HELIOS

A Context-aware Distributed Social Networking Framework

On DHT, libp2p and HELIOS Platform

June 15, 2021

Tommi Meskanen, Valtteri Niemi
University of Helsinki

Links

Youtube video:

https://www.youtube.com/watch?v=qN-OChfUPP4

Conference paper:

https://fruct.org/publications/fruct27/files/Mes.pdf

Peer identities and related concepts in libp2p

PeerId (or peer id) is a randomly generated identity for the node. This is needed before the node can join the p2p network or the p2p overlay. The id can be used persistently or it can be ephemeral.

PeerId is a unique identifier for a node and it is the verifiable link between the public key and the node. A human readable presentation of PeerId in libp2p can be a base58 encoded value of the public key.

PeerInfo combines the identity of a peer with its known addresses that may be public/accessible IP addresses or connection relay addresses.

The identify protocol provided by libp2p allows nodes to exchange information e.g. their public keys and known network addresses.

DHT privacy issues

At least some actions in a public DHT are visible to other nodes, since all connection requests between nodes are visible.

Various DHT implementations are available, and they may suffer from different conceptual/security weaknesses. DHTs can be misused in many ways, they can be flooded with data, they can suffer from traditional DDoS attacks, and so on.

Routing tables tell the path from the requestor to the provider of the content, or actually paths to providers.

Nodes may have stored different values with the same key in situations where this key is updated by multiple nodes at the same time.

DHT security issues

There are some basic attack scenarios against p2p/DHT systems:

- Sybil attacks, where new identities are created to overwhelm the network.
- Eclipse attacks, where target nodes of the network are eclipsed/denied of service or provided with false data.

One could create a private DHT, requiring a shared key to join it, for the set of users that are allowed to share the information to tackle some of the problems. Such a private DHT does not differ that much from storing the information directly in the nodes. In addition, removal of users from the group can be directly noticed by the remaining users.

Privacy issues

Potentially sensitive information includes:

1. What is user's IP address?
2. Friendship/contact details of the user?
3. Who are user's contacts?
4. How often are specific users in contact?
5. When does the user connect to the (p2p) network?
6. What information/content the user is interested in?
7. What information/content the user provides of themself?

Distributed Hash Table

- DHT is a lookup table that is distributed between several nodes.
- The DHT stores (key, value) pairs. The key is the hash value of information that is used to determine the location for storing the value.
- Each node in the DHT stores values for some range of keys.
- Typically the value is stored to several nodes.

