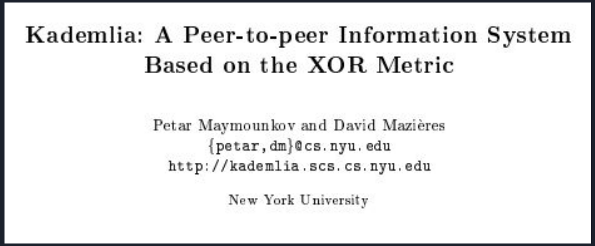
SDLE 2022/2023 DECENTRALIZED TIMELINE

Group T3G13

Tiago Antunes
Tomás Fidalgo
Vasco Alves

Technologies

- Python
  - Kademlia
  - ZeroMQ
  - Asyncio and Threading



Kademlia: A Peer-to-peer Information System Based on the XOR Metric

Petar Maymounkov and David Mazières
{petar,dm}@cs.nyu.edu
http://kademlia.scs.cs.nyu.edu

New York University

Functionalities

- Register/Login
  - Create a new account with username and password
  - Login with existing username and correct password
- Write a post
  - Create a post and send it to your followers
  - Posts are stored in permanent memory, so they can be recovered by the user after a logout
- Follow and unfollow user
  - Add/remove their posts to your timeline
- Show user’s timeline
  - See your own posts
- Show complete timeline
  - See both your own and other users’ posts
- Show followers
  - See your followers
- Show following
  - See who you are following

DHT

- We used Kademlia DHT for the network
- Network is started with a bootstrap node
- New nodes are connected to bootstrap
- Network stores information about the users
- We assume bootstrap node is always running

DHT

Information saved on kademlia using username as key:

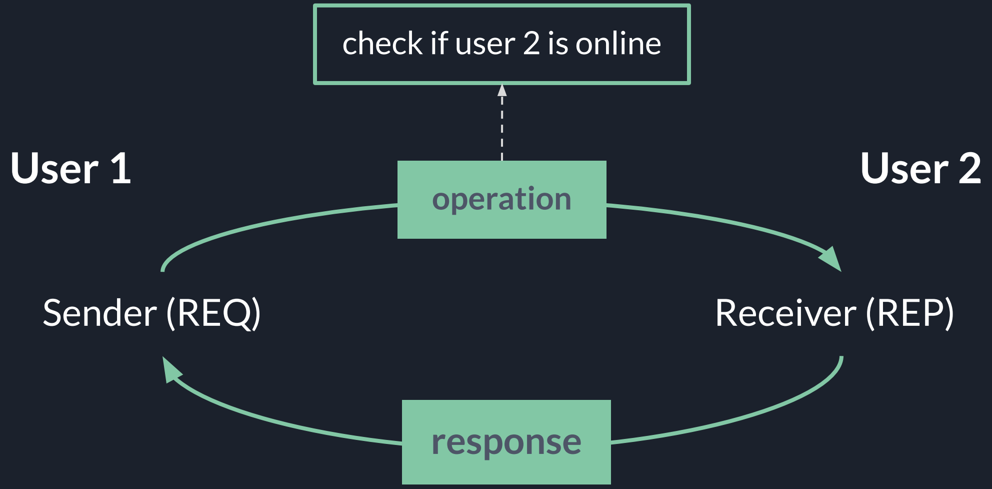
- ip
- port
- followers
- following

Operations:

- set(username, information) - Each node only sets their own information, not others’
- get(username) -> information

Communication

The communication was implemented using ZeroMQ REQ-REP sockets with the Lazy Pirate Pattern, where each node has a sender and a receiver:

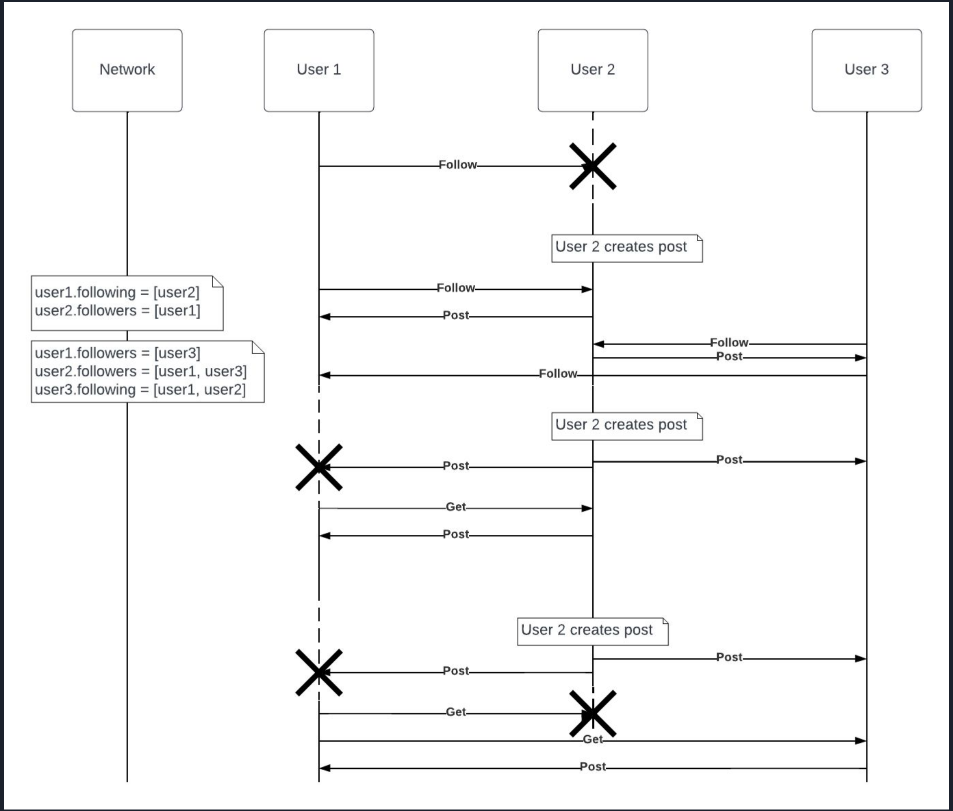


Communication

Types of messages:

- follow <sender_username>
  - Sender sends message to receiver
  - If receiver is offline it is not possible to follow
  - If receiver is online they send their posts and the information on both nodes is updated
- unfollow <sender_username>
  - Sender sends message to receiver
  - If receiver is offline it is not possible to unfollow
  - If receiver is online their posts are removed from sender's timeline and the information on both nodes is updated
- post <full_post>
  - Message sent to followers when user makes a new post or is asked for posts
  - If the follower is online, they receive the post and add it to the timeline
- get <requester owner>
  - When user logs back in this message is sent to all the users they are following to get new posts
  - If the owners are online they answer to requester with their posts
  - If the owner is offline, requester asks his followers who follow the owner to send the posts

Possible Scenario



[illegible]

Conclusion and Reflections

- The service satisfies the user and project's needs
- The bootstrap node needs to be running at all times
- Since timelines aren't stored locally, a user can lose the timeline of a user they follow if no one can provide him the posts after a login, which is something to improve (We lose the timeline of the user we follow, they still have their own posts, but no one can send them to us)
- We would like to improve the user interface
- We would like to find a better way to handle follow/unfollow operations when users are offline