COP5615 DOSP PROJECT 4 PART 2 REPORT

Group Member Names:

Prapti Akolkar UFID: 22287019

Nikhil Kumar Singh UFID: 40500443

Youtube link for video explanation of our project:

https://youtu.be/tsAecZjJlTE

To run the code:

- Enter into erlang interactive mode using command erl
- We will run the server and client in two different terminals
- First compile the server using the command c(twitterserver).
- Then, compile the client using the command c(twitterclient).
- Run the server using the command twitterserver:start_server().
- We will get the message that server has started then start the client
- In a new terminal, run it using the command twitterclient:start_client().

Implementation:

We have implemented a command line interface through which user can perform the operations like register, tweet, retweet, mentions, viewing the subscribed users, search for hash tags and subscribe. The server and client implementations are as follows:

Server:

- Upon starting both the server and the client, the client can send the messages or requests to the server
- The requests to the server are in JSON format and the server also responds in JSON format.
- The response format of the server is as follows:

  {

  UserName: string

  action: string

  message: string

  code: string

}

- The username will be the name of the client that the server is responding to.
- Action is the type of the operation that is being performed by the server such as register, login, tweet, retweet, subscribe to the users, query, logout.
- Message will be the response each of the above actions
    - **Register:** First, we need to register a new user before performing login operation. In case the user has already been registered then we can directly login with registered username before starting the session.
    - **Login:** After registration, we can login with the registered username and then perform different operations.
    - **Tweet:** The content of the tweet is in the message and is returned to the user's subscribers. Also in case user mentions any other user then it also receives the mentioned user.
    - Retweet: It will be the same action as the tweet with just the content of the message will be added to the front of the tweet.
    - **Subscribe:** It takes the username as input and subscribes to the user
    - **Query:** We can perform three types of queries based on the user input
    - Search for my mentions: It will return the tweets in which the current user is mentioned.
    - **Search for hashtags:** It will return the corresponding tweets based on the hash tag being searched.
    - **View subscribed users tweets:** Taking the input as the username, it will return all the tweets made by the subscribed user.
    - **Logout:** In case the users wish to end the session, they can perform logout

Client:

- In order to perform any action, the user has to first register and login.
- The client will send the request of actions to be performed to the server.
- To establish the connection with the server we use the port number 1204 and the IP address as localhost address.
- These requests are of the following format:

{

username: string

value: string

typereq: string

}

- The username will be the name of the client

- The value attribute will be the type of operations that can be performed:
    - **Register:** The value will be the username of the user.
    - **Login:** The value will be the username of the user.
    - **Tweet:** The value will be the content of the tweet.
    - **Retweet:** The value will be the content of the retweet.
    - **Subscribe:** The value will be the username which the user wishes to subscribe to
    - **Query:** The value will be the name of the subscriber username or hashtag to be searched for
- Typereq: It is the type of request made by the user.

Websocket:

- We used WebSocket for handling large number of tweets and huge user database.
- The number of tweets and users should be handled dynamically and it should be scaled in such a way that new users can register without facing any difficulties
- By following the traditional way the above task will be highly time consuming so by using websockets, as connection to the server is maintained continuously and these tasks will handled easily
- The communication between the server and the client can be performed synchronously as there is no need to perform a request each time a user tweets.
- When a user successfully logs in, a websocket connection is established with the server.

CONCLUSION:

As this project is extension of the previous project where we implemented twitter clone. In this section we optimized the various commands performed by the user and scale the system to handle large number of users and tweets dynamically. We used websockets for optimizing twitter operations such as register, login, tweets, retweet, query, subscribe and logout. We designed a JSON based API that represents all messages and their responses including errors. We also revised the previous version of client and server implementation to include websocket interface. In this way we were able to optimize the actions performed by the user.