

COP5615 DOSP Project 4- Part 2

Cowboy web framework to implement a WebSocket interface on part 1

Team Members:

- Shriyans Nidhish, UFID - 19616510
- Namita Namita, UFID – 48479313

Problem Statement:

Use Cowboy web framework to implement a WebSocket interface to your part I implementation.

Specifically:

- Re-write parts of the engine using Cowboy to implement the WebSocket interface

Now the Cowboy-based Twitter engine (Server) has been implemented with the following functionality:

- Register account
- Send tweet. Tweets can have hashtags (e.g., #COP5615isgreat) and mentions (@DOSPClass)
- 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, and tweets in which the user is mentioned (my mentions)
- If the user is connected, deliver the above types of tweets live (without querying)

YouTube Video Link: - <https://youtu.be/-qyg5Kw-kng>

Steps to Run the Project:

1. Extract the contents of the zip file.
2. Move to the relevant directory using the command `cd` and execute “make run”.
3. Navigate to `http://localhost:8083` in the browser.

Architecture:

Client side:

Each client can perform the following functionalities:

- 1) Register: Before submitting any queries, each client must register with the server.
- 2) Subscribe: Any client can sign up for another.
- 3) Tweet: Only a predetermined number of tweets per input of tweets are allowed from each client. These tweets continue to follow the Zipf distribution, demonstrating that more well-liked people tweet. Random user mentions (shown by the @ sign) and hashtags (indicated by the # sign) may appear in tweets.
- 4) Retweet: Each client gets some tweets that can be retweeted. Retweets begin with "rt:," but otherwise they are communicated to the server in the same way as ordinary tweets. The more popular it is, the more frequently the client will retweet.
- 5) Query my mentions: Each client can choose to search for tweets that include their name. The 100 most recent tweets regarding them are available upon request.
- 6) Query hashtags: Any client can look up any hashtag. They will receive the 100 most recent tweets that contain those hashtags if they request them.
- 7) Disconnect: Each user is free to disconnect from Twitter at any time. When it disconnects, it no longer receives live tweets. Furthermore, it is unable to run hashtags or mention searches.
- 8) Connect: After being disconnected, the client can be reconnected. Upon reconnecting, the client immediately receives every tweet that was sent while it was disconnected. After then, things can resume as normal.

Handler-side:

Tweet Handler: The tweet handler will send the tweets to all the subscribers.

Hashtag Handler: The hashtag handler filters hashtags from every tweet and returns the same to the server.

Get Usernames Handler: The get username handler will fetch the usernames from the list and return the same.

Mentions Handler: The mentions handler will extract the mentions from the tweets and return the same.

