P2P-Decentralized Timeline

T4-Group 13

Francisco Goncalves (up20174790@edu.fe.up.pt)

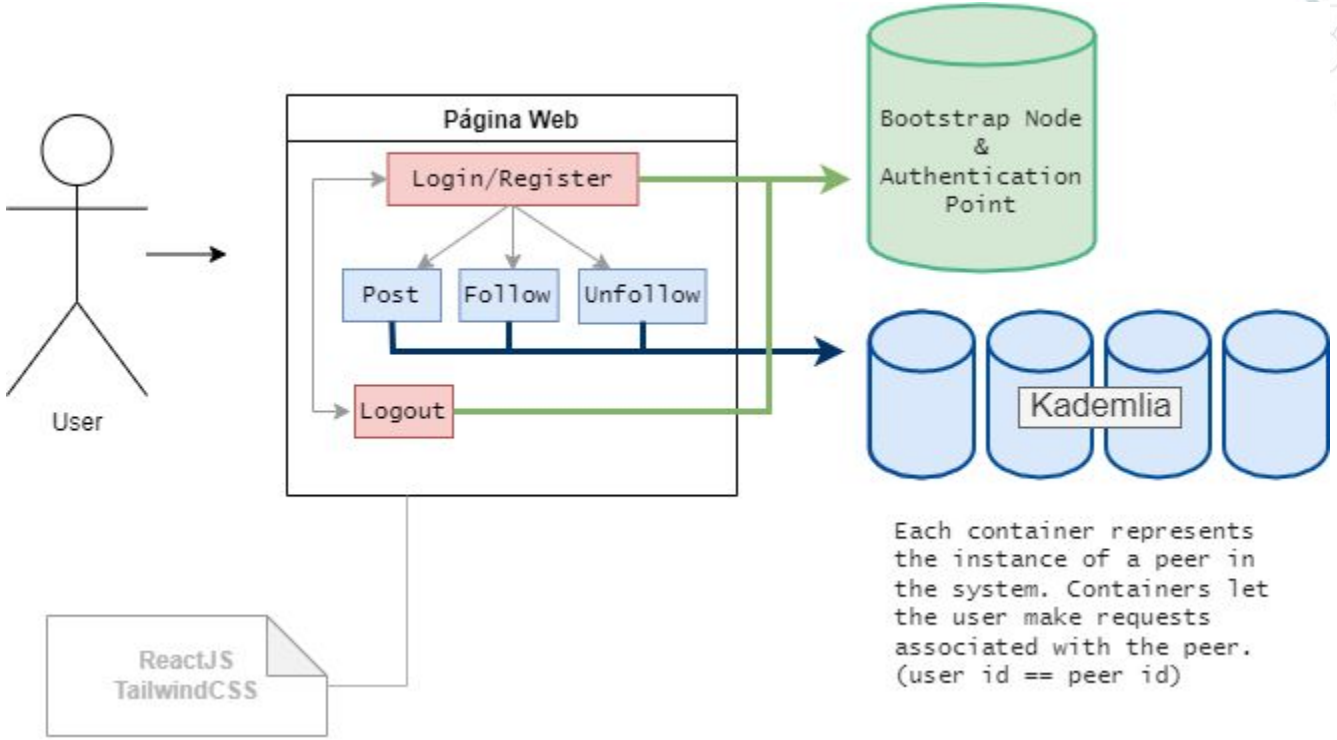
Inês Quarteu (up201806279@edu.fe.up.pt)

Jéssica Nascimento (up201806723@edu.fe.up.pt)

Rúben Almeida (up201704618@edu.fe.up.pt)

