DOS Project 4 Part 1 - Twitter Clone

Rupayan Das - 63992113
Pirouz Naghavi - 30158432

Getting Started

Requirements

- **.NET 5 SDK on both machines**

Running the project

Run on server
- *dotnet fsi serv.fsx*

Run on client
- *dotnet fsi sim.fsx <IP> <server port> <no. of clients>*
- eg: dotnet fsi sim.fsx localhost 9002 1000

To make server a remote machine, please change ip address inside config of server file

What is working

Basic Requirements - ALL FINISHED
Bonus - Authentication Implemented using RSA(2048) with SHA256 hashing of stored password

- Account Registration along with Login and Logout - Finished
- Sending Tweets along with mentions and hashtags - Finished
- Subscribing to user's tweets - Finished
- Re-tweets - Finished
- Querying for subscribed tweets, hashtags and user mentions - Finished
- Above requirement only for logged in users - Finished
- Simulate periods of live connection and disconnection for users - Finished
- Simulate a Zipf distribution on the number of subscribers - Finished

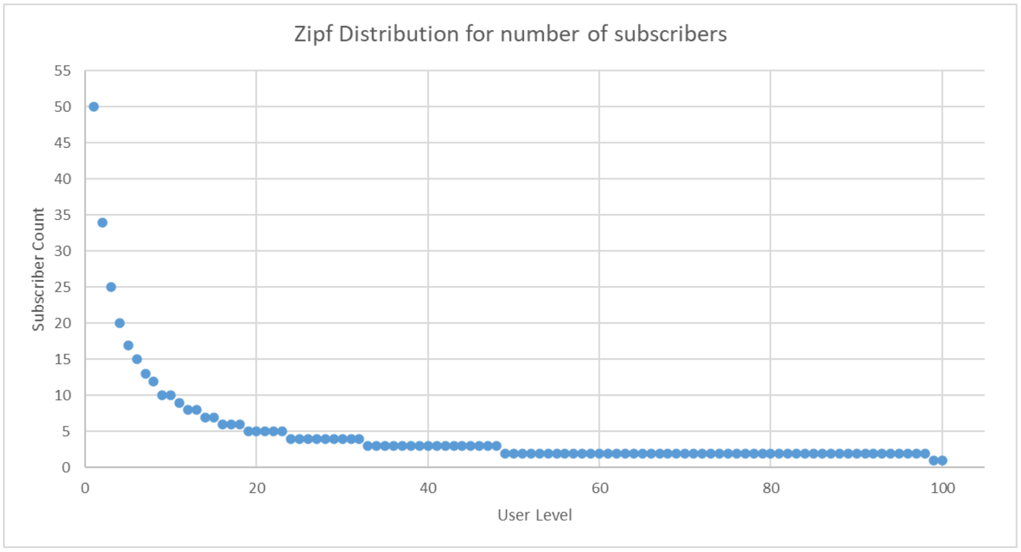
Implementation

- Registration
  The simulator instantiates the number of actors defined by the user. The actors use their actor IDs to generate the username and a random string as password. For the bonus part the actor first asks the server for its RSA sized 2048 bit public key. Once the actor receives the public key, it encrypts the password and sends it with the username to the server. The server decrypts the password with the RSA private key and hashes it with SHA256 hashing algorithm and saves it in a dictionary. The regular code that does not have encryption and hashing will not ask for a public key from the server and sends passwords unencrypted to the server for registration and login. Also, it saves passwords in the dictionary as they are, which are plain text and unhashed.

- Login
  The actor saves the encrypted password in its memory and sends it to the server along with the username to the server to login. The server checks to see if the username is saved in the users dictionary and registers if it doesn’t exist, but if it exists it checks the hashed password dictionary for the given username. It also decrypts the password and hashes it to SHA256 and compares it with the password in the dictionary. If they are the same the actor is logged in successfully, but if they are not the same, it prints an error. The login status of the user is saved in a dictionary based on the username. In tru implementation all these dictionaries would be combined in a database table with a unique ID more unique than the username.

- Logout
  To logout, the actor sends a message to the server with the username, and checks if the status dictionary shows if the user is logged in; if so it logges out the user.

- Adding Subscribers
  Once registration is done, following the ZipF law we distribute subscribers of other users such that if there are 100 users, User 1 will get (100/2) 50 subscribers, User 2 will get (100/3) 33 subscribers, User 3 will get (100/4) 25 subscribers and so on. After the twitter engine adds all the subscribers based on the user's level, it sends the simulator a message saying it’s ready for tweets.



