Project 4.1 Report
Twitter Engine

Submitted by:
Naman Arora
(UFID: 3978-0439)
Drona Banerjee
(UFID: 4662-7749)

Objective:

The goal of this first part of a two part project is to implement the user/engine interaction and test it using a simulator built over it. It is supposed to be a twitter clone with functionalities that mimic the real social networking service. This was to be implemented using the concurrent programming model of Elixir[1], The Actor Model.

Architecture:

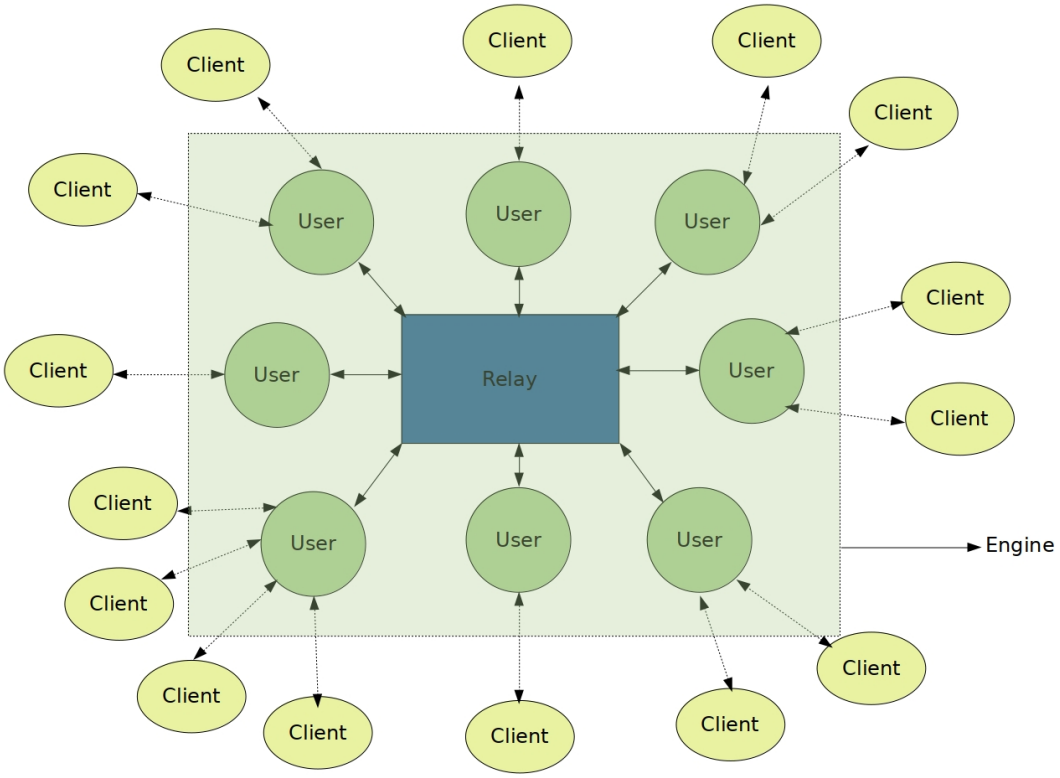

Fig. 1 (Over view of the architecture)

The engine has three main parts:

→ The Client:

This module is a GenServer that mimics the functionality of *logged in* application provided to the user by the service. For example *Twitter for Android.*

→ The User:

This module is a GenServer that identifies as a part of the engine itself and demark the presence of a *signed up* user in the whole network. The customer/client can interact with it, yet it runs as a part of the engine.

→ The Relay:

This is the component that forms the communication bridge between various *signed up* users. All the tweets, follow notifications etc. crossover from one user to the another via *relay* as the mediator.

Functionalities:

We provide a plethora of functionalities, which are:

→ Signup for a new user

→ Login/Logout for an existing user

→ Follow for existing users from A to B

→ Tweet by an existing user

→ Live delivery of tweets if a user is logged in via a registered client

→ Support for mentioning another user in tweets and retweets, which generates a notification for the person mentioned

→ Support for creating new, following and using hastags in tweets and retweets

→ Retweet by an existing user of an already existing tweet by another existing user

→ Query tweets by an existing user on the basis of

- → Tweets that have been mentioned in
- → Tweets by users they follow
- → Tweets by hashtags they follow
- → Tweets they made

→ Fetch one users followers as well as all the users they follow

→ Population of an users timeline by leveraging the above functionality

→ Deletion of a user if requested along with its followed-by table and its entry in all the other users' followed-by tables

Modules:

The above functionalities are provided through an intricate interplay of the modules, which are:

