Singer

A decentralized timeline service

G16
André Flores - up201907001
António Oliveira - up202204184
Diogo Faria - up201907014
Tiago Rodrigues - up201906807

Project Description

- Develop a social network akin to Twitter, but decentralized.
- Users need to be able to post, follow and unfollow other users.
- Users can forward other users' content, if needed.

Technologies Used

- The entirety of the project was developed using **Python**.
- A **Kademlia** implementation on **Python** was used as the Distributed Hash Table (DHT).
- For peer-to-peer communication **asyncio** was used.
- For data persistence, a **SQLite 3** embedded transactional database was used.
- Lastly, for the GUI, **PySide6** was used, which provides access to the **Qt** framework.
- We used an **MVC** architecture.

Communication

- Each peer continuously runs an **asyncio** server on the IP and port specified in the command line arguments, this server listens for and handles incoming messages.
- To send messages we use **asyncio**'s *open_connection*.

GUI

- The GUI is updated periodically through help of signals and a worker thread. When it receives a signal it retrieves information from the database. This ensures a user never has to click a refresh button to be displayed with updated information.

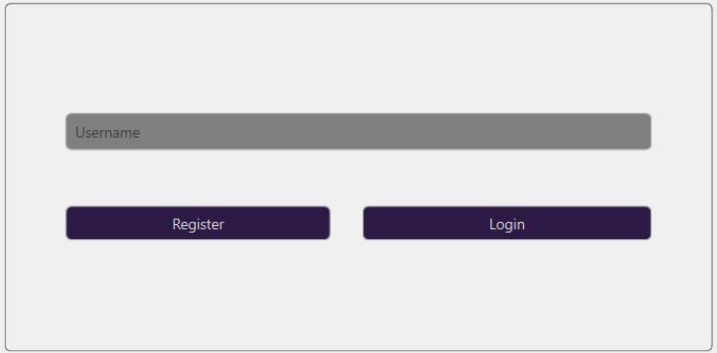
Joining the Network

- When a node joins the network it will start a Kademlia node server at the port specified in the command line arguments.
- It will also try to bootstrap itself with the nodes specified in the file in the command line arguments.
- When a node joins the network it will also send a message to each of its bootstrap nodes for them to set their own info on the network. This is a workaround on one of our DHT library's shortcomings because it only lets us set a key value pair when there is more than a node on the network and bootstrap nodes are the first on the network.
- In DHT we store a user's followers, following, IP and port, whether they are online or not and their last post's id as the value, the key is the username.

Authentication

- When a user **registers**, first a check is made to find whether the name already exists, in which case the register fails, followed by setting their Kademlia node server information.
- Lastly, a local database is created to store whatever is necessary.
- For **login**, there's an attempt at getting the user's information from other nodes in the network.
- If it gets it, then it sets its information to that, else it checks if there's a local database for that user and gets their information from it.
- On the case that their info isn't found, then the login simply doesn't work.
- After the user's information is set, their timeline is synced depending on whatever posts they might've missed from their followers while offline.

Singer



Diogo Logout

Write something... Create Post

André 2022-12-11 10:58:37

I made a post!

Diogo 2022-12-11 10:56:05

Hello

Followers:

Tiago

Following:

Tiago Unfollow

André Unfollow

Username Follow

Subscribe/Follow

- When a user subscribes or follows another user, the application updates their information in the DHT and persists new follow in a database to reflect this requested change.
- If the other user is also in the DHT, their followers list is updated. If the additional condition of this user being online and logged, a follow message is sent and their database is updated.

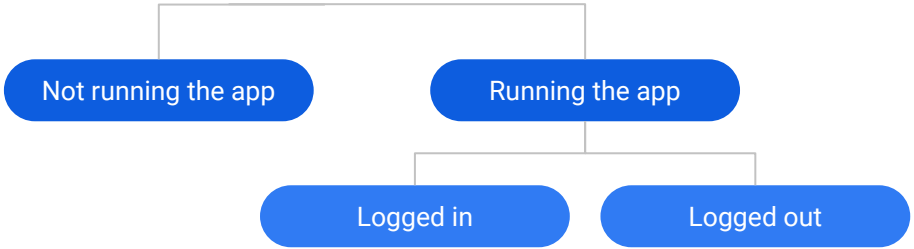
Unsubscribe/Unfollow

- When a user unsubscribes or unfollows another one, the application updates their information and persisted followers list to reflect this requested change.
- If the other user is also running the application, their followers list is updated. If the additional condition of this user being online and logged, an unfollow message is sent and his persisted data is updated.

Post

- The application offers the user the functionality to create text posts that are shared with their followers.
- The followers that receive this message update their persisted posts data.
- If any follower is not online to receive this message, their persisted posts are updated with the ones missed during his offline period the next time he logs in the application.

Fault Tolerance



- Missed posts from followed users are retrieved after login by checking if the followed user is online or if other of their followers are online and then retrieving the missing posts.
- A user can unfollow or follow an offline user and this process is consistent immediately.
- If the node went down the process is only eventually consistent:
  - Follow - When the user who followed logs in they will notify any unnotified follows.
  - Unfollow - When the unfollowed user tries to send a post to the unfollowing user and that user is online they will ignore the post and resend the unfollow message.

Demo

Questions...?