- Sending Tweets and Retweets
  When a client sends a tweet, the twitter engine stores the tweet in three dictionaries. The first dictionary is the tweets dictionary which has tweetID as the key and the tweet as the value. The second dictionary consists of userID as the key and a list of all the tweets made by that user as the value. In the third dictionary, we have userID as the key and a list of all the tweets made by publishers who the user is subscribed to as the value.

  For retweets, we have a list storing all the tweetID's. The twitter engine chooses a random already existing tweet and adds it to the user's tweet dictionary.

- Parsing Tweets
  Each tweet has a user mention and a hashtag. When the engine receives the tweet, a function parses it and adds it to the respective dictionaries by the user mentioned and hashtag.

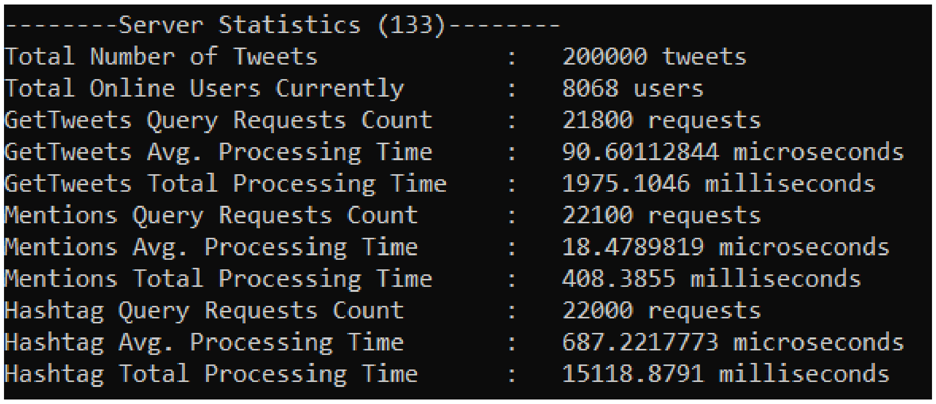
- Get Tweets, Get Mentions And Get Hashtags
  When a client makes either of these requests, the twitter engine handles it by fetching from the respective dictionaries with userID as the key in case of get tweets and get mentions and by the hashtag in case of get hashtag. All the tweets that are fetched get printed on the client side.

- Live Connection and Disconnection
  We simulated periods where the user may be logged in or not. We do this by creating an onlineUsers dictionary. Whenever the engine gets a request for logout, we simply remove the user from the onlineUsers dictionary. Any request made by the user during the logged out period is discarded. When the user logs back in, we add the user back to the onlineUsers dictionary.

- ZipFDistribution
  Along with following the Zipf distribution for the number of subscribers, we also assign each user a level from 1-N where N is the number of users simulated by the simulator. Based on the level of the user, it will send a request after every (level) milliseconds. For example, User 1 will send a request after every 1 millisecond while User 100 will send a request after every 100 milliseconds.

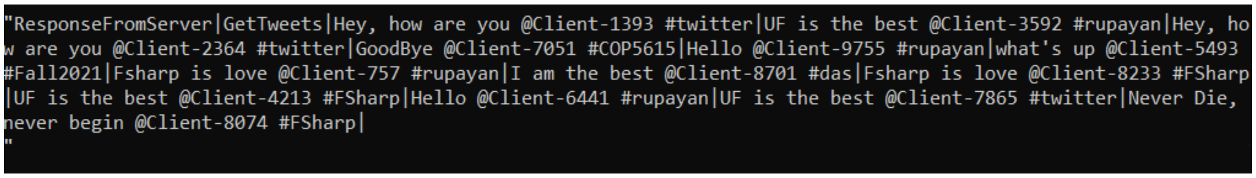
Output

For 10K clients (on server side)



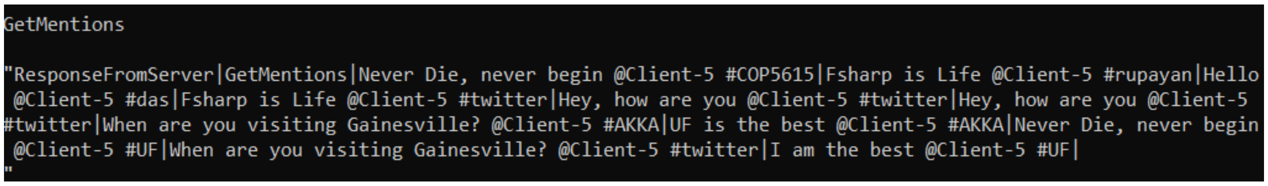
```
--------Server Statistics (133)--------
Total Number of Tweets            :   200000 tweets
Total Online Users Currently      :   8068 users
GetTweets Query Requests Count    :   21800 requests
GetTweets Avg. Processing Time    :   90.60112844 microseconds
GetTweets Total Processing Time   :   1975.1046 milliseconds
Mentions Query Requests Count     :   22100 requests
Mentions Avg. Processing Time     :   18.4789819 microseconds
Mentions Total Processing Time    :   408.3855 milliseconds
Hashtag Query Requests Count      :   22000 requests
Hashtag Avg. Processing Time      :   687.2217773 microseconds
Hashtag Total Processing Time     :   15118.8791 milliseconds
```