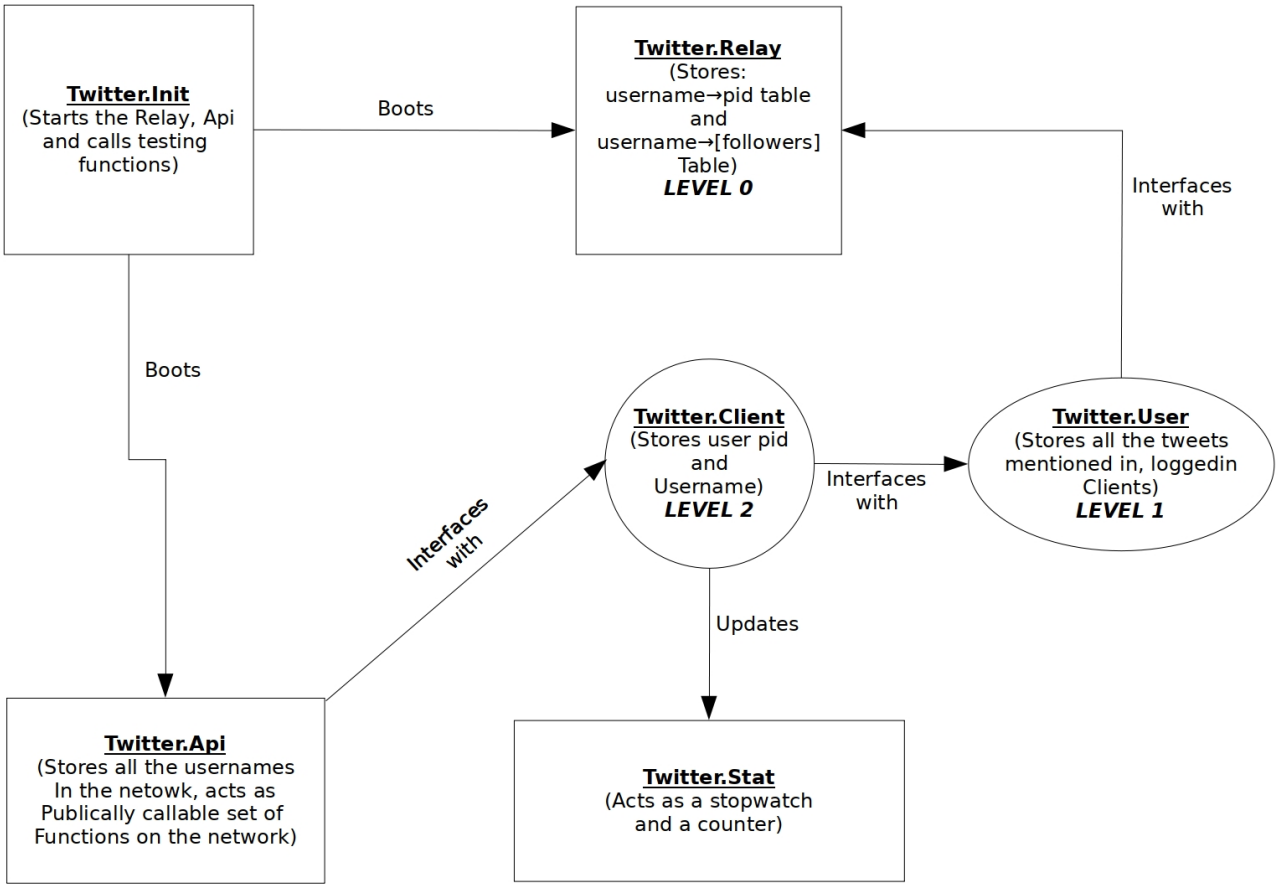

Fig. 2 The modules in the project

Observations:

During the multiple simulation tests conducted, the following obervations were gathered:

Number of Users	Time Taken (in ms)
100	1
1000	10
5000	50
10000	1462
15000	4745
20000	9593

The above table shows how much time was taken for each user to receive a copy of a tweet on its *logged in client*. Note that each user other than the one tweeting was subscribed to receive tweet. We would also like to mention that, after the 5000 users mark, the CPU of the machine started to throttle due to heat and hence the results after that are not on the same processor clock speed. This was tested on an Intel i7-8750H hex core CPU.

Result:

Even though we were able to host a maximum of 100,000 users on the test machine, we are certain that this engine can easily do more due to its highly distributive nature, even within the engine. This Engine provides a layered architecture which finally exposes a top layer API to host a twitter like engine along with compatible clients.

References:

[1] https://elixir-lang.org