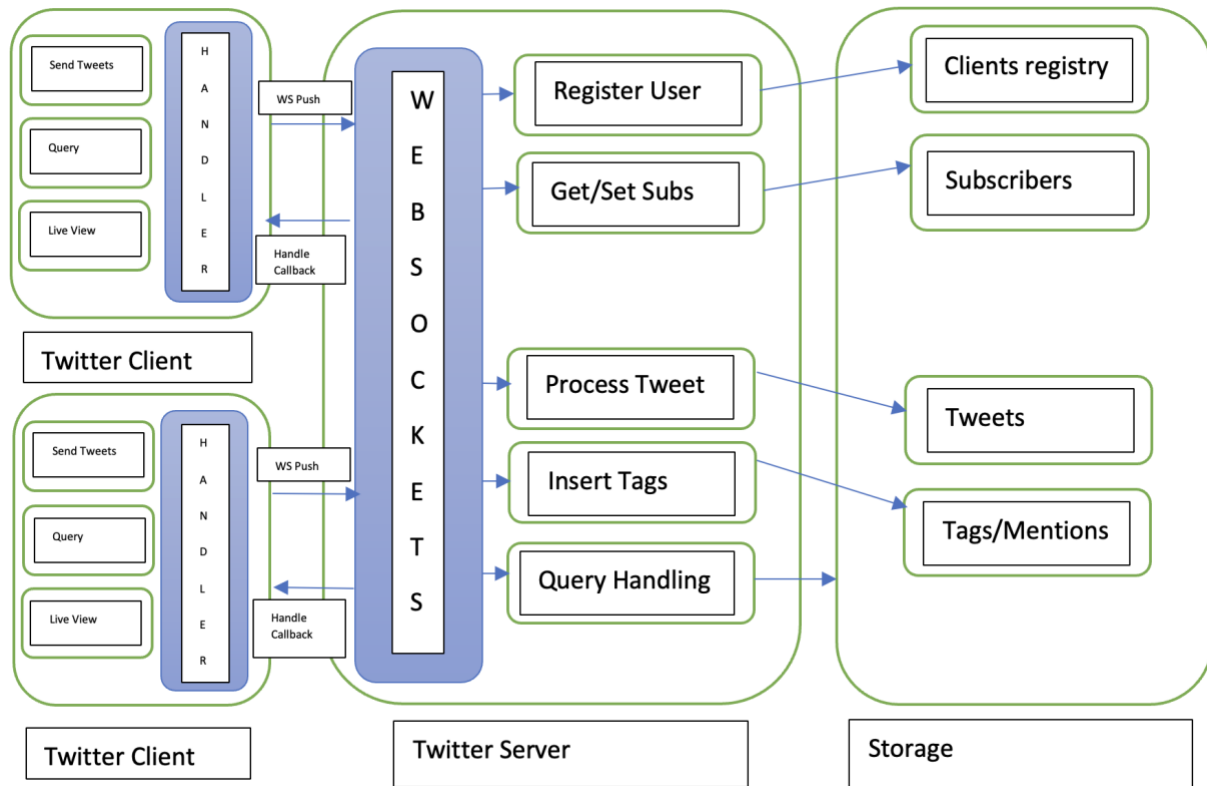


Fig 1.

Implementation:

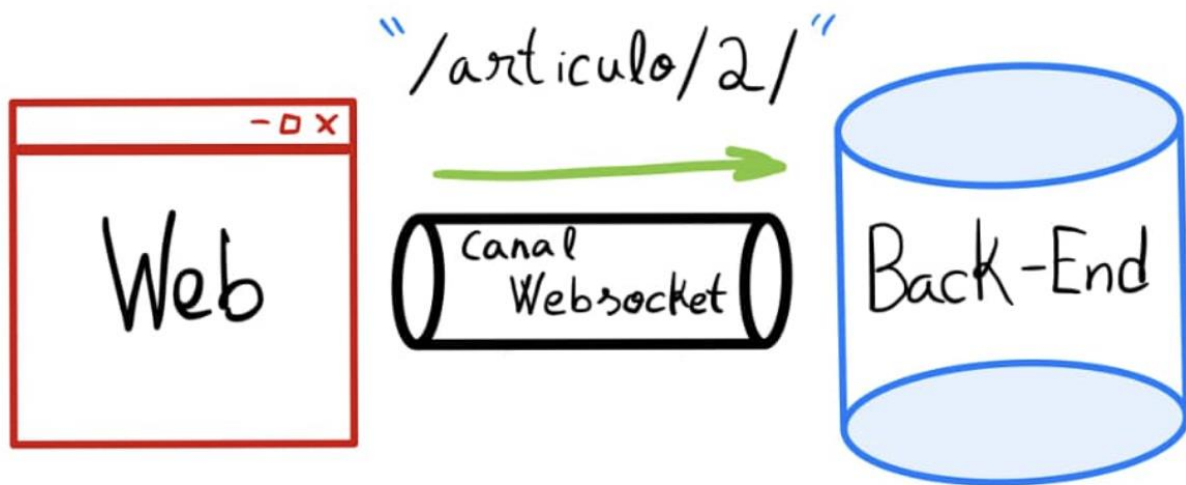


Fig. 2

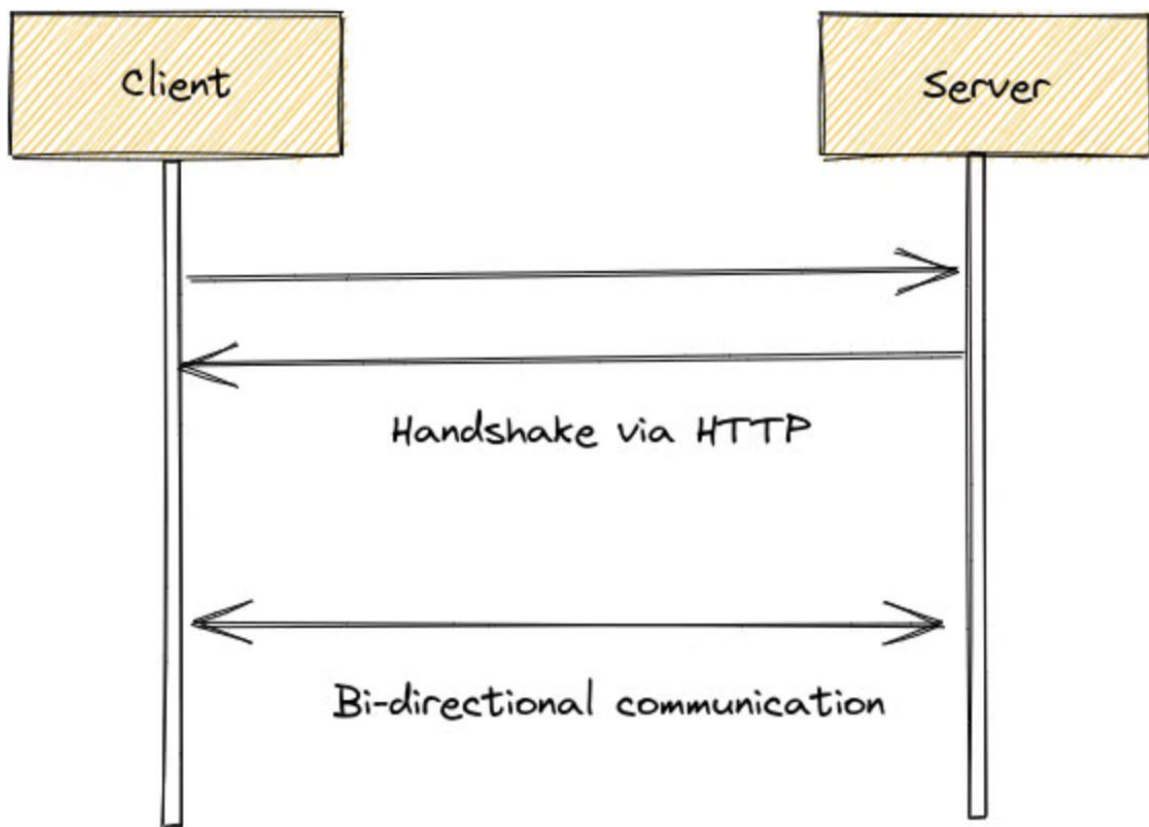


Fig. 3

WebSocket implementation:

According to the flowchart above, the client typically sends an HTTP request to the server to initiate a WebSocket connection.

The WebSocket handshake is then completed by the server by sending a response with an HTTP header to start the WebSocket connection.

The WebSocket protocol messages can be sent asynchronously between the client and server once the connection has been made. We use a straightforward Erlang web server named cowboy to control the socket and the WebSocket protocol to interface the Erlang runtime system with WebSockets.

Cowboy:

Cowboy is the name of the Erlang framework that is most frequently used to build web servers. It's pretty amazing to see what Cowboy accomplishes with such little code. Cowboy works in combination with Ranch (which is a socket worker pool for tcp protocols) and Cowlib (library for message processing). The

process tree begins once your server has been established because Cowboy is meant to construct servers. The only process active in the standalone Cowboy app is cowboy clock. A typical method for launching a cowboy http server is as follows.

```
cowboy:start_http(http, 10, [{port, 80}], [{env, [{dispatch, Dispatch}]}]).
```

Steps to create and run a cowboy application on a Mac device.

