Ledger Carpool Exchange [LCE]

Carpooling Use Case

Users of our carpooling platform are carpooling operators and not end-users. That means passengers and drivers do not interact directly with the blockchain. Each carpooling operator maintains a blockchain node in which they can publish their carpooling offers.

The scenario starts with a driver from operator 1 who proposes a carpooling offer. The offer is stored locally in the carpooling operator 1 then pushed to the carpooling platform. Once the offer is created and stored in the Blockchain, passengers from different carpooling operators can fetch it.

A passenger can search for an offer with an origin, a destination, and a date through a carpooling operator 2. The carpooling operator 2 sends a request to its blockchain node which responds by sending a list of offers that meets the given criteria. The list includes offers belonging to different carpooling operators. The passenger chooses an offer that suits him and books it.

The booking transaction is then stored in the blockchain and the platform notifies operator 1 about the booking. The driver can then confirm or reject the booking. Confirmation/Rejection transaction is also stored in the Blockchain and the passenger is notified through his carpooling operator 2. In the case of confirmation, the number of available seats decreases. The passenger may decide to cancel its booking, then a Booking cancelation transaction is stored and the driver' operator 1 is notified. The number of available seats is updated.

With the same logic, a driver can also delete his offer. In this case, the number of available seats is set to 0 and if there are bookings, they will be canceled automatically by the smart contract, and owners are notified.

The carpooling platform also proposes a carpooling proof. The proof is generated automatically based on transactions stored in the blockchain.

To recapitulate, the blockchain platform manages Offer, Transaction (booking, confirmation/rejection, booking cancelation), and Proof entities.

Global architecture

In Figure 1, we present an overview of the architecture of our carpooling platform. The carpooling solution is a web application provided for all partners (operators), where they interact and communicate through the blockchain. We detail from top to bottom different blocks.

- Carpooling operator Services: The first layer, presented in orange, includes different operators' services used by end-users via each operator web application. This layer is specific for each operator and presents the already used services without the interoperability platform. These services allow passengers and drivers to propose a carpooling offer, search for a carpooling offer, exchange messages, confirm or refuse a booking. They provide also access control services to manage their end-users.

- Carpooling Platform API Services: The second layer includes the carpooling platform services introduced to provide interoperability between operators. This layer provides services to carpooling operators such as pushing or retrieving an offer, a booking (called transaction here) or a proof in the blockchain, and uses notification service to notify different carpooling operators when they are concerned. To ensure interoperability, this layer defines governance rules in the blockchain. The Key Management System (KMS) service is added to address the privacy requirement and it is responsible for generating different used keys for our cryptographic solution. The carpooling platform is based on a REST API server written in NodeJS. This service layer is mainly used by operators to interface with the blockchain. All the actions to register an operator, retrieve or store data, as well as generate keys are launched via the carpooling platform API using simple REST APIs.
- Chaincodes: Then, we have the blockchain responsible for managing smart-contracts, storing and retrieving offers, transactions and proofs, and managing keys used for privacy. Smart contracts Offer, Transaction and Proof are also responsible for encrypting and decrypting entities. Chaincodes are developed in Typescript (NodeJS) with version 1.4 of the Fabric Shim API.
- Hyperledger Fabric Blockchain: The blockchain network used is Hyperledger Fabric version 1.4. The network contains two organizations, each containing two peers with their couchDB database, a fabric certificate authority server, and a fabric client. Carpooling platform services use the Hyperledger Fabric Client version 1.4 to interact with Peers of the Fabric network and to send transaction invocations or perform chaincode queries. It also uses the Hyperledger Fabric-CA Client version 1.4 to register operators with operator name as parameter.
- Proxy: the proxy entity is added with the Proxy re-encryption scheme. It's responsible for managing re-encryption keys and re-encrypting data. The proxy is considered as a trusted third party.
- RNPC: This is a facultative functionality that can be used to send carpooling proof to the National Register of Carpooling Proofs. For this, an account with RNPC is needed. An operator has the choice to enable or not this functionality.

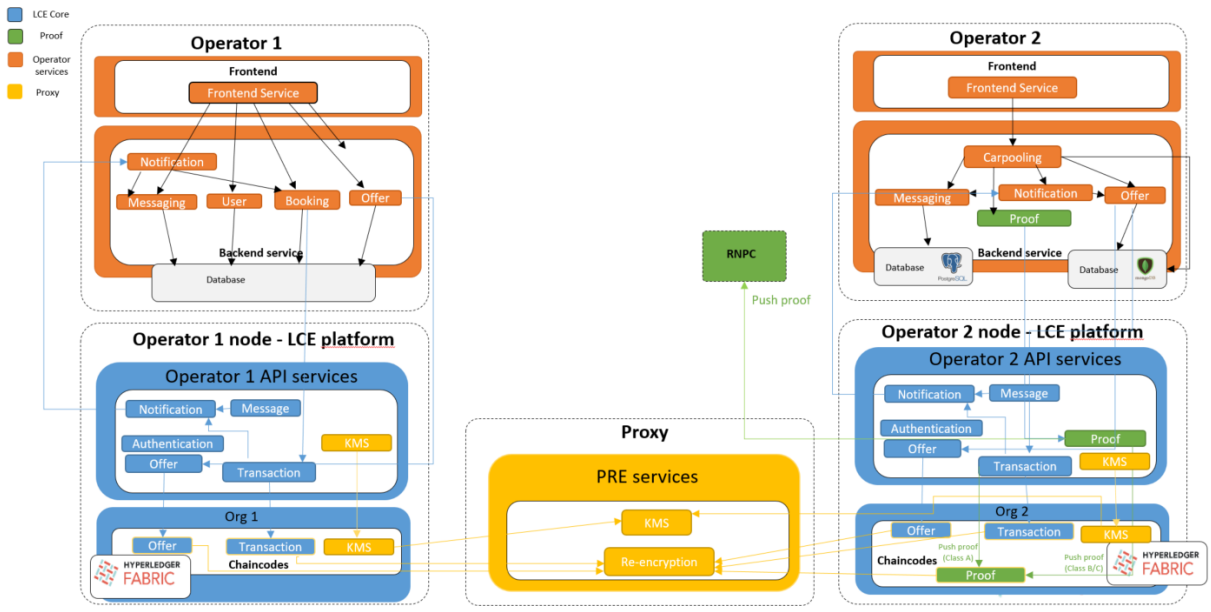

Figure 1: Global architecture