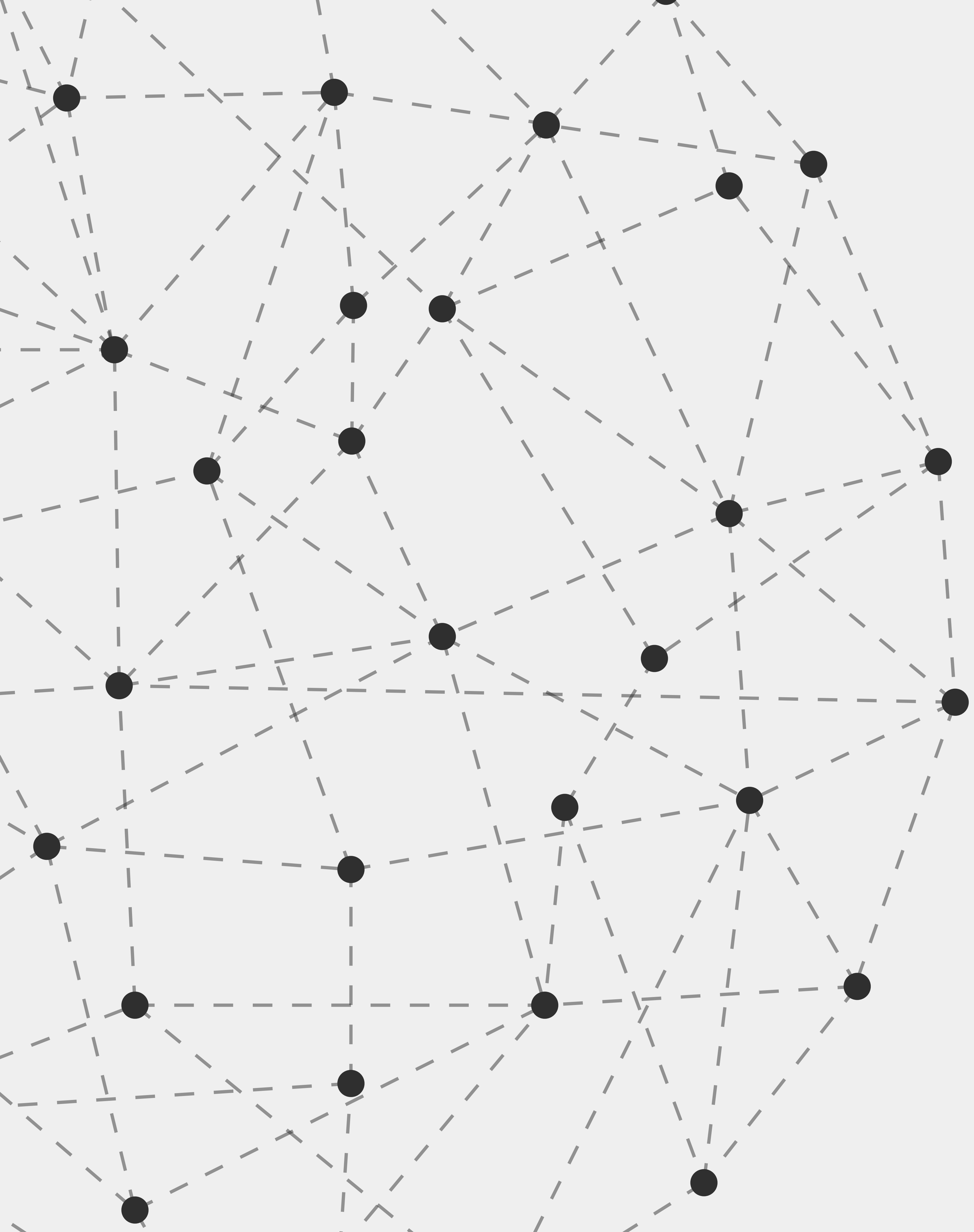


Decentralized Timeline

Diogo Maia - up201904974@fe.up.pt
Luis Viegas - up201904979@fe.up.pt
Mariana Monteiro - up202003480@fe.up.pt
Miguel Lopes - up201704590@fe.up.pt