- First, let's create the directory for our application all in lowercase.

```
$ mkdir twitterws
```

```
$ cd twitterws
```

- Install erlang.mk: `curl -O https://erlang.mk/erlang.mk`
- Bootstrap the application : `make -f erlang.mk bootstrap bootstrap-rel`
- Run the application : `make run`
- Now, add cowboy to the existing dependencies(in Makefile)

```
PROJECT = twitterws
PROJECT_DESCRIPTION = New project
PROJECT_VERSION = 0.1.0

BUILD_DEPS += relx
PROJECT = twitterws

DEPS = cowboy
dep_cowboy_commit = 2.9.0

DEP_PLUGINS = cowboy

include erlang.mk
```

Fig. 4

- Add Routing and listening in the `<app name>_app.erl`
- Run the application : `make run`

Code Screenshot

Web-socket handler

```
init(Request, ProtocolState) ->
  {cowboy_websocket, Request, ProtocolState,#{idle_timeout => 30000}}.

Used 0 times | Cannot extract specs (check logs for details)
websocket_init(ProtocolState) ->
  {[], ProtocolState}.

Used 0 times | Cannot extract specs (check logs for details)
websocket_handle({text, Data}, ProtocolState) ->
  SplitString = string:split(binary_to_list(Data), "-",all),
  Event = lists:nth(1, SplitString),
  if
    Event == "update_handler"->
      ProfileId = helper:getProfileId(),
      client:updateHandler(lists:nth(2, SplitString),self(),ProfileId),
      String = list_to_binary(""),
      [{text, <<String/binary>>}, ProtocolState];
    Event == "tweet"->
      tweet(lists:nth(2, SplitString), lists:nth(3, SplitString)),
      String = list_to_binary(""),
      {reply,{text,<<String/binary>>},ProtocolState};
    Event == "subscribe"->
      subscribe(lists:nth(2, SplitString), lists:nth(3, SplitString)),
      String = list_to_binary(""),
      {reply,{text,<<String/binary>>},ProtocolState};
    Event == "hashtag"->
      ResponseString = hashtag(lists:nth(2, SplitString), lists:nth(3, SplitString)),
      String = list_to_binary("search-++ResponseString"),
      {reply,{text,<<String/binary>>},ProtocolState};
    Event == "mention"->
      ResponseString = mention(lists:nth(2, SplitString), lists:nth(3, SplitString)),
      String = list_to_binary("search-++ResponseString"),
      {reply,{text,<<String/binary>>},ProtocolState};
    Event == "search"->
      ResponseString = searchTweet(lists:nth(2, SplitString), lists:nth(3, SplitString)),
      String = list_to_binary("search-++ResponseString"),
      {reply,{text,<<String/binary>>},ProtocolState};
    Event == "retweet"->
      retweet(lists:nth(2, SplitString), lists:nth(3, SplitString)),
      String = list_to_binary(""),
      {reply,{text,<<String/binary>>},ProtocolState};
  true->
```

Routing: -

```
start(_Type, _Args) ->
  Id = twitter:startServer(),
  {ok, Fd} = file:open("server.txt", [write]),
  file:write(Fd, pid_to_list(Id)),
  file:close(Fd),
  Dispatch =
    cowboy_router:compile([{'_', [
      {"/", cowboy_static, {file, "/Users/shriyansnidhish/Rahul/UF/Fall22/DOSP/Project4/part2/twitterws/static/new/regist", []},
      {"/register", register_handler, []},
      {"/:name/main", cowboy_static, {file, "/Users/shriyansnidhish/Rahul/UF/Fall22/DOSP/Project4/part2/twitterws/static", []}},
    ]}],

  DispatchWebSocket = cowboy_router:compile([{'_', [{"/", main_handler, []}]}]),
  {ok, _} =
    cowboy:start_clear(my_http_listener, [{port, 8082}], #{env => #{dispatch => Dispatch}}),
  {ok, _} = cowboy:start_clear(ws, [{port, 8889}], #{env => #{dispatch => DispatchWebSocket}}),
  twitterws_sup:start_link().

Used 0 times | Cannot extract specs (check logs for details)
stop(_State) ->
  ok.
```