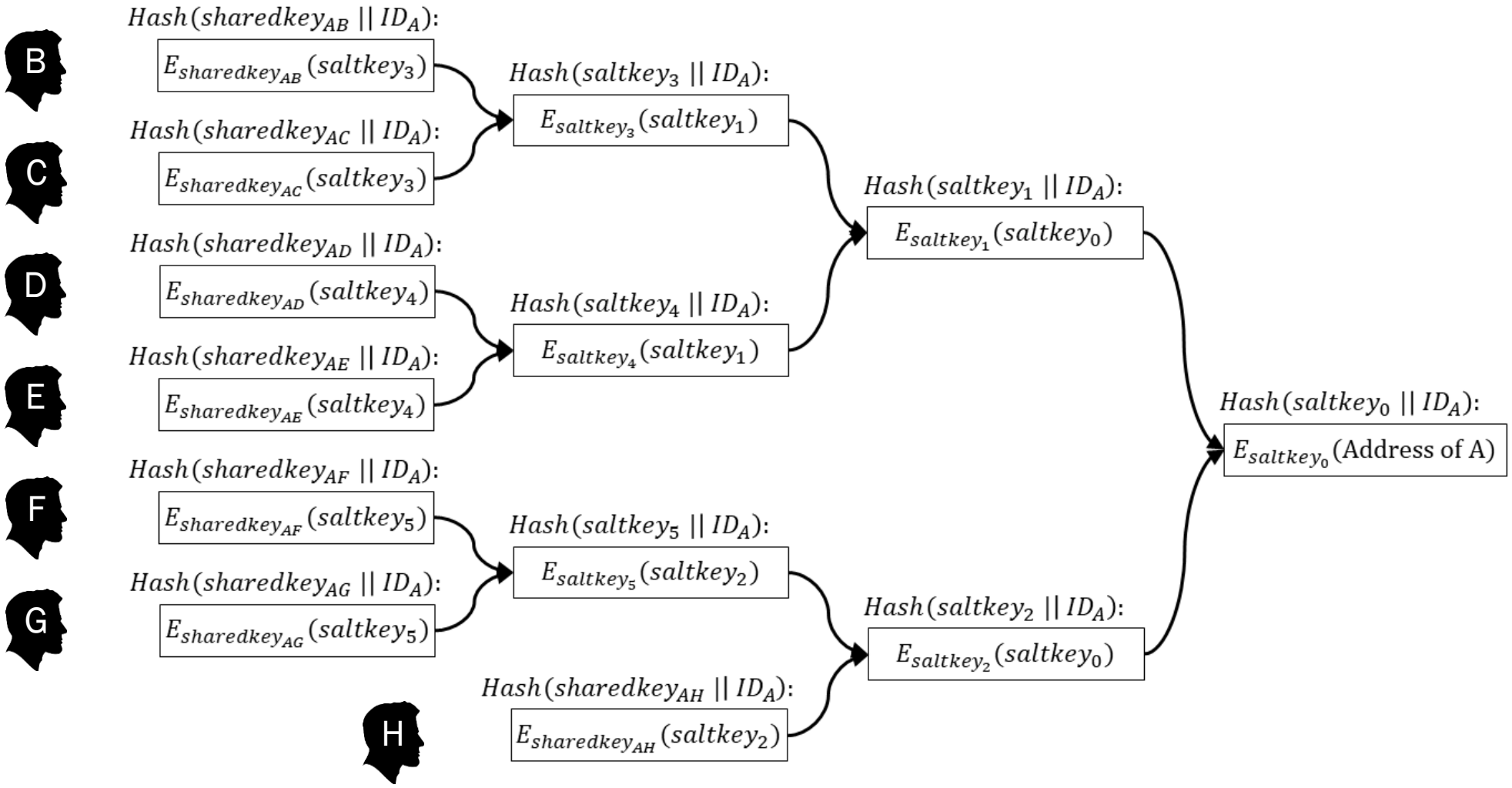
We use the DHT in a such a way that

- key is secret
- value is encrypted

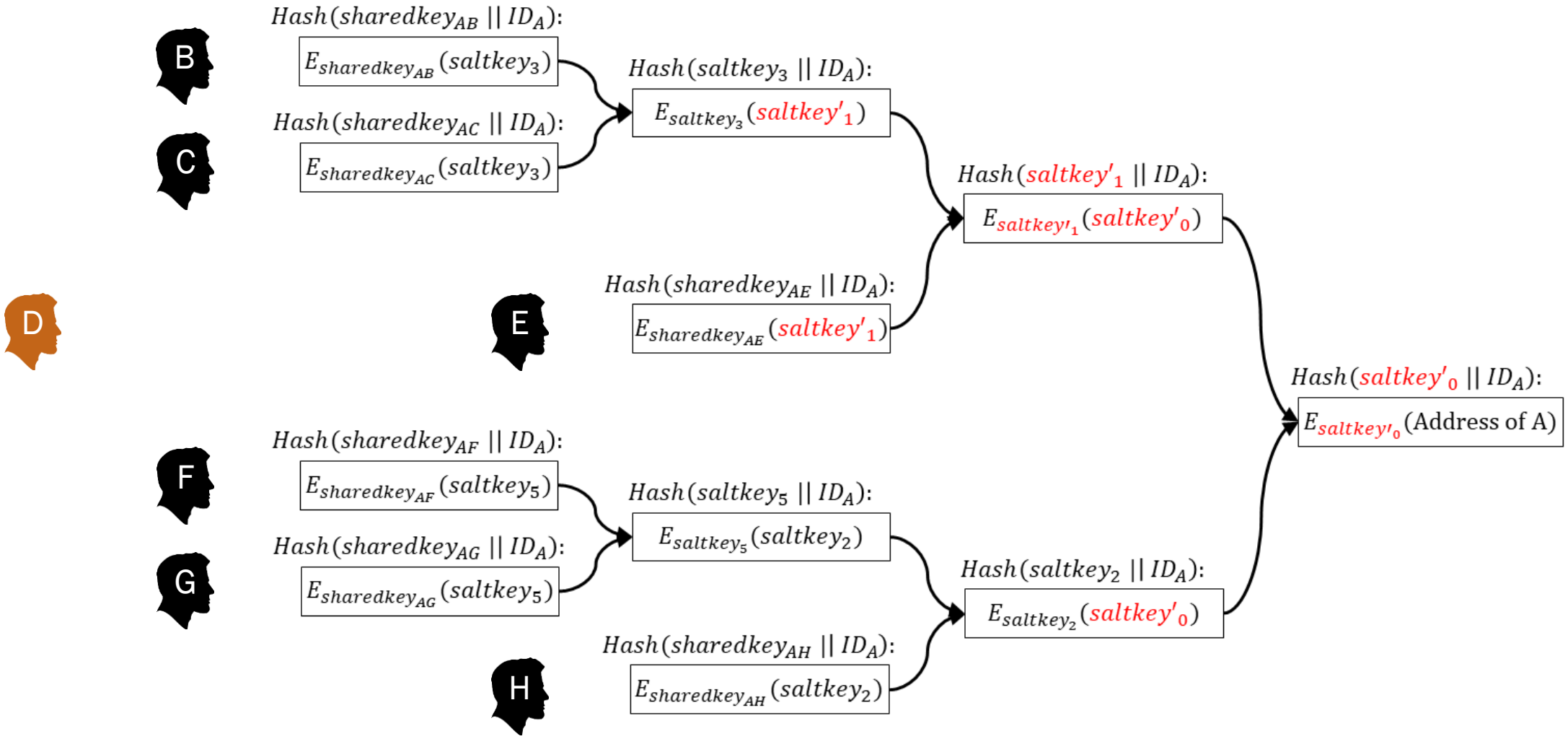
Objective

- Develop a method to share user's location information (e.g. IP address) to a set of users, i.e. the user's friends or members of some club.
- Using the location information the friends can connect to the user.
- Other users than the friends should not learn the user's IP address.
- Easy to add and remove friends
- Easy to change the user's address
- Efficient

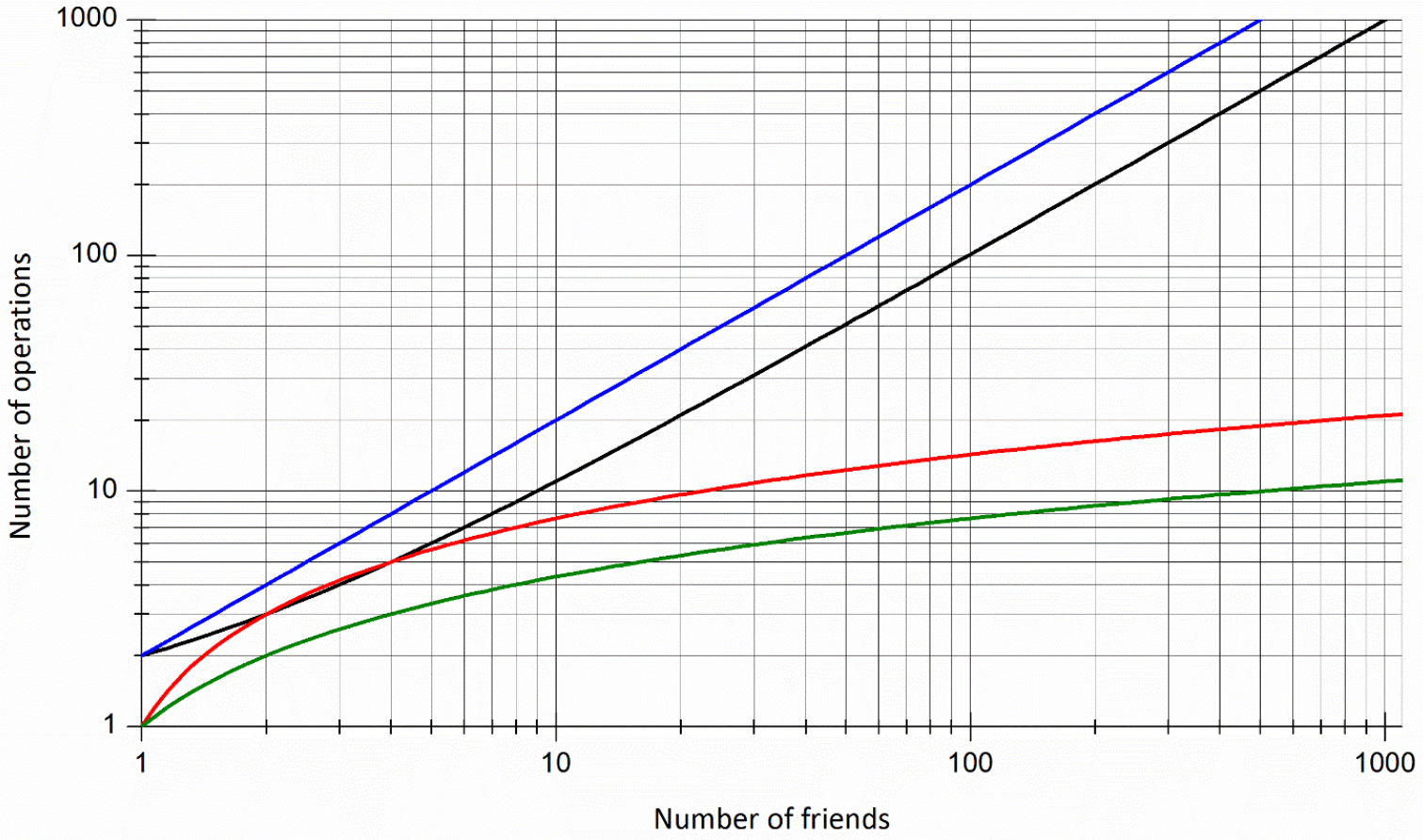
Protocol 2


B
C
D
E
F
G
H
$Hash(sharedkey_{AB} \mid\mid ID_A)$:
$E_{sharedkey_{AB}}(saltkey_3)$
$Hash(sharedkey_{AC} \mid\mid ID_A)$:
$E_{sharedkey_{AC}}(saltkey_3)$
$Hash(sharedkey_{AD} \mid\mid ID_A)$:
$E_{sharedkey_{AD}}(saltkey_4)$
$Hash(sharedkey_{AE} \mid\mid ID_A)$:
$E_{sharedkey_{AE}}(saltkey_4)$
$Hash(sharedkey_{AF} \mid\mid ID_A)$:
$E_{sharedkey_{AF}}(saltkey_5)$
$Hash(sharedkey_{AG} \mid\mid ID_A)$:
$E_{sharedkey_{AG}}(saltkey_5)$
$Hash(sharedkey_{AH} \mid\mid ID_A)$:
$E_{sharedkey_{AH}}(saltkey_2)$
$Hash(saltkey_3 \mid\mid ID_A)$:
$E_{saltkey_3}(saltkey_1)$
$Hash(saltkey_4 \mid\mid ID_A)$:
$E_{saltkey_4}(saltkey_1)$
$Hash(saltkey_5 \mid\mid ID_A)$:
$E_{saltkey_5}(saltkey_2)$
$Hash(saltkey_1 \mid\mid ID_A)$:
$E_{saltkey_1}(saltkey_0)$
$Hash(saltkey_2 \mid\mid ID_A)$:
$E_{saltkey_2}(saltkey_0)$
$Hash(saltkey_0 \mid\mid ID_A)$:
$E_{saltkey_0}$(Address of A)

Protocol 2


B
C
D
E
F
G
H
$Hash(sharedkey_{AB} \,||\, ID_A)$:
$E_{sharedkey_{AB}}(saltkey_3)$
$Hash(sharedkey_{AC} \,||\, ID_A)$:
$E_{sharedkey_{AC}}(saltkey_3)$
$Hash(saltkey_3 \,||\, ID_A)$:
$E_{saltkey_3}(saltkey'_1)$
$Hash(sharedkey_{AE} \,||\, ID_A)$:
$E_{sharedkey_{AE}}(saltkey'_1)$
$Hash(saltkey'_1 \,||\, ID_A)$:
$E_{saltkey'_1}(saltkey'_0)$
$Hash(sharedkey_{AF} \,||\, ID_A)$:
$E_{sharedkey_{AF}}(saltkey_5)$
$Hash(sharedkey_{AG} \,||\, ID_A)$:
$E_{sharedkey_{AG}}(saltkey_5)$
$Hash(saltkey_5 \,||\, ID_A)$:
$E_{saltkey_5}(saltkey_2)$
$Hash(sharedkey_{AH} \,||\, ID_A)$:
$E_{sharedkey_{AH}}(saltkey_2)$
$Hash(saltkey_2 \,||\, ID_A)$:
$E_{saltkey_2}(saltkey'_0)$
$Hash(saltkey'_0 \,||\, ID_A)$:
$E_{saltkey'_0}$(Address of A)

Comparing the number of DHT operations



For Protocol 1, the black curve represents the number of operations needed for the sharing phase, and approximately also for the removing friends phase.
For Protocol 2, the blue curve represents the number of operations needed for the sharing phase, the red curve for the removal of friends phase and the green curve for the retrieving the address phase.

Contact Us

social-media@helios-project.eu

www.helios-h2020.eu

@heliosEUproject

@HeliosEUProject

Thank you