Decentralized Timeline

A Decentralized and Distributed Timeline service

Miguel B. Rodrigues Rita M. Mendes Tiago C. Silva
Tiago P. B. Rodrigues

Faculty of Enginnering
University of Porto

December 2022

Outline

Outline

The Problem

- By this time, society has already realised the power that social networks have in shaping the world we live in.

The Problem

- By this time, society has already realised the power that social networks have in shaping the world we live in.
- Nevertheless, most of these social networks operate in a **centralized** paradigm.

The Problem

- By this time, society has already realised the power that social networks have in shaping the world we live in.
- Nevertheless, most of these social networks operate in a **centralized** paradigm.
- This, obviously, raises concerns regarding **privacy**, **security** or **freedom**, as one entity rules all the others.
- Most of the current platforms have **single points of failure** that, in case of failure, cause DoS to the other connected entities.

The *Possible* Solution

- The solution to this is to use a **decentralized** service.

The *Possible* Solution

- The solution to this is to use a **decentralized** service.
- Such services, usually P2P, can overcome the obstacles imposed by the centralized counterpart.
- They are much more **robust** to DoS attacks and **fault tolerant** if particular criteria is met.

The *Possible* Solution

- The solution to this is to use a **decentralized** service.
- Such services, usually P2P, can overcome the obstacles imposed by the centralized counterpart.
- They are much more **robust** to DoS attacks and **fault tolerant** if particular criteria is met.

A problem yet to be solved

Even though, P2P netwroks is decades old already, *crypto* brought them back to the mainstream. Sadly, the fact is that a large scale timeline service, like Twitter, which is **fully decentralized** is something **missing** in distributed systems field.

Outline

Technological Vision

- Technology choice was free.
- We have tried to keep it as clean and simple as possible, while using what is the latest and greatest.

Tools

- We used Node.js as the runtime environment.
  - npm provides lots of helpful packages.
- Together with TypeScript as the driving programming language - mainly to ensure type safety.

Libraries and Tools

1. libp2p
    - Peer Discovery
    - Content Routing
2. express
    - RESTful API
3. jose
    - Authentication
    - Signatures
4. node-cache
    - Caching
5. Vue.js
    - UI Prototyping

Outline

Main Functionalities[1]

1. Fully decentralized service within a local network.
2. Registration and Authentication performed through asymmetric cryptography, i.e. signatures.
3. Posts are small text written and published by the users.
4. Users can follow other users and like or repost their posts, similar to what happens in Twitter.
5. Posts might include topics, e.g like Twitter hashtags, which users can then group or filter.
6. Users can view their own timeline, their peers' timelines or just mix them all!

[1] Short demo of the developed service.

Challenges

1. The very first problem in P2P systems development is the **discovery** problem. How does one node find and connect to another node in the network?

Challenges

1. The very first problem in P2P systems development is the **discovery** problem. How does one node find and connect to another node in the network?
2. Then there is the need to distribute the content across multiple peers, i.e. **content routing**.

Challenges

1. The very first problem in P2P systems development is the **discovery** problem. How does one node find and connect to another node in the network?
2. Then there is the need to distribute the content across multiple peers, i.e. **content routing**.
3. However, to make content routing function, peers must distinguish themselves - **identities**.

Challenges

1. The very first problem in P2P systems development is the **discovery** problem. How does one node find and connect to another node in the network?
2. Then there is the need to distribute the content across multiple peers, i.e. **content routing**.
3. However, to make content routing function, peers must distinguish themselves - **identities**.
4. How will the user be able to interact with the service? It is necessary to implement an easy-to-use **interface** that meets the service requirements.

Solutions

- Thankfully, libp2p is able to deal with peer identity, peer discovery and content routing under the hood.
  - Peer identity are already is already baked into the library.
  - Peer discovery is done using MulticastDNS. It has the plus fully decentralization, but only works in local networks.
  - DHT implementation is based on *kademlia* with some slight modifications, e.g. SHA-256 instead of SHA-1.

Solutions

- Thankfully, libp2p is able to deal with peer identity, peer discovery and content routing under the hood.
  - Peer identity are already is already baked into the library.
  - Peer discovery is done using MulticastDNS. It has the plus fully decentralization, but only works in local networks.
  - DHT implementation is based on *kademlia* with some slight modifications, e.g. SHA-256 instead of SHA-1.
- The interaction with the service is done using a RESTful API.
  - Front-end is responsible attach itself to the back-end. This way there is a beneficial separation of concerns. For instance, users have the ability to customize the front-end.

