COP 5615 – Distributed Operating Systems

Project 4 Part – II

Group members:

Privy Thomas [UFID: 8285-1181]

Soutrik Paul [UFID: 6954-1900]

Introduction

In the part 2 project 4 we have changed the protocol for communication between the server and client to use WebSockets. All communication between server and client is in the form of JSON objects.

Implementation

As all the functionalities of the server and client remain the same from project 1, we only elaborate on the differences in this document. The client opens the connection on a well-known (fixed) socket address of the server. The server socket accepts the connection and forwards the message to the server actor. The server actor then responds through the socket to the client.

All messages between server and client are through a JSON object called Message (defined in DS.fsx). Message object has four fields, and its structure is described in Table 1

Field Name	Description
User	User who made the request
Operation	Unique key value which defines the purpose of the message
Payload	Carries additional data to carry out the operation
TimeStamp	Local time when the message was made

Table 1: Structure of JSON Message object

A client message can have 9 possible values in the Operation field. They are “Register”, “Login”, “MakeTweet”, “SearchMyMentions”, “SearchTweetsByFollowing”, “FollowRequest”, “SearchTweetsByTag”, “Logout”, and “Retweet”. Each one of these messages will have a reply from the server bearing one of “INFO”, “LoginSuccess”, “LoginFailed”, “Error”, or “DisplayTweets”.

```
"Message": {
      "User": {
            "UserName": "Alice",
            "Password": "Passw0rd"
      },
      "Operation": "FollowRequest",
      "Payload": "Bob",
      "TimeStamp": "2020-12-16T15:08:18.7568832-05:00"
}
```

Figure 1: Follow message example

For example, if a user *Alice* with password *Passw0rd* wants to follow *Bob,* the JSON query send by her will be as shown in Figure 1.

The reply to this message will have a Operation as “INFO” and Payload will be “FollowSuccess” or “FollowFailed”. (Fails if user does not exit).

The *Payload* can have different types of data depending on the objective of the message specified in *Operation*. For example, if a client queries for tweets made by people whom he is following, the Server replies with a JSON Array of Tweet objects in the Payload, with a corresponding Operation field which says “DisplayTweets”.

We have used the Suave package for the implementation of WebSockets.

As the server communicates over WebSockets and uses JSON objects to pass data, the client can be made in any language using any framework as long as the JSON messages conform to the standard key values and formats.

How to run the code

Step 1: Unzip the package

Step 2: Run the server on console using “dotnet fsi Proj4Server.fsx”

Step 3: Run the client on a new console using “dotnet fsi Proj4Client.fsx”. Open multiple consoles with the same command to simulate multiple users

Step 4: Follow the onscreen prompts on the client console to Register, Login, Make tweets, etc