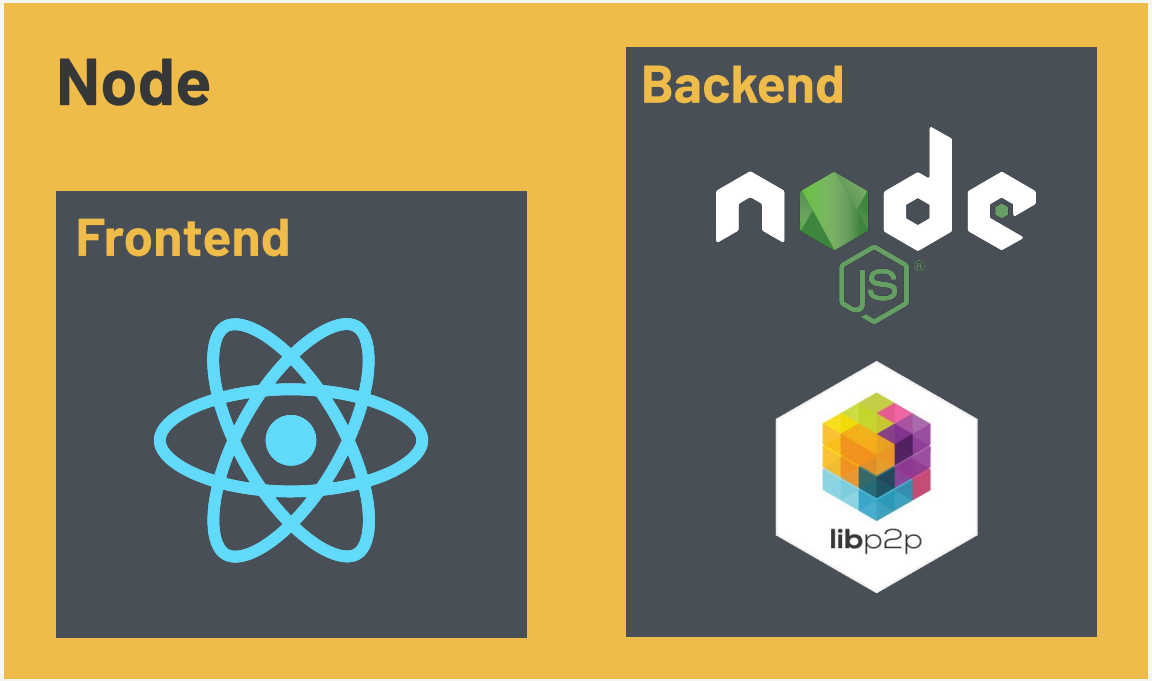
Decentralized Timeline

G14
António Bezerra up201806854
Catarina Fernandes up201806610
Gonçalo Alves up201806451
Pedro Seixas up201806227

Choice of Technology



Peer Communication

For communication between users, **TCP** was used for the following reasons:

- **Reliability**
- **Error Detection**
- **Security**

Peer Discovery

As this project is a Proof of Concept, a simple discovery mechanism in a private network was implemented. For this, we made use of **libp2p**'s **Multicast DNS** and **Bootstrap** methods.

Kademlia

Distributed hash tables can scale for large volumes of data across many nodes. With that in mind, we thought it would be helpful for this project to use a DHT.

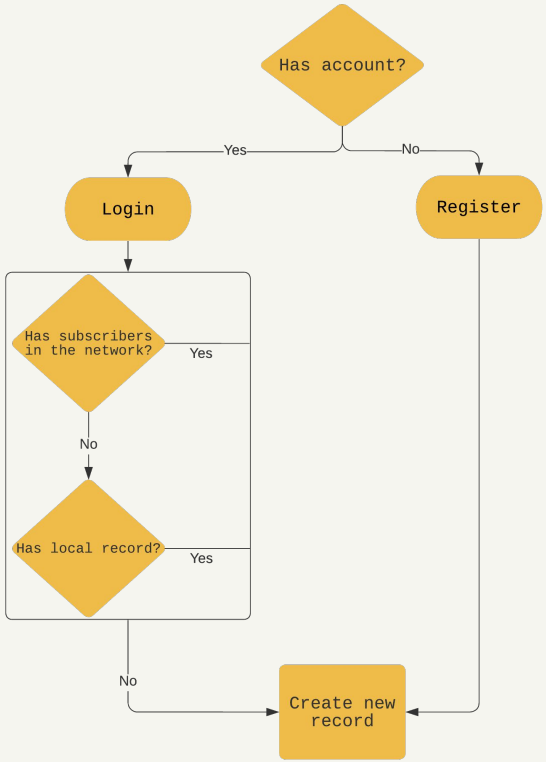
Kademlia DHT was used both in **user authentication** and **content discovery**.

Authentication

Authentication is possible using the Kademlia DHT:

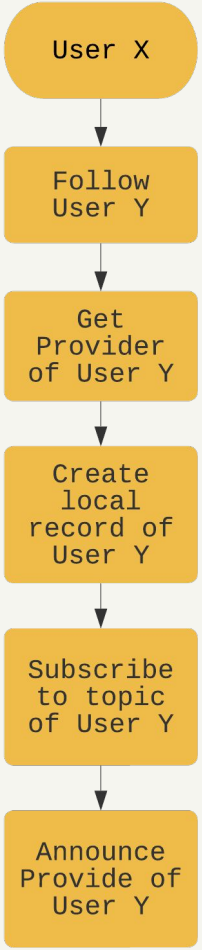
- **Register**: Hashed password is put in the DHT with the username as key;
- **Login**: Check for the hashed password in the DHT and login if passwords match.

Uniqueness of the username is also ensured using this method.