SDLE 2021/2022 T4G12

Architecture

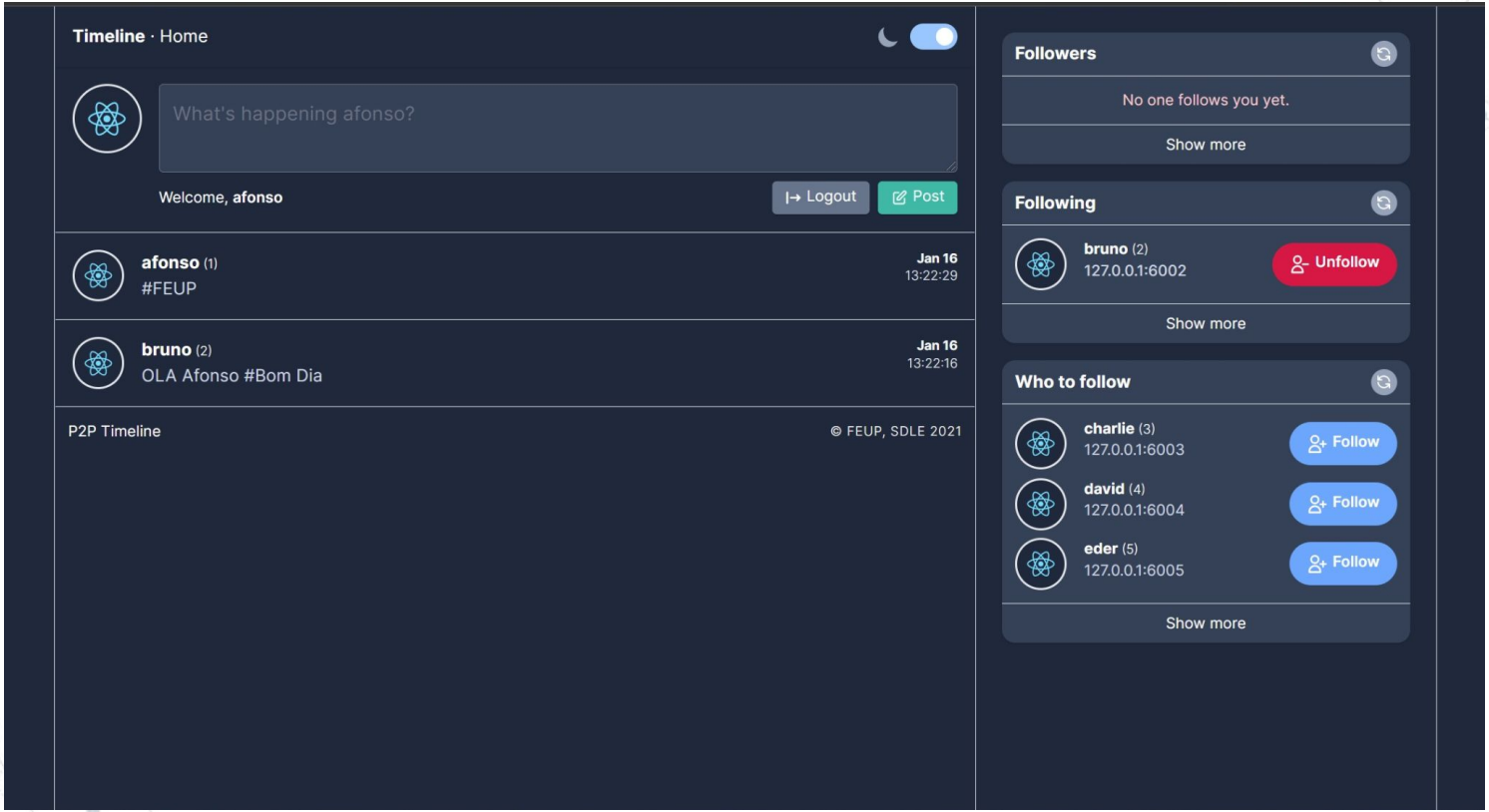


Kademlia

- DHT System. Uses XORs based distances to crawl the Network.
- MIT licensed library by @bmuller with 702 stars in Github.
- Built on top of Python asyncio library.
- Introduces a concept of a Bootstrap Node.
- Inserts Centralization.😢
- Exposes an API:
  - **async get(key): DHT_Entry**
  - **async set(key)**
  - **bootstrap(ip, port)**

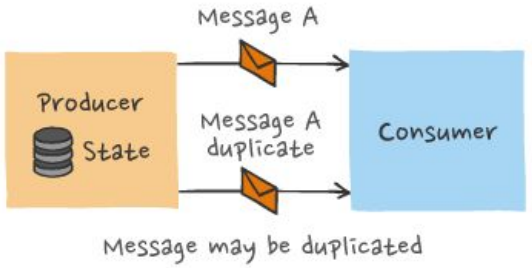
```python
class DHTEntry:

    def __init__(self, identifier: int, ip_address: str, port: int):
        self.identifier = identifier
        self.ip_address = ip_address
        self.port = port
```

FRONTEND


Timeline · Home
What's happening afonso?
Welcome, afonso
Logout
Post
afonso (1)
#FEUP
Jan 16
13:22:29
bruno (2)
OLA Afonso #Bom Dia
Jan 16
13:22:16
P2P Timeline
© FEUP, SDLE 2021
Followers
No one follows you yet.
Show more
Following
bruno (2)
127.0.0.1:6002
Unfollow
Show more
Who to follow
charlie (3)
127.0.0.1:6003
Follow
david (4)
127.0.0.1:6004
Follow
eder (5)
127.0.0.1:6005
Follow
Show more

Protocol (I/)

- Gossiping Techniques.
- Mainly Push Gossiping -> Single Round Trip😎
- Use of Zeromq REQ-REP TCP Sockets.
- At Least Once Semantics.
- **Assumption:**
  - All subscribers receive the same content with a similar delay.
  - There aren't followers with different stages of an external timeline.



Protocol (II/)

- **PING**
  - Periodic Ping towards know neighbours.
  - Periodic Ping towards followings(Special type of neighbour).
  - Ensure they are alive
- **SHARE_ENTITY**
  - Inspired by genetic algorithms philosophy of crossover.
  - Send random neighbour(A) towards random neighbour(B).
  - A != B.
  - In each repetition are chosen 3 elements to compose the A group.
- Tested in our local machine in a 50 peers network and it was enough to converge. It took less than 3 gossiping rounds to converge.

Protocol (II/)

- **SUBSCRIBE:**
  - We can only subscribe to a peer that is already known, one neighbour.
  - Starter peer sends a direct request towards destination.
  - Peer receives this message and includes it in the list of followers.
- **UNSUBSCRIBE**
  - Starter peer sends a direct message towards following.
  - Peer receives these messages and removes this follower from its list of followers.
- **TIMELINE**
  - A periodic push of my timeline towards my followers.
- **USERS**
  - Bootstrap result.
  - A list of five online followers to start the p2p protocol.

Protocol (III/)

- **SUBSCRIBE_INDIRECT:**
  - Pull Gossiping for Data Retrieve When destination peer is dead(zombie).
  - Protected with Time to Live field to prevent loops / Zombies.
  - Includes the desired follower.
  - Peer process this message either returns the content in its possession (SUBSCRIBE_INDIRECT_ACK) or forwards the message.
- **SUBSCRIBE_INDIRECT_ACK:**
  - Reply to a SUBSCRIBE_INDIRECT
  - Includes the concept of indirection identifier to keep track of who was the third party

Entity Liveness | Entity Authentication

Liveness:

- All messages are protected by timeouts.
- This system isn’t a silver bullet.
- Periodic Pings allow recovering from false positives.

Authentication:

- Login and Register Functionality.
- Users are stored in persistent storage.
- Bootstrap node condenses authentication tasks.
- After successful authentication, the peer bootstraps.
- The result of peer bootstrap is the retrieve of 5 initials entities to start the decentralized message exchange

Entity Recognition

How can peers know each other?

- Requirements:
  - Scalability.
  - P2P approach.
  - Retrieve all peers.
- Hypothesis Tested:
  - Rely on DHT Content Crawling -> Too much network activity
  - Rely Fully on Authentication Server -> Centralized Solution

Final Solution:

- Bootstrap with N peers + Periodic Pings.

Message Sequencing: NTP

Initial Implementation:

- A request to a network time protocol (NTP) server is made each time a message is published.

Final Implementation:

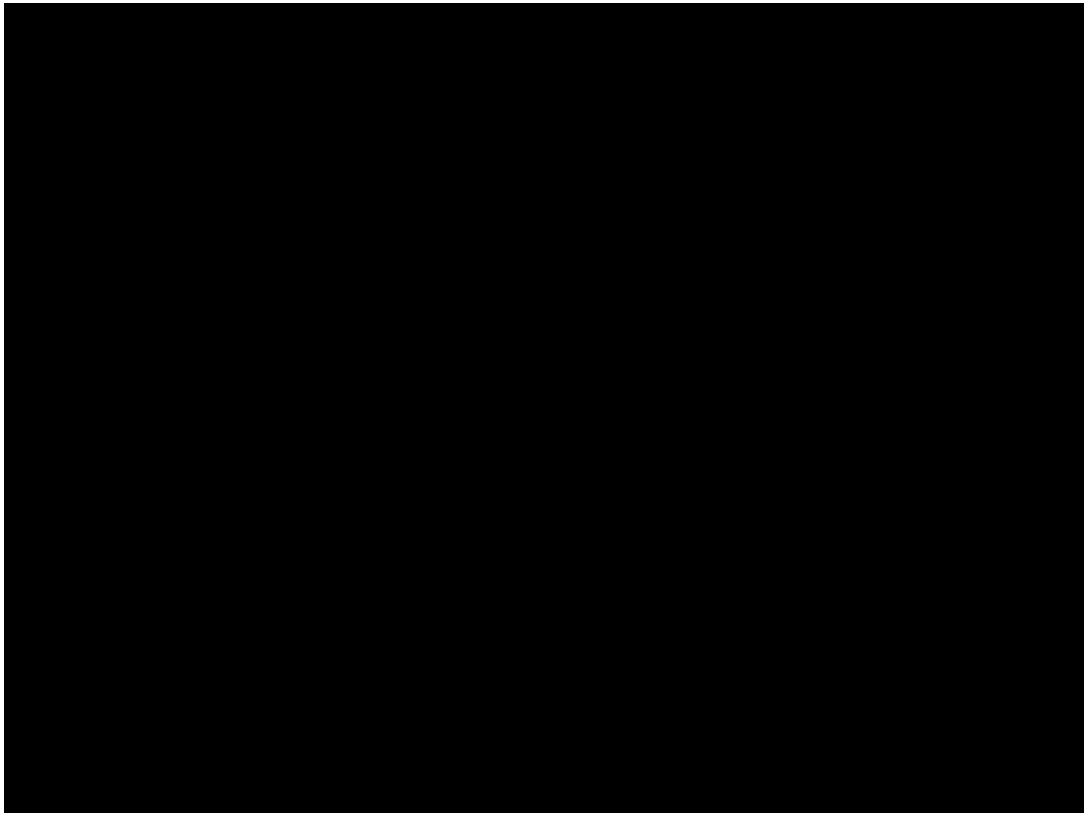
- A peer keeps a value that represents the delay between its internal clock and an NTP server. It updates the value only after a certain interval of time has passed.

Server used: pt.pool.ntp.org

Ephemeral storage of messages

- Each post has a timestamp that represents when a peer received it.
- Each peer has a clock synchronizing with the NTP server.
- If a peer has a post for a certain amount of time, the post will be deleted.

Demonstração



PERGUNTAS?