them. Hence, assuming that they will no be no longer than 500 characters (bytes), and having some spare memory for exceptional cases, I decided on a maximum size of 2048 bytes. The maximum is kept as a power of two so as to have the best compatibility with the hardware (word-aligned, double-aligned). This is also recommended in the official documentation of socket API (Python).

10. The state of sockets is noted whenever data is sent or received, and if there is some error (either at client or server), both the socket connections (sending and receiving) are closed, and the process terminates gracefully.

11. Both client-side and server-side scripts **do not** handle timeouts and will indefinitely wait for acknowledgements from the other side. (like response from server to sender, and from recipient to server)