Class 3, Group 13

Diogo Samuel Fernandes
Hugo Guimarães
Paulo Ribeiro
Telmo Baptista

Large Scale Distributed Systems

Supervised by
Carlos Baquero-Moreno
Pedro Ferreira Souto

Camellia Social Network

Peer-to-peer network using Kademlia

Index

Technologies

- Python
  - Simple Term Menu
  - Asyncio
  - ZeroMQ (with Tornado)
  - Kademlia
  - NTPLib
  - PyCryptodome

Functionalities

- Register / Login / Logout
  - Create a new account or login with an existing one, and logout from the system
- Follow / Unfollow
  - Choose to view all of another user's posts in their timeline
- Post Message
  - Publish a new message that becomes available to all your followers
- View Timeline
  - See your timeline, composed of the posts of the users that you follow and your own posts
- View Profile
  - See your own profile, with information about the people you follow, your followers and how many posts you have made
- Recommendation System
  - Get suggestions of users based on people you

Kademlia Network

Kademlia is a distributed hash table for decentralized peer-to-peer computer networks.
In order to enter the network, a node must connect to a currently active node.

- Every node in our network can be considered a bootstrap node.
- It is assumed that our network always has at least 2 active nodes
- The network stores information regarding its users, except their timeline and associated posts, which are stored locally

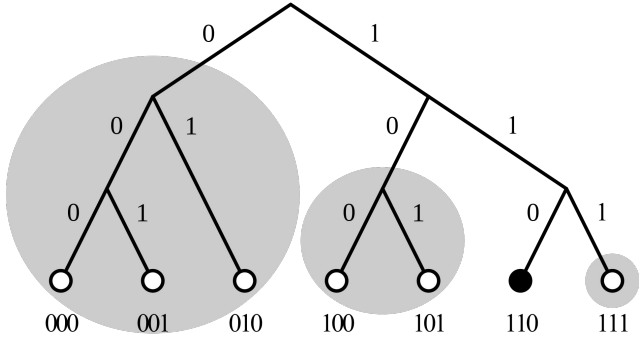


Fig. 1 - Example of a Kademlia Network

Information Stored on Kademlia

Private

username:private

salt
hash_password

public

username:public

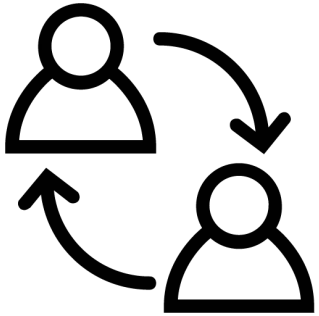
followers
following

connections

username:connections

public_key_n
public_key_e
listening_ip
listening_port

Send and receive posts



Direct communication is needed in the following situations:

- When a user follows another, the followed user must send its posts to the new follower.
- When the followed user is offline, the follower must try to reach his followers in order to fetch its posts.
- When a new post is made, its author must send this post to all of their followers.
- After rejoining the network, an user must send all its posts to their followers.

Thi communication was implemented using ZeroMQ sockets, in a PUSH-PULL pattern, so that every time a message is published, the follower receives it without having to perform any request.
ZeroMQ Tornado was used to provide non-blocking I/O.

Security

In order to guarantee a secure network, some safety measures were implemented

- Passwords are hashed using a random salt. Although these fields are public because they are stored in Kademlia, using salt is a powerful tool since passwords like “123456” or “password” aren’t immediately revealed when a hashed password is identified.
- Upon account creation, every user is given a private and public key. The public key will be stored on kademlia network as seen before, and the private key will be store inside a folder, on user machine. The user will use the private key to sign every message sent through direct connection, ensuring the integrity of the message.

