DOS Project 4 Part I

Bharath Shankar (UFID: 9841-4098)
Dinesh Singamsetti (UFID: 8336-1334)

Brief Description:

The goal of this project is to implement multiple independent client processes that are spawned during an iteration and handled by a single server process.

The Twitter engine (Server) has been implemented with the following functionality:

- Register account
- Send tweet. Tweets can have hashtags (e.g. #COP5615isgreat) and mentions (@bestuser)
- Subscribe to user's tweets
- Re-tweets (so that your subscribers get an interesting tweet you got by other means)
- Allow querying tweets subscribed to, tweets with specific hashtags, tweets in which the user is mentioned (my mentions)
- If the user is connected, deliver the above types of tweets live (without querying)

Implementation Details:

Server.fsx:

We give Number of Users as the Input. This file contains the code for Twitter engine implementation that is responsible for processing and distributing tweets. The Server Side has few Dictionaries which stores subscribers, tweets, hashtag tables and mentions tables that help the server in processing tweets. It also keeps track of users by maintaining a list of clients. It also communicates directly with client using the host name and port number to distribute the tweets to subscribers and send the query results

Client.fsx:

We give the Number of Users as Input. This file consists of the information pertaining to a single client participant that stands for a single twitter user. This includes the information of its userId which is stored as a numeric string passed to it upon initiation. The underlying logic of maintaining process state is done by a Array that keeps track of various actors and is useful for disconnection and reconnection logic.

Zipf distribution - As per the requirements we were asked to simulate a Zipf distribution on the number of subscribers. For accounts with a lot of subscribers the number of tweets had to be increased so as to increase load on the engine for tweet distribution. This was done by ensuring that for any user account, we sent extra tweets based on the total followers.

Retweets are handled by making every client randomly pick a tweet from one its tweet feed and then retweet it to its own subscribers. A flag is maintained to differentiate retweets from normal tweets as in original Twitter.

<u>Periods of live connection and disconnection for users</u> – Users will tweet, request feeds, retweet when they are alive. They go offline after they get what they need. Simulated our Twitter in a way that initially all users are active and when users decide not to perform any action they go offline.

All the queries with tweets subscribed to, tweets with specific hashtags, tweets in which the user is mentioned (my mentions) were handled successfully by message passing to the server and displaying the list of tweets. The tweets will be delivered live to a user that is online by means of Message passing.

<u>Running the Code:</u>

To Run Server:

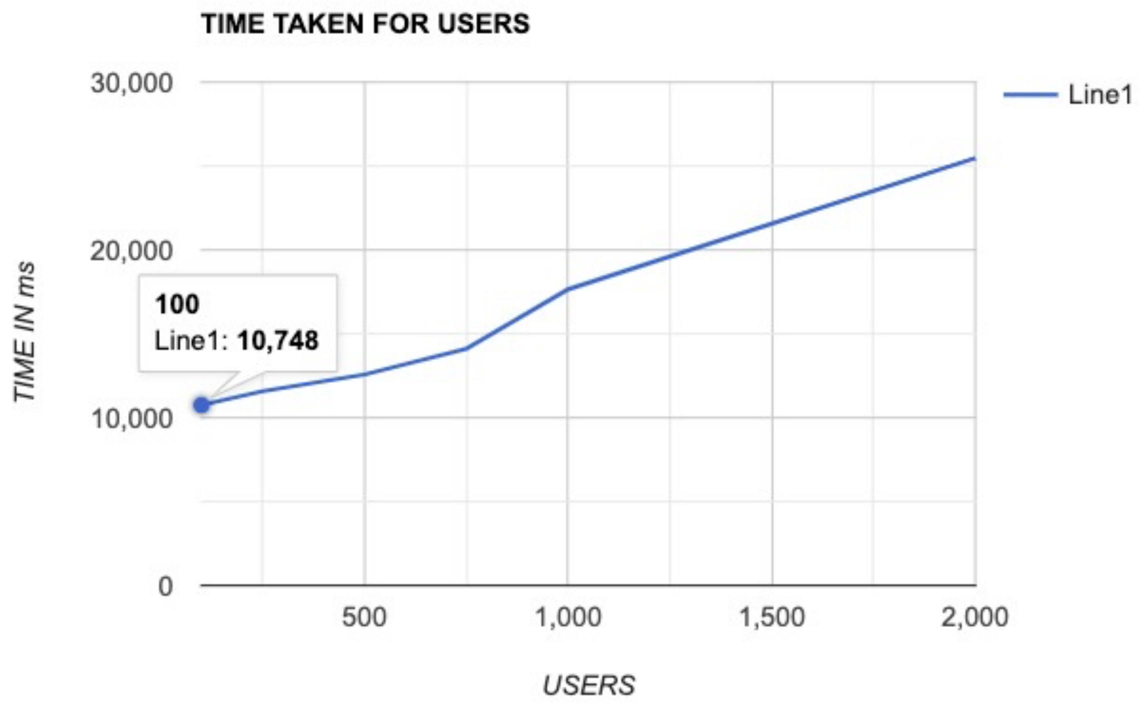
```
dotnet fsi --langversion:preview Server.fsx <NumberOfUsers>
```

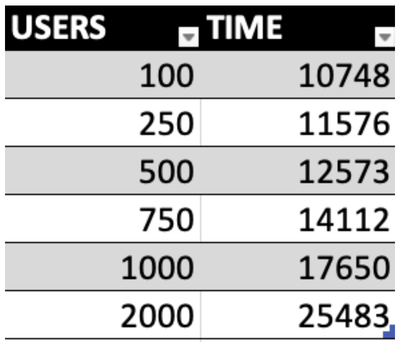
To Run Client:

```
dotnet fsi --langversion:preview Client.fsx <NumberOfUsers>
```

<u>Performance Results:</u>

The time taken for the simulation to run(in milliseconds) vs the number of clients is plotted as a graph.





USERS	TIME
100	10748
250	11576
500	12573
750	14112
1000	17650
2000	25483

We have also printed the stats that is being called from the client at random intervals. We can see the time elapsed, tweet per time, no. of connected and disconnected users and the total users.