Introduction

Project Specification and Technologies Used

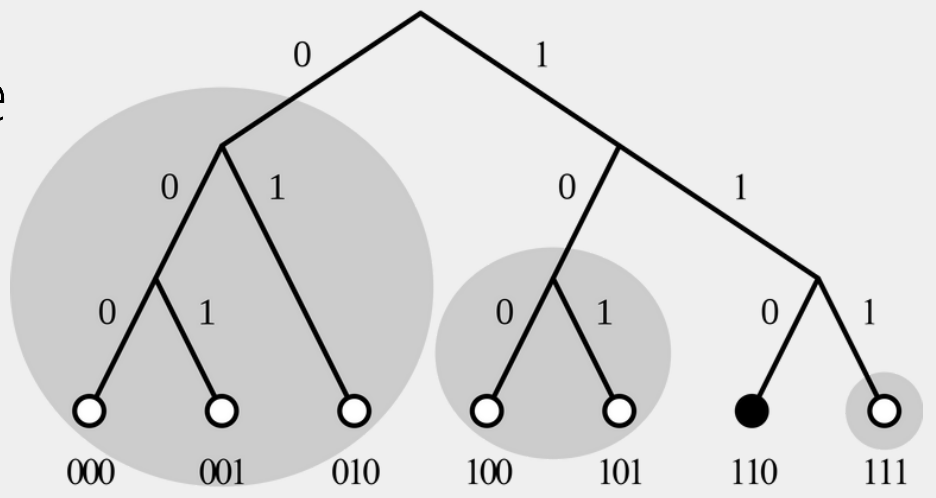
- The main objective of this project is to create a decentralized timeline service, similar to Twitter, Facebook or Instagram
- Kademlia DHT in Python to implement peer lookup and data storage in the network.
- In the front end, we use React.js.
- To connect the DHT node and the front-end, we use Flask

Basic Architecture

First Iteration

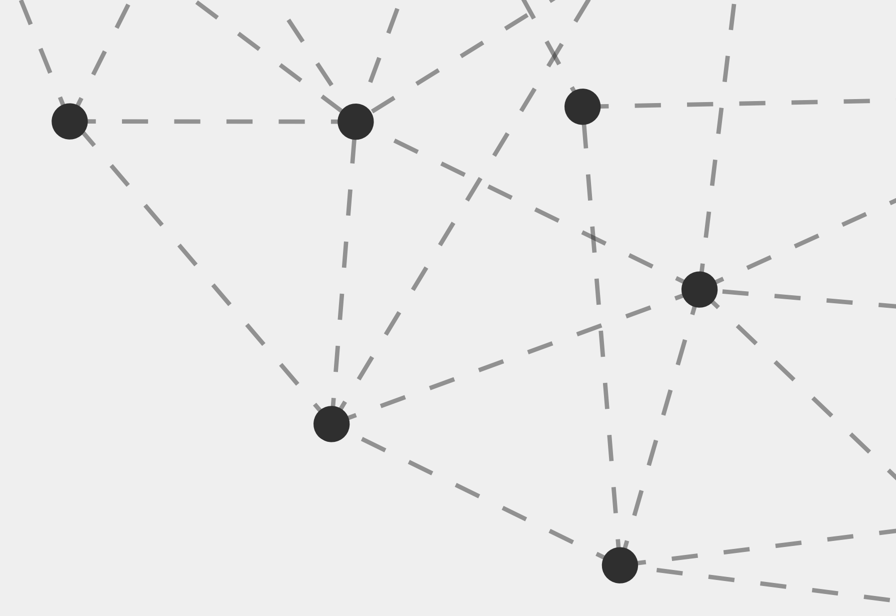
Communication Network

- The network is built on top of a Kademlia DHT
- All public information is stored in the DHT
- Main operations include:
  - Posting a message
  - Following / Unfollowing a user
  - Access your "Timeline", the place that combines the most recent posts of the users you follow



Issues Identified

- Lack of authentication means bad actors can pose as other users
- Authentication is crucial in any social network. Users want to know the content they are reading is from the person they follow and not just from a random individual posing as someone else
- Multiple users using the same username
- Needs a bootstrap node to work

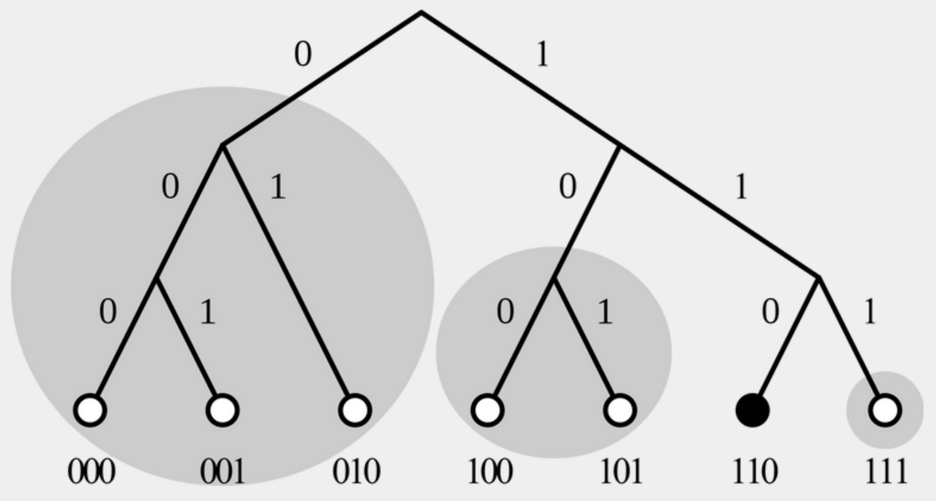


Basic Architecture

Second Iteration

Communication Network with RSA Signatures

- Same as the previous system, but with a RSA signature scheme
- Users sign all the content they post with their private key and their followers can verify the content's authenticity with the public key
- Users exchange public keys with each other directly so that no bad actor can tamper with the key



Issues Identified

- Public key exchange process requires both users to be online at the same time
- If a user loses their private key they lose access to their account
- Private keys are not easily memorizable (not very user friendly)
- No username control, so users can still use the same username to post
- Needs a bootstrap node to work

Basic Architecture

Final Solution

Communication Network with RSA Signatures

- The network is built on top of a Kademlia DHT
- All public information is stored in the DHT
- Main operations include:
  - Posting a message
  - Following / Unfollowing a user
  - Access your "Timeline", the place that combines the most recent posts of the users you follow
- Information is signed and verified using RSA cryptography
- Information that can't be verified is ignored

Authentication Network

- Network of servers that hold the users' keys
- A user chooses one server to store its keys
- Servers communicate with each other to share public keys and prevent users from creating multiple accounts with the same username
- The user sets up a password that allows them to login into the server and retrieve their private key (improves user experience)
- Users can ask the server for each others public keys
- Can act as the bootstrap node

Kademlia

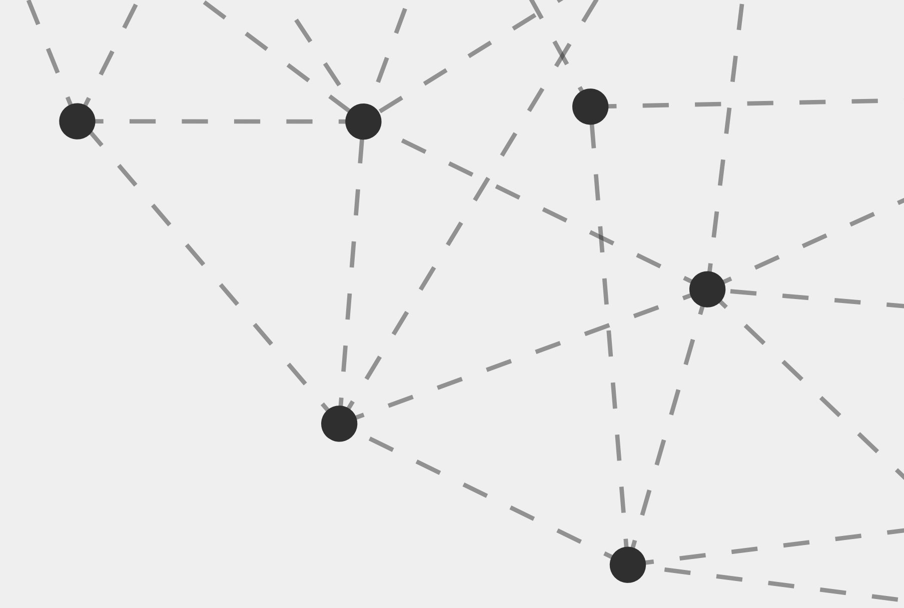
Overview and Implementation Details

Kademlia Overview

- Distributed hash table for decentralized peer-to-peer computer networks
- Nodes communicate among themselves using UDP.
- Each node has a unique ID and is responsible for storing values associated with a subset of the keys in the network
- If a node receives a request for a key that it is not responsible for, it will return the IDs and addresses of the nodes closest to the key. This allows the requesting node to continue routing the request until it reaches the correct node
- Kademlia also has mechanisms for maintaining the network, such as periodic pings to check if nodes are still alive and a process for repairing the routing table when nodes fail or leave the network

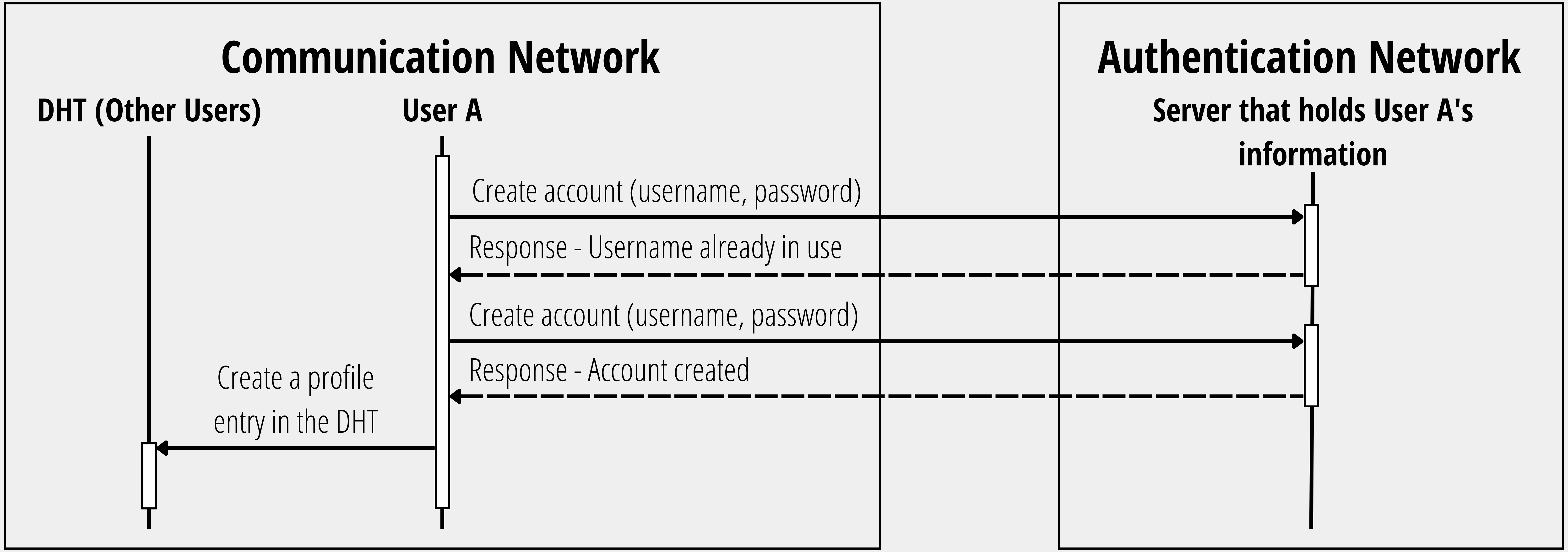
Implementation Details

- Used a Python implementation of Kademlia that aims to be as close to a reference implementation of the Kademlia paper as possible.
- We used the hash of the tweets as their keys, but for the user accounts we used the username as the key



Sequence Diagram

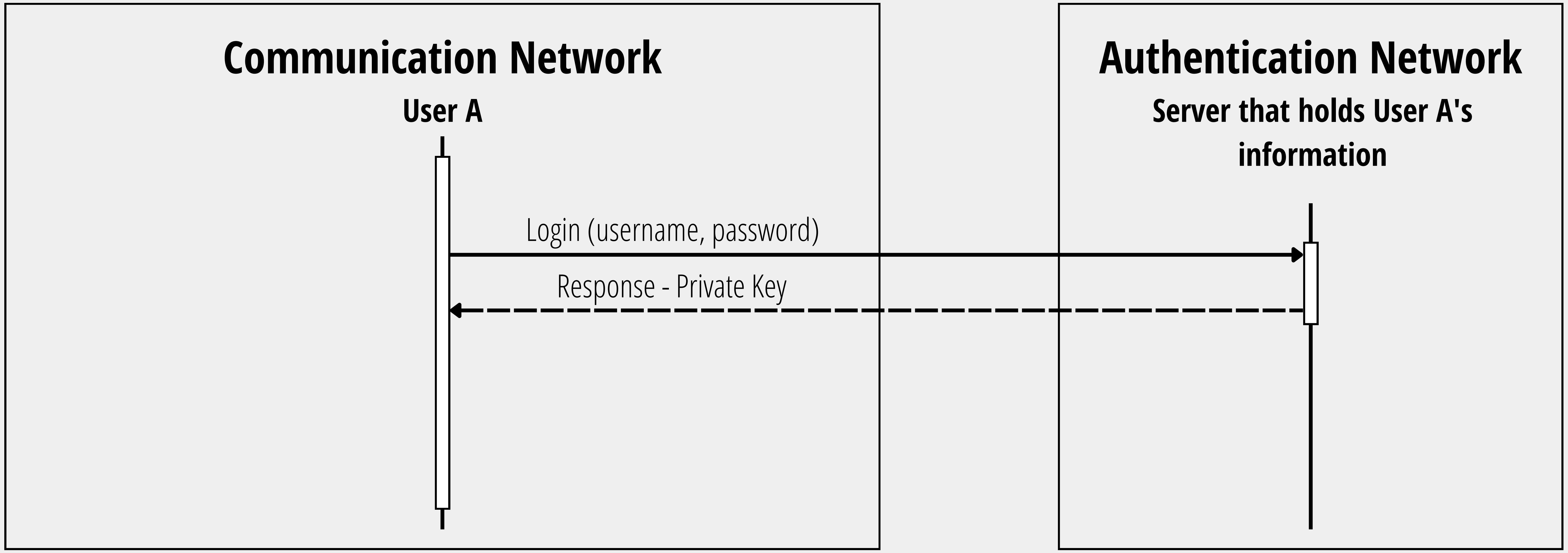
Create Account


Communication Network
DHT (Other Users)
User A
Authentication Network
Server that holds User A's information
Create account (username, password)
Response - Username already in use
Create account (username, password)
Response - Account created
Create a profile entry in the DHT