Send and Receive Posts

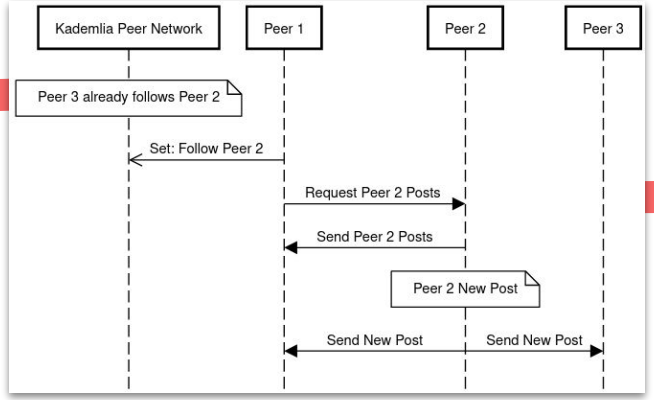


Fig. 3 - Normal flow off a follow event

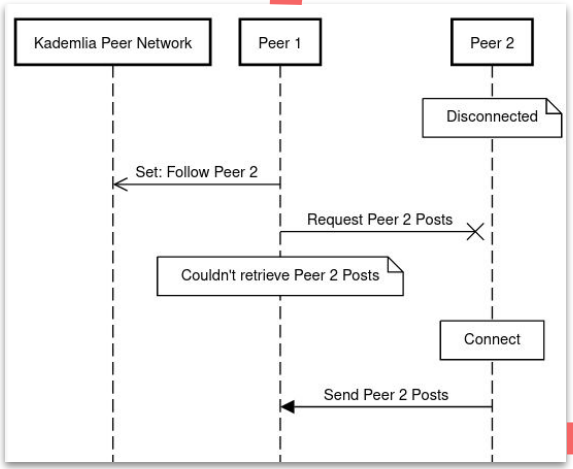


Fig. 4 - Peers send updated timeline to its followers on connection

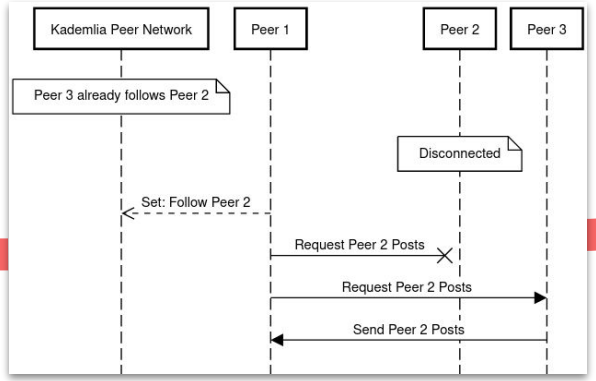
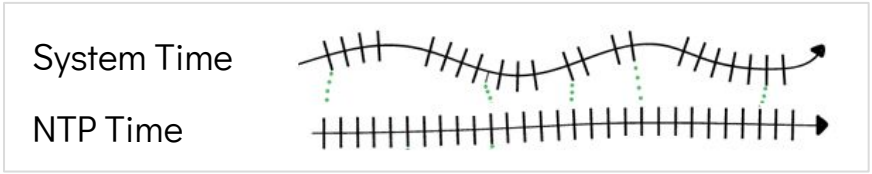


Fig. 5 - Request updates to the followers of the request's original target

Order of Posts

Since Camellia is a Timeline App, we need to ensure that posts follow a causal order. At first, we started to search of ways doing this, like vector clocks and interval tree clocks. However, these alternatives had too much overhead so we started searching for others.

To ensure the causality of posts, Twitter uses SnowflakeID. The first 41 bits of this ID are a timestamp. The following bits represent a machine ID, being followed by the sequence number of message, to allow creation of multiple messages in the same millisecond.



To ensure the order of posts, we used Network Time Protocol to synchronize the clocks of the each peer once a hour.

Handling disconnected users

As soon as a user rejoins the network, it requests the posts from the followed users:

- First, it tries to reach the followed user. If they are online, the posts are sent successfully through a direct connection.
- If the followed user is offline, the request starts being sent to each of the other followers of the followed user, until one of them retrieves the required posts.

The same strategy is applied when an user follows another and immediately requests their posts to update their timeline.
After rejoining, they also send their posts to each of their followers:

- This way, we guarantee that the followers are always updated, even if the previous strategy failed (for example, when the followed user is offline and has no other followers that we could reach).

Garbage Collector

In order to remove old messages that have lost interest. a garbage collector was implemented.

- The garbage collector allows the removal of posts that have been published over a certain period of time (post lifespan)
- The default value used for the user post lifespan is 3 minutes.
- The lifespanof a post can be changed by each user in the config.ini file, located in the root of the project.
- The garbage collector is called whenever a user logs into its account, and every minute afterwards.

Conclusion and Future Work

During the development of this peer to peer network we learnt how to build a reliable large scale network timeline.

Due tue time constraints, we couldn't add more features to this network. However in the future we plan to add more restrictions to the namespaces. For example, each namespace could have a difference scope. The namespace private could be encrypted and then it will be only accessed by that user. Also, the connections table should only be setted by that user, but retrieved by everyone.

We could also improve the suggestion user in case that user don't have follower by searching neighbors of current node on Kademlia and the visualization of other user profiles

References

- Accessed December 2021. https://github.com/bmuller/kademlia
- Accessed December 2021. https://github.com/cf-natali/ntplib/
- Accessed December 2021. https://zeromq.org/
- Accessed December 2021. https://en.wikipedia.org/wiki/Snowflake_ID
- Accessed January 2021. https://pycryptodome.readthedocs.io/
- Petar Maymounkov and David Mazières 2002. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf