Chat Protocol Requirements

Computer Networks Class
Winter 2020
Suzanne Gerace

Introduction

The purpose of this document is to list out the requirements for this protocol. The requirements are numbered and can be mapped to comments in the source code for the chat protocol.

Requirements

1. DFA Implementation

1a. All states must be changed in the server side with appropriate communication phase between the server and the client. Details of communication and the states that map to this can be found in the Chat Protocol design document under DFA section. **File to find implementation**: server/main.py

1b. All commands coming from the client much check state in the server before taking any action. **File to find implementation:** server/main.py

2. Initialize Server

A command must be run to start the server. **File to find implementation:** server/main.py

3. Initialize Client

3a. A command must be run to initialize the client with a parameter of server hostname or IP address. **File to find implementation:** client/main.py

3b. Once client has been initialized, connect to the server. **File to find implementation:** client/main.py

3c. a command in the client to automatically find the server port. **File to find implementation:** client/main.py TODO

3d. a command in the server to return '210 found chat protocol' if the client has identified the correct port of the server. **File to find implementation:** server/main.py TODO

4. Log in

4a. The ability for client to ask user for username and send username to server to be validated. **File to find implementation:** client/main.py

4b. The ability for server to accept username and store for later validation based on design DFA. **File to find implementation:** server/main.py

4c. Once the server has confirmed that it has received the username, the client should ask for password. **File to find implementation:** client/main.py

4d. Once the user has entered the password in the client user interface, the client should send password to the server. **File to find implementation:** client/main.py

4e. Once the server has received the password, it should validate the username and the password. **File to find implementation:** server/main.py

4f. The server should send a response to the client identifying the outcome of the username and password verification. **File to find implementation:** server/main.py

4g. If username and password are not valid the client should repeat the process until valid. **File to find implementation:** client/main.py

5. Send message

5a. When the user has selected to send a message, the client user interface should first ask the user for the username of the person they want to send a message. **File to find implementation:** client/main.py

5b. When the user has selected another user to send message, the client user interface should ask for the message. **File to find implementation:** client/main.py

5c. When asking for the message from the user, the client user interface should specify that the message should be limited to 1009 characters. **File to find implementation:** client/main.py

5d. Once the client has received message, it should send the message to the server. **File to find implementation:** client/main.py

5e. Once the server has received a command from the client that contains a message to send to a user, it should parse data into the following elements: command, user, message. **File to find implementation:** server/main.py

5f. After the server parses the message from client, it should store the message, username sender and username or receiver in the database. **File to find implementation:** server/main.py

5g. Once message has been processed send appropriate response to the client. **File to find implementation:** server/main.py

5h. Once the client has received the response from server containing the status of processing the message – notify user of response. **File to find implementation:** client/main.py

6. Receive new messages

6a. When the user has selected to receive new messages, the client should send the command to the server to retrieve new messages for user that is signed in. **File to find implementation:** client/main.py

6b. When server receives request from client to send new messages for the user, it should query database for new messages for given user. **File to find implementation:** server/main.py

6c. If server has found new messages for given user, send to the client each message one at a time. **File to find implementation:** server/main.py

6e. For each new message sent to the client from the server, print user that sent the message and the message itself on the command line for the user to see. **File to find implementation:** client/main.py

6f. If server has not found any new messages, notify client. **File to find implementation:** server/main.py

6g. When client receives response code from server, display response message to user in the client user interface. **File to find implementation:** client/main.py

7. Terminate

7a. When user requests to terminate program, send command to server. **File to find implementation:** client/main.py

7b. when client receives terminate command from client, terminate the thread running processes, returning server state to idle. **File to find implementation:** server/main.py

7c. When user requests to terminate program, end client programming currently running. **File to find implementation:** client/main.py

8. User Interface

8a. Once username and password have been verified, give user instructions on what commands are not available to them. Send message, receive new messages, end program. **File to find implementation:** client/main.py

8b. User Interface will be via the command line **File to find implementation:** client/main.py

8c. Once client has been initiated give user instructions on how to log in to the server. **File to find implementation:** client/main.py

9. PDU – client to server

9a. The username command to server. **File to find implementation:** client/main.py

4 bytes	1-10 bytes
USER	Variable: username

9b. The password command to server. **File to find implementation:** client/main.py

4 bytes	1-15 bytes
PASS	Variable: password

9c. The send message command to the server. **File to find implementation:** client/main.py

4 bytes	1-10 bytes	Delimiter	Message 1-1009 bytes
SMSG	Variable: username of receiver of message	:	Message

9d. The receive new messages command to the server. **File to find implementation:** client/main.py

4 bytes
RMSG

9e. The response codes to the server. **File to find implementation:** client/main.py

3 bytes	1-1014 bytes
Response code	Variable: response message

9f. The terminate connection from server command to the server. **File to find implementation:** client/main.py

4 bytes
TERM

9g. Test connection to server – TODO

10. PDU – Server to Client

10a. The response Codes to the client. **File to find implementation:** server/main.py

3 bytes	1-1014 bytes
Response code	Variable: response message

10b. The new messages command to the client. **File to find implementation:** server/main.py

4 bytes	1-10 bytes	Delimiter	Message 1-1019 bytes
RMSG	Variable: username of sender of message	:	Message

11. Concurrency

The server of the chat protocol must be able to handle multiple clients at the same time and stay true to the states of each unique client-server connection.