Design Visualization

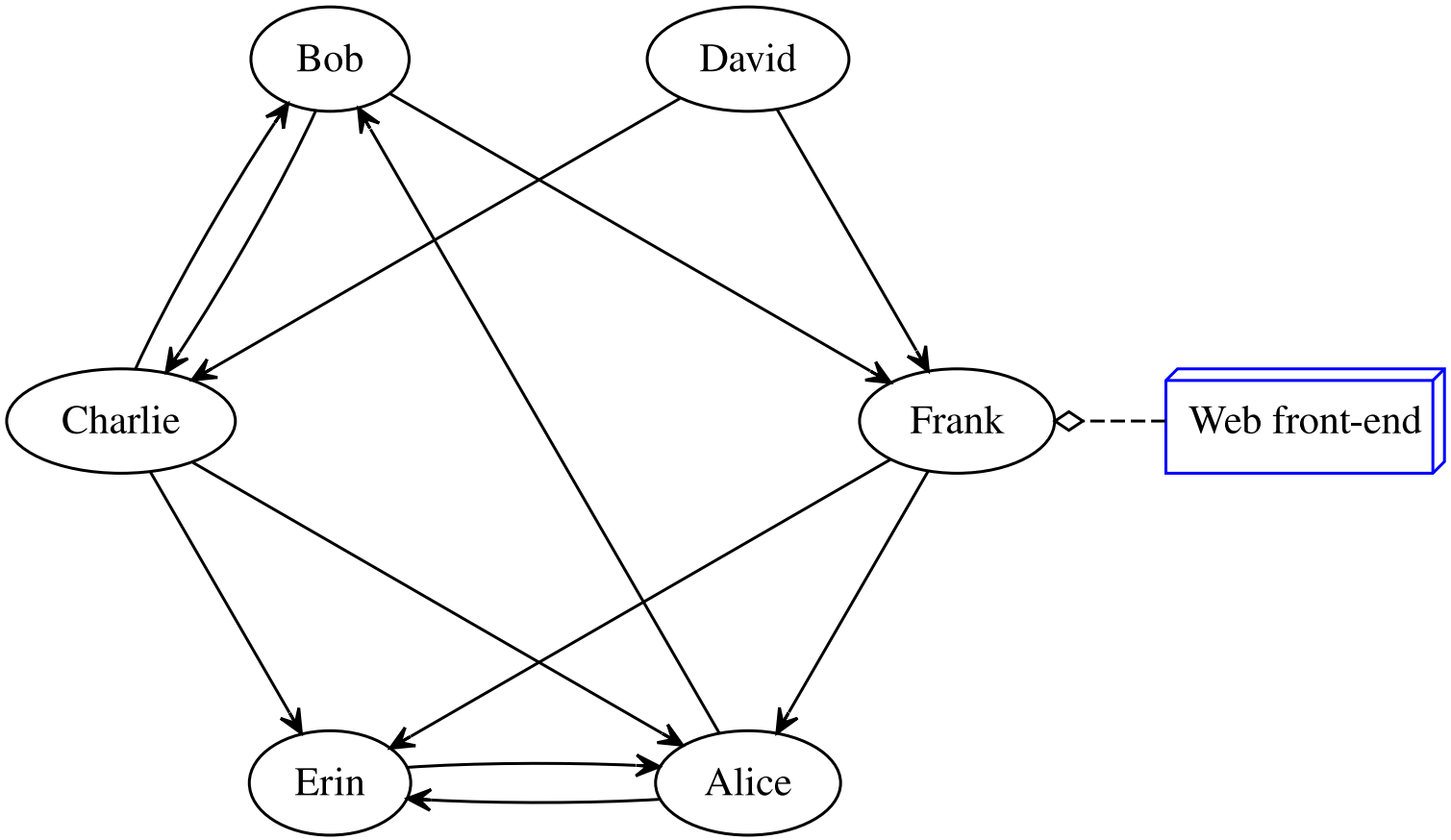

Figure: Timeline service's network backbone

Outline

A *Constant* Work in Progress

- We are aware that our implementation is neither ideal or perfect. There is room for adjustments and improvements.

A *Constant* Work in Progress

- We are aware that our implementation is neither ideal or perfect. There is room for adjustments and improvements.
- Since keeping the system fully decentralized is a top priority, it seems obvious to us that a big improvements is to make the service work in a wider network, despite of a series of use cases inside LANs.

NAT Transversal

- On top of that we must take into account that most of modern day internet infrastructure is behind NATs, in part because of the short IPv4 address space.

[2] libp2p hole punching by Max Iden @ IPFS

NAT Transversal

- On top of that we must take into account that most of modern day internet infrastructure is behind NATs, in part because of the short IPv4 address space.
- Yet this can be solved in a variety ways:
  1. Ensure that at least one machine behind the NAT is executing the service. Then it can work as a *rendezvous* point. More straight forward but might require manual intervention.
  2. Hole punching[2]. Harder to get done right but easier to use!

[2] libp2p hole punching by Max Iden @ IPFS