Note: After following User X, User A stores User X's Public Key locally, hence reducing the dependency on the Authentication Network

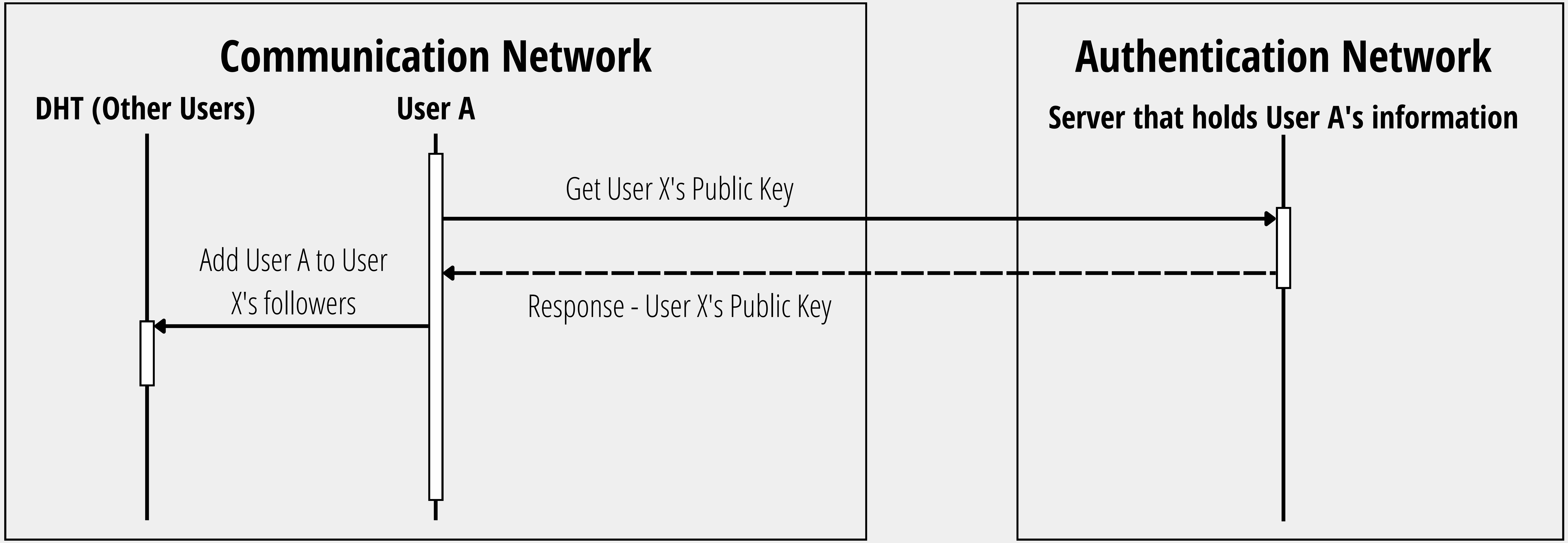
Sequence Diagram

Login (Private Key Retrieval)



Sequence Diagram

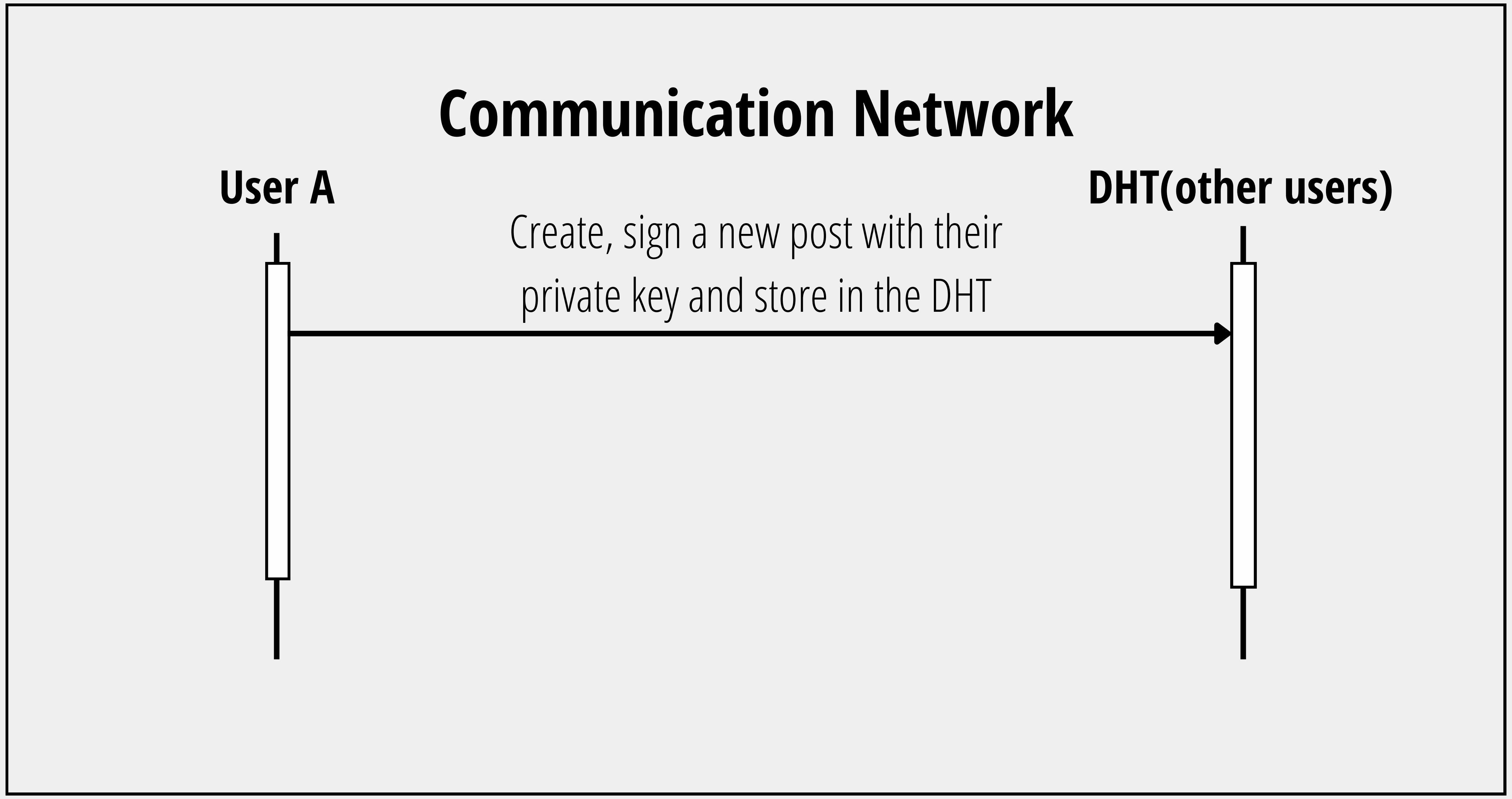
Follow User


Communication Network
DHT (Other Users)
User A
Authentication Network
Server that holds User A's information
Get User X's Public Key
Response - User X's Public Key
Add User A to User X's followers

Note: After following User X, User A stores User X's Public Key locally, hence reducing the dependency on the Authentication Network

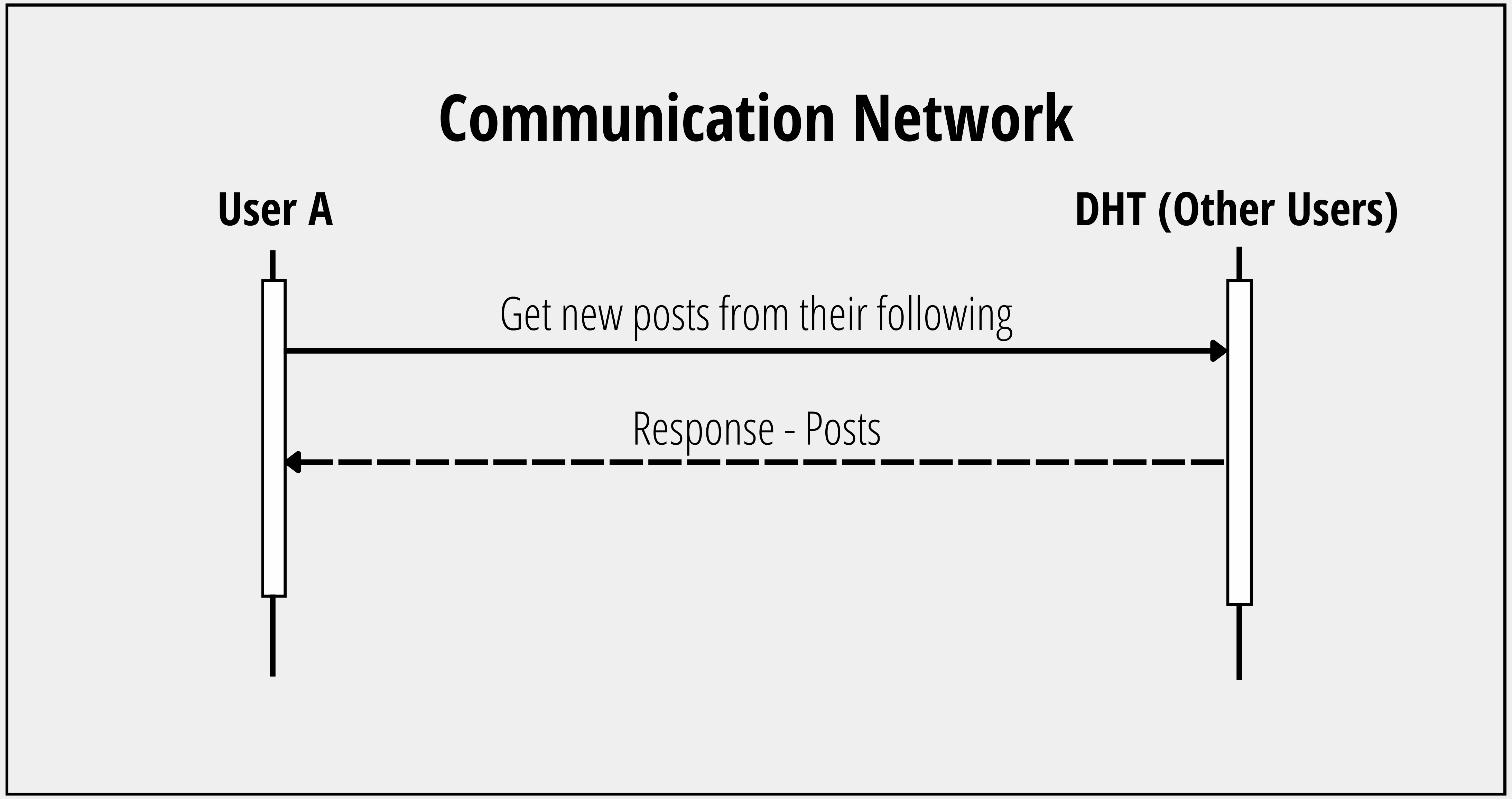
Sequence Diagram

Create a Post


Communication Network
User A
DHT(other users)
Create, sign a new post with their private key and store in the DHT

Sequence Diagram

Create Timeline / Get Posts



Note: The authenticity of each post is verified locally by the User when they retrieve the post

Conclusion and Future Work

Conclusions

- Decentralized systems present a set of challenges:
  - Trust-related issues
  - Equal storage distribution

Future Work

- Improve the user interface
- Implement other functionalities, like reposting (retweeting)
- Implement tweet deletion