Functionality: -

```
tweet(Username, Tweet, ServerId) ->
  ServerId ! {tweet, Username, Tweet, self()},
  receive
  | {ok, ReplyString} ->
  |   ReplyString
  end.


Used 2 times | Cannot extract specs (check logs for details)
subscribeOtherUser(FirstUsername, SecondUsername, ServerId) ->
  ServerId ! {subscribe, FirstUsername, SecondUsername, self()},
  receive
  | {ok, ReplyString} ->
  |   ReplyString
  end.

Used 2 times | Cannot extract specs (check logs for details)
sendRetweet(Username, Tweet, ServerId) ->
  ServerId ! {retweet, Username, Tweet, self()},
  receive
  | {ok, ReplyString} ->
  |   ReplyString
  end.

Used 2 times | Cannot extract specs (check logs for details)
subscribeSearch(FirstUsername, SecondUsername, ServerId) ->
  ServerId ! {search_by_subscribe, FirstUsername, SecondUsername, self()},
  receive
  | {ok, ReplyString} ->
  |   ReplyString
  end.
```


Output Screenshot:

Register here to proceed.



Register

Username

Shriyans

Password


Email

nidishsawdwar@gmail.com

Submit

Registration Page

Twitter



What's happening?

Tweet

Subscribe

Enter username

Subscribe

Search Hashtags here.

Enter hashtags

Search

Search Mentions here.

Enter mentions


Search

Search Subscribers here.

Enter subscriber

Dashboard

Twitter



#DOSPIsbest

Tweet done!

Close

Subscribe

Enter username

Subscribe

Search Hashtags here.

Enter hashtags

Search

Search Mentions here.

Enter mentions

Search

Search Subscribers here.

Enter subscriber

Tweeting with success alert

Twitter

Tweet

Subscribe

Enter username

Subscribe

Search Hashtags here.

Enter hashtags

Search

Search Mentions here.

Enter mentions

Search

Search Subscribers here.