Timeline

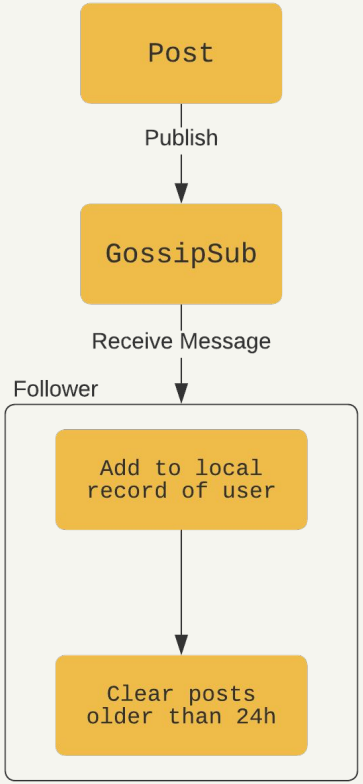
In order to provide a **Timeline** of a user, we made use of **libp2p's** Kademlia DHT, namely the **provide** and **findProviders** features.



Updating Followers

In order to keep all followers updated about some user, we opted for a publish/subscribe pattern, using the **libp2p**'s **GossipSub** implementation.

- **Follow**: User is subscribed to the other user's topic;
- **Unfollow**: User in unsubscribed from the other user's topic.



Updating Followers

Messages received in the topic *user1* can be:

- **POST <POST>**: *user1* has published a new message;
- **FOLLOW *user2***: *user2* has followed *user1*
- **UNFOLLOW *user2***: *user2* has unfollowed *user1*
- **FOLLOWED *user2***: *user1* has followed *user2*
- **UNFOLLOWED *user2***: *user1* has unfollowed *user2*

Ephemeral Content

A subscriber of a user keeps a record of the timeline of said user for a **limited** amount of **time** and **space**.

A pure spatial or a pure temporal approach to limiting this forwarding of information would not be ideal, since both of them has downsides. To work around this, the subscriber uses both spatial and temporal information to limit the storage.

- Only stores messages more recent than some **T** value.
- Only stores at maximum **N** messages of a user.

Local Storage

A user keeps his own profile in local storage, so that his timeline is never lost, even if no followers are active.

When a user logs in, he will first try to get the latest version of his profile from his followers. If no followers are online, the local storage version is loaded, to restore all posts, followers and following users.

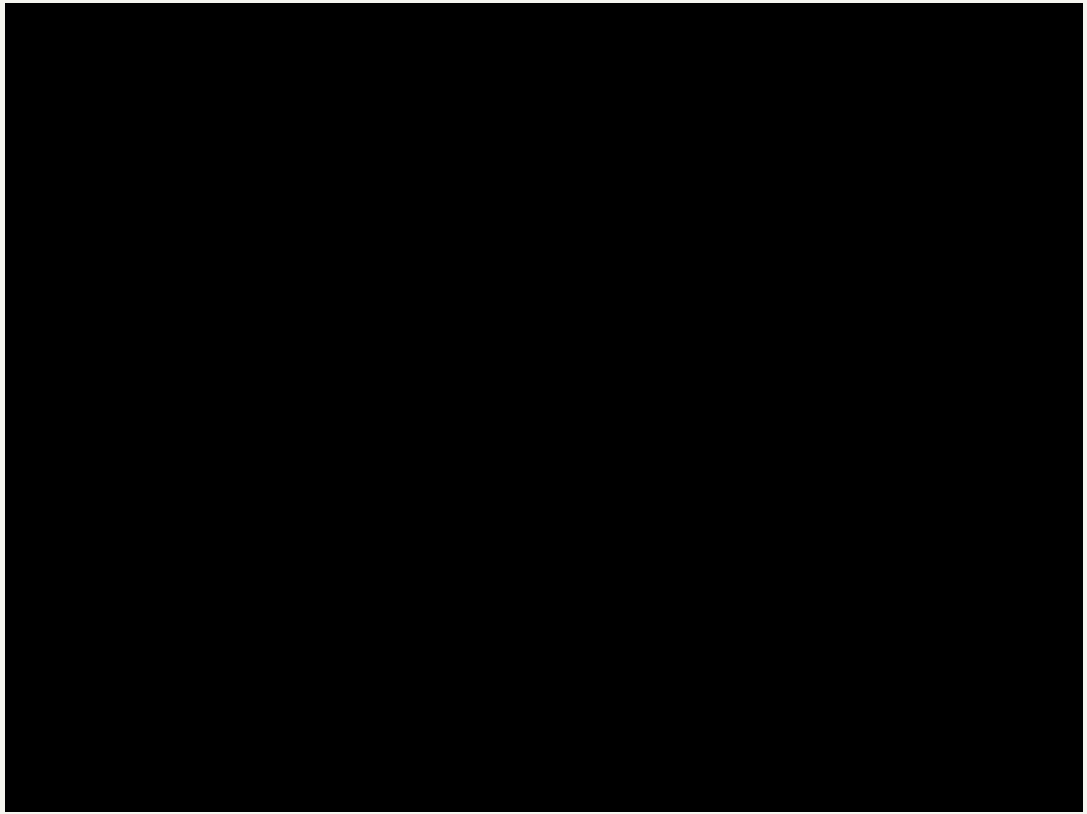
The **posts** and **following** users are guaranteed to be **updated**, since they can only change whenever the user is online, but this version of the profile may have **outdated followers**.

Heartbeat

To prevent the outdated followers whenever loading the local storage version of a user's profile, the **heartbeat** protocol was made. Whenever a user logs in, he will confirm with his followers if they are still following and update his followers list.

This task is being done in the background, so it does not affect the user experience.

Demo



Thank you!