**Project Report**

**COP5615 - Distributed Operating Systems Principles**

The goal is to create an F# application to implement Twitter functionality. Number of users will be provided as command line to the program. The basic functionality includes creating user, sending tweets, subscribing to user’s tweet, retweets and querying tweets.

**Group Members: -**

* + Name: Nikhil Kotian

UFID: 0699-9663

* + Name: Ramandeep Singh

UFID: 8019-7991

**Functionality:**

The command dotnet fsi –langversion:preview project4.fsx 100 will run the project for 100 users. First it prints the total time taken for all the user to tweet once. Next it asks user1 to search a tweet. We can enter mentions using @, hashtags using # and even keywords. It will print the tweets based on the search value entered. Suppose we search using mentions @User87. It will print the tweets that includes user87 as mentions. It then prints the time taken for the program to perform search.

Next it prints the time taken for users to retweet. The timeline of the user will show tweets as well as retweets.

It then randomly selects a user and make it online and create another tweet for that user. example: - User80. Before this the users have only tweeted once. To verify that the online/Offline works, the program first takes a random user that follows User80 and print the timeline while that user is still offline. Since the user is still offline the timeline will show 1 tweet by User80. As soon as User goes live, the timeline will be updated, and it will also include the new tweet by User80 as shown in below image.

**Code Explanation:**

First, we start the simulator and initiate the following map which has the information of the

account that the user is following. It is stores as a set. Next, we create the users and ask the users to tweet. These functionalities are divided into 4 different parts. Below is the detail

explanation of each of these parts.

**Simulate:** This is the first process. Below are the different methods in this process.

* CreateUsers (numusers) – It calls a method in ClientParent process to eventually create the users. It takes the number of users as input.
* Makesets (numusers) – It initiates the following user map for each user. It gives an empty set as value to the map.
* Makefollowers (unum,uname) – It randomly adds the followers and following set for each users to the respective map.
* SendMap (numusers) – It send the user followers and following map to GetMap method in server process.
* UserTweet (numusers) – It starts the tweet process for each user.
* PokeSearch (userId) – It initiates the tweet search process for each user.
* LiveConnection (numusers) –

**ClientParent**: This process has one method.

* TotalUsers (numusers) – This is called by the CreateUsers method in simulate process. This method is responsible of creating the number of users based on the input value. It creates individual actor for each user and later calls the MakeFollowers and SendMap methods in Simulate process.

**Client:** The major functionality implemented in client process is sending and receiving tweets.Below are the different methods in this process.

* Receive (uid,followers,following) – This is called in a method in simulate process which passes the followers and following map which assigned to the local map variable in client process.
* Tweet (uid) – This creates and prints the tweet for each user. This method is called by a method in Simulate which passes the user id as input to this method. It creates the tweet for that particular user and prints it.
* Retweet (uid) – This implements a functionality of retweet where a user retweets a tweet of the user it is following. It is displayed on the timeline of the user and can be seen by its followers.
* GetNewsfeedTweets (tweetersid,userOrRetweetersid,tweetid,tweet,tweetType) – It adds the tweet to the time line of the user. It adds the data to a Set called timelinetweets in the code
* DisplayNewsfeed (uid) – It prints the user and its following list. It also displays the timeline of the user. Same method is used in both tweet and retweet method to display the timelines.
* Performsearch (uid)- It askes the user to enter the search keyword. It passes that value to Lookupstring method in server process which then performs the search operation.
* Displaysearchresults (uid,str,res) – This method displays the search result. If the keyword entered for search does not match with any tweet it gives a message that there are no such tweets else, it displays all the tweets which contains this search keyword.

**Server**: The major functionality it implements is the distribution of tweets. Below are the methods in this process.

* GetMap (sFollower,sFollowing) – It assigns the follower and following list to local maps
* GetTweet (uid,tweetid,tweet) – It is called by the tweet method in client process. It adds the tweet content and tweet id to a Tweet map which is a map data structure. It then calls the GetNewsFeedTweets method in client process to add these tweets to the user timeline
* GetRetweet (tweetersid,tweetid,retweetersid,retweetid) – It has a similar functionality as GetTweet method. The only difference is that this method is for the retweets.
* Createsearchmaps (uid,tweetid,tweet) – This method is used to create different maps that can be later used to search the tweet. It splits the tweet and populates the data in 3 different maps each for hashtags, mentions and the keywords. Later if a user tries to search the tweet it is checked in these maps and tweet id can be fetched from these maps.
* Lookupstring (uid,str) – The above functionality for searching the tweet and checking the searched words in different maps is implemented in this method.

**Results:**

User Tweets

Text

Description automatically generated

Timeline and following list for users:

Text

Description automatically generated

**Time taken to perform tweet, retweet and search for a keyword:**

In this test case we will search a tweet using keyword tweet. Since all the tweets have that keywork It will print all the tweets.

Text

Description automatically generated

Text

Description automatically generated

**Time taken to perform tweet, retweet and search for a hashtag:**

In this test case we will search a tweet using #User. It will print the tweets which has searched hashtags in it.

Text

Description automatically generated

**Time taken to perform tweet, retweet and search for a mention:**

In this test case we will search a tweet using @User. It will print the tweets which has mentions of the user in it.

Text

Description automatically generated

**Time taken to perform tweet, retweet and searching a keyword that doesn’t exist in any tweet:**

In this testcase we have entered a search value which does not exist in any tweet. It will print a message that the string does not exist in any tweet.

Text

Description automatically generated

**Making users online and offline:**

The command dotnet fsi –langversion:preview project4.fsx 100 will run the project for 100 users. First it prints the total time taken for all the user to tweet once. Next it asks user1 to search a tweet. We can enter mentions using @, hashtags using # and even keywords. It will print the tweets based on the search value entered. In below image we have used mentions and searched for @User87. It will print the tweets that includes user87 as mentions. It then prints the time taken for the program to perform search.

Next it prints the time taken for users to retweet. The timeline of the user will show tweets as well as retweets.

It then randomly selects a user and make it online. In our case it makes User50 online and makes it tweet. Before this the users have only tweeted once. To verify that the online/Offline works, the program first takes a random user that follows User50 and print the timeline while that user is still offline. In our case it randomly selected User40 which follows User50. Since User40 is offline the timeline still shows 1 tweet by User50. As soon as User 40 goes live, the timeline will be updated, and it will also include the new tweet by User50 as shown in below image.

Text

Description automatically generated

**Maximum number of users tested: 5000**

**Text

Description automatically generated**