Enter subscriber

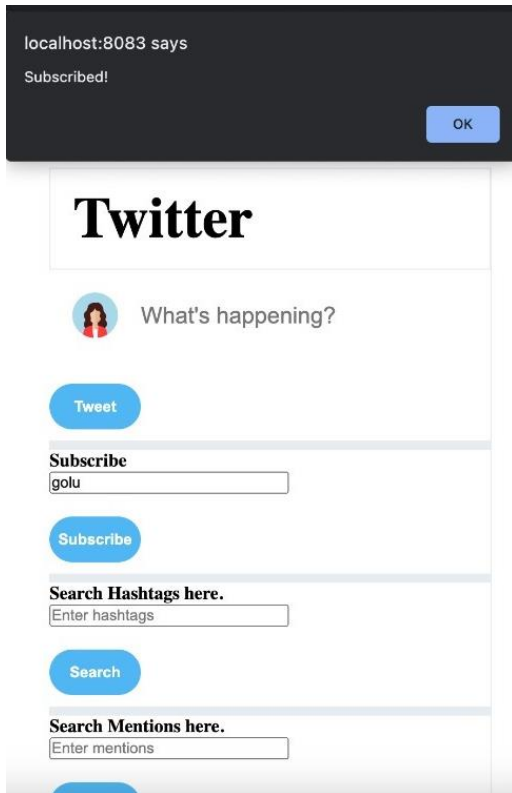
#DOSPIsbest

retweet

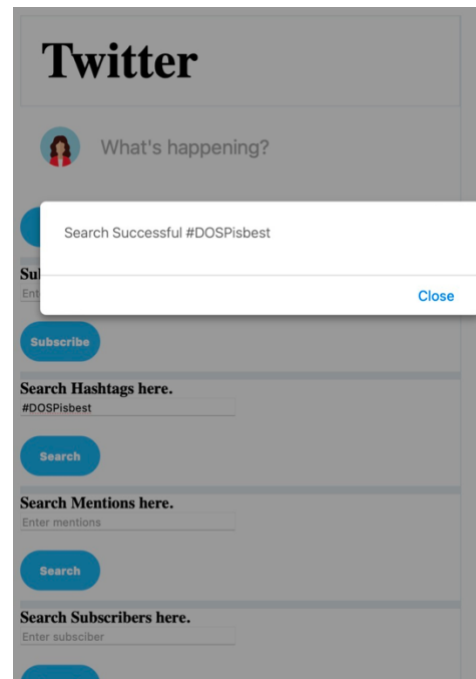
#Project4

retweet

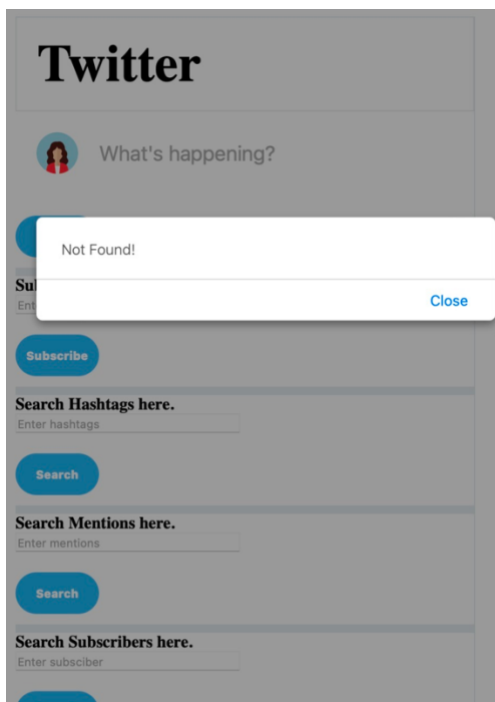
Successful tweets visible at bottom of page



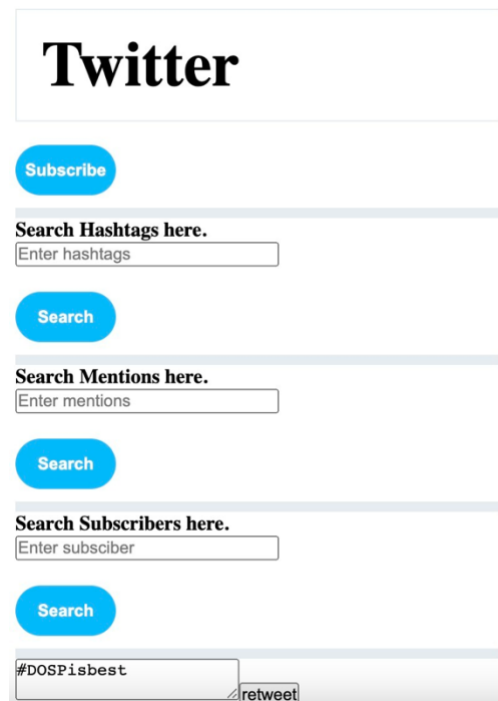
Subscribing with success alert



Tweet search successful



Tweet search unsuccessful



Tweet Visible at bottom of page

localhost:8083 says
Search Successful #hello

OK

Twitter

Subscribe

Enter username

Subscribe

Search Hashtags here.

Enter hashtags

Search

Search Mentions here.

Enter mentions

Search

Search Subscribers here.

golu

Search

Upon searching subscriber,
tweet of subscriber is displayed

localhost:8083 says
Search Successful @rahul

OK

Twitter

Tweet

Subscribe

Enter username

Subscribe

Search Hashtags here.

Enter hashtags

Search

Search Mentions here.

@rahul

Search

Search Subscribers here.

Search mention successful