Tweets Query (client side)



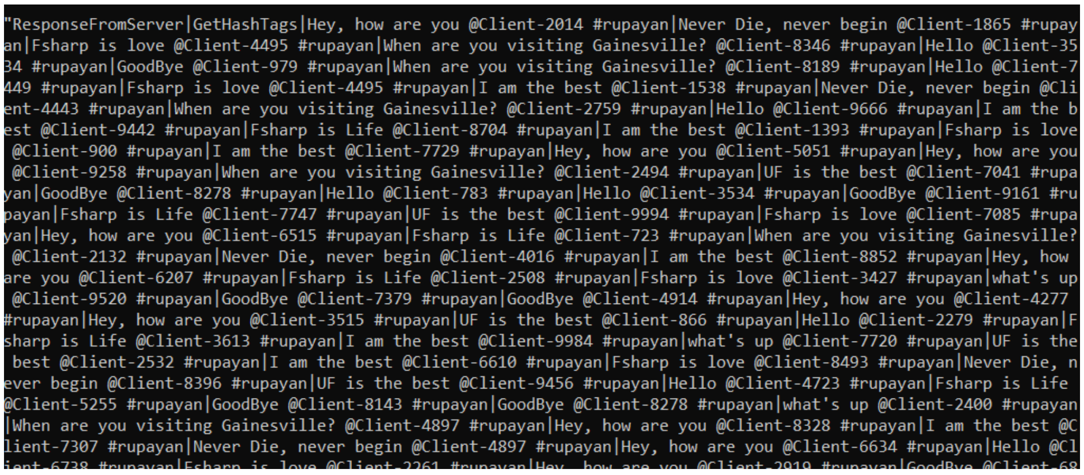
```
"ResponseFromServer|GetTweets|Hey, how are you @Client-1393 #twitter|UF is the best @Client-3592 #rupayan|Hey, ho
w are you @Client-2364 #twitter|GoodBye @Client-7051 #COP5615|Hello @Client-9755 #rupayan|what's up @Client-5493
#Fall2021|Fsharp is love @Client-757 #rupayan|I am the best @Client-8701 #das|Fsharp is love @Client-8233 #FSharp
|UF is the best @Client-4213 #FSharp|Hello @Client-6441 #rupayan|UF is the best @Client-7865 #twitter|Never Die,
never begin @Client-8074 #FSharp|
"
```

Mentions Query (client side)



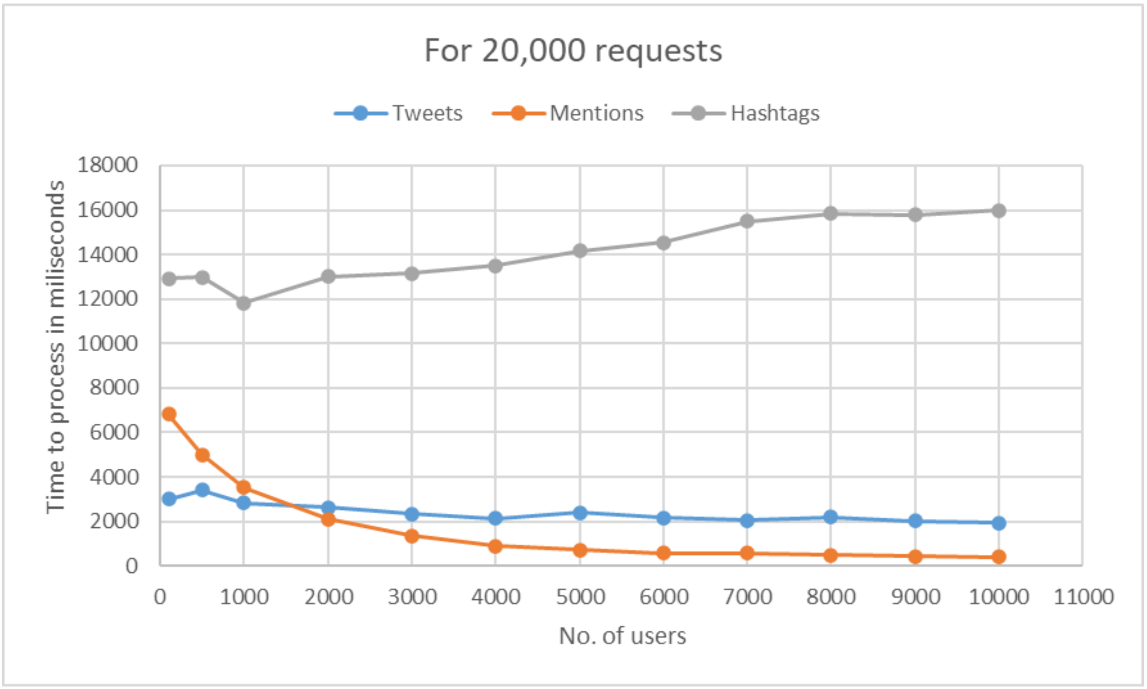
```
GetMentions

"ResponseFromServer|GetMentions|Never Die, never begin @Client-5 #COP5615|Fsharp is Life @Client-5 #rupayan|Hello
 @Client-5 #das|Fsharp is Life @Client-5 #twitter|Hey, how are you @Client-5 #twitter|Hey, how are you @Client-5
#twitter|When are you visiting Gainesville? @Client-5 #AKKA|UF is the best @Client-5 #AKKA|Never Die, never begin
 @Client-5 #UF|When are you visiting Gainesville? @Client-5 #twitter|I am the best @Client-5 #UF|
"
```

Hashtags Query (client side)



```
"ResponseFromServer|GetHashTags|Hey, how are you @Client-2014 #rupayan|Never Die, never begin @Client-1865 #rupay
an|Fsharp is love @Client-4495 #rupayan|When are you visiting Gainesville? @Client-8346 #rupayan|Hello @Client-35
34 #rupayan|GoodBye @Client-979 #rupayan|When are you visiting Gainesville? @Client-8189 #rupayan|Hello @Client-7
449 #rupayan|Fsharp is love @Client-4495 #rupayan|I am the best @Client-1538 #rupayan|Never Die, never begin @Cli
ent-4443 #rupayan|When are you visiting Gainesville? @Client-2759 #rupayan|Hello @Client-9666 #rupayan|I am the b
est @Client-9442 #rupayan|Fsharp is Life @Client-8704 #rupayan|I am the best @Client-1393 #rupayan|Fsharp is love
 @Client-900 #rupayan|I am the best @Client-7729 #rupayan|Hey, how are you @Client-5051 #rupayan|Hey, how are you
 @Client-9258 #rupayan|When are you visiting Gainesville? @Client-2494 #rupayan|UF is the best @Client-7041 #rupa
yan|GoodBye @Client-8278 #rupayan|Hello @Client-783 #rupayan|Hello @Client-3534 #rupayan|GoodBye @Client-9161 #ru
payan|Fsharp is Life @Client-7747 #rupayan|UF is the best @Client-9994 #rupayan|Fsharp is love @Client-7085 #rupa
yan|Hey, how are you @Client-6515 #rupayan|Fsharp is Life @Client-723 #rupayan|When are you visiting Gainesville?
 @Client-2132 #rupayan|Never Die, never begin @Client-4016 #rupayan|I am the best @Client-8852 #rupayan|Hey, how
are you @Client-6207 #rupayan|Fsharp is Life @Client-2508 #rupayan|Fsharp is love @Client-3427 #rupayan|what's up
 @Client-9520 #rupayan|GoodBye @Client-7379 #rupayan|GoodBye @Client-4914 #rupayan|Hey, how are you @Client-4277
#rupayan|Hey, how are you @Client-3515 #rupayan|UF is the best @Client-866 #rupayan|Hello @Client-2279 #rupayan|F
sharp is Life @Client-3613 #rupayan|I am the best @Client-9984 #rupayan|what's up @Client-7720 #rupayan|UF is the
 best @Client-2532 #rupayan|I am the best @Client-6610 #rupayan|Fsharp is love @Client-8493 #rupayan|Never Die, n
ever begin @Client-8396 #rupayan|UF is the best @Client-9456 #rupayan|Hello @Client-4723 #rupayan|Fsharp is Life
@Client-5255 #rupayan|GoodBye @Client-8143 #rupayan|GoodBye @Client-8278 #rupayan|what's up @Client-2400 #rupayan
|When are you visiting Gainesville? @Client-4897 #rupayan|Hey, how are you @Client-8328 #rupayan|I am the best @C
lient-7307 #rupayan|Never Die, never begin @Client-4897 #rupayan|Hey, how are you @Client-6634 #rupayan|Hello @Cl
ient-6738 #rupayan|Fsharp is love @Client-2261 #rupayan|Hey, how are you @Client-2919 #rupayan|GoodBye @Client-68
```

Performance



We observe that as the number of users increase there is varying behaviour between the different types of requests, with the time taken to fetch tweets and mentions decreasing while for hashtags it creeps upwards slightly.