CSCI5673 Spring 2020 Group Project
DeRiS: Blockchain based Ride Sharing

Nirvan S P Theethira
nith5605@colorado.edu

Zachary McGrath
zamc8857@colorado.edu

Abstract

Ride sharing is a concept that has gained popularity over the past decade with drivers offering up free spaces in their vehicles to riders in need for a price. While this has helped reduce congestion and automobile pollution, current ride sharing systems use a centralized structure where a single entity orchestrates interactions between riders and drivers. This leads to a single point of failure. As a consequence, this centralized structure raises concerns about data privacy and is vulnerable to attack from malicious users. Furthermore, as the central entity takes a certain cut of the profit from drivers who bear the brunt of the costs of gas and vehicle wear and tear. The propose solution to this problem is DeRiS, (De)centralized (Ri)de (S)haring. DeRiS, as its name implies, is a decentralized ride sharing system based on the Ethereum Blockchain.

Related work

Related work in this area includes implementations such as Ridecoin (1) and LaZooz (2) that exist, but are either just a concept or attempt to build their own cryptocurrency and/or blockchain. Instead, we choose to use Ethereum (3) for its wide adoption and support from its community. Others, such as Chasyr (4), are built atop the Ethereum blockchain, however their methods are proprietary and the methods and organization that they use can be assumed to be different to that of what is proposed in B-Ride, if for no other reason, that Chasyr (4) allows for riders to choose their driver. Other blockchain based ridesharing apps, such as DASCEE (5) and TADA (6) are region locked to countries outside of the United States. The B-Ride (7) paper goes into great detail about how to implement a secure system for decentralized ride sharing based on the Ethereum blockchain. The paper emphasized security in all rider and driver interactions. Most of the inspiration and ideas for this project are taken from the B-Ride (7) paper.

Proposed Solution

The proposed solution to the problem above aims to use a public blockchain that enables drivers to offer ride-sharing services eliminating the need for a third party (7). We hope to use Ethereum for the public blockchain (3). The Ethereum block chain is an open-source block chain with plenty of documentation for developers to build applications upon. Furthering the notion that the Ethereum block chain is a viable public option is that the Ethereum cryptocurrency is one of the most popular on the market. We hope to build a basic application that would pair a rider with an available driver using Ethereum smart contracts. We plan on building an application atop the Ethereum block chain. The smart contracts themselves will be written using Solidity (8). The application to interact with smart contract will be written in HTML/JS with the web3js package being used for interfacing with the smart contract (9).

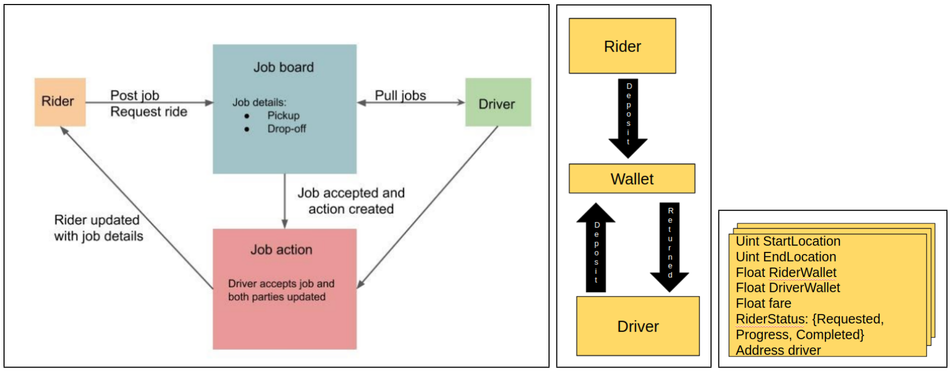

Figure 1: Rudimentary system design on the left. Commitment algorithm in the middle. Rudimentary solidity structure to store rider information on the right.

System Design

As previously stated, DeRiS will be built upon the Ethereum blockchain. This open-source and well documented blockchain is ideal for development given the large community it has and widespread support. Many existing tools and frameworks for development already exist, making the overall development process cleaner and easier. As of any ride sharing application, the preliminary need is to have riders post there trip details and for drivers to select rides to be completed. A rudimentary design of this system can be seen in figure 1. The elementary design of the system can be outlined as follows:

- Rider post the start and end location of the trip along with his/her Ethereum account address to request a ride.
- The rider request is added to the blockchain.
- The Driver can now view all ride requests.
- Driver selects preferred rider.
- Rider pays the driver as the trip is completed.

All the data/transactions happening between the rider and driver will be handled using the Ethereum blockchain thereby satisfying the distributed requirement of the project. The Ethereum blockchain works on smart contracts which are essentially pieces of code that execute transactions between the rider and the driver. Solidity (8) will be used to write these smart contracts. The smart contracts handle all transaction logic between the rider and the driver which includes but not limited to: collecting new rider trip details, posting trip details on the driver page, updating status of the trip etc. The rudimentary solidity structure to add a single riders information to the blockchain is show in figure 1. The commitment algorithm which will use smart contracts helps deters riders from not committing to trips or drivers from backing out of trips. The algorithm take the following steps:

- The rider makes a deposit that is some percentage of the fare.
- Once the driver picks a rider, the driver deposits some percentage of the fare in good faith.
- When the driver reaches the pickup location, the deposit is returned to the driver.
- The rider deposit goes towards the trip fare.
- If rider or driver back out they lose the deposit.

The pay as you ride algorithm ensure the rides pays the appropriate amount for the trip. The algorithm take the following steps:

- The driver sends elapsed distance to rider.
- Rider verifies elapsed distance.
- Driver gets paid for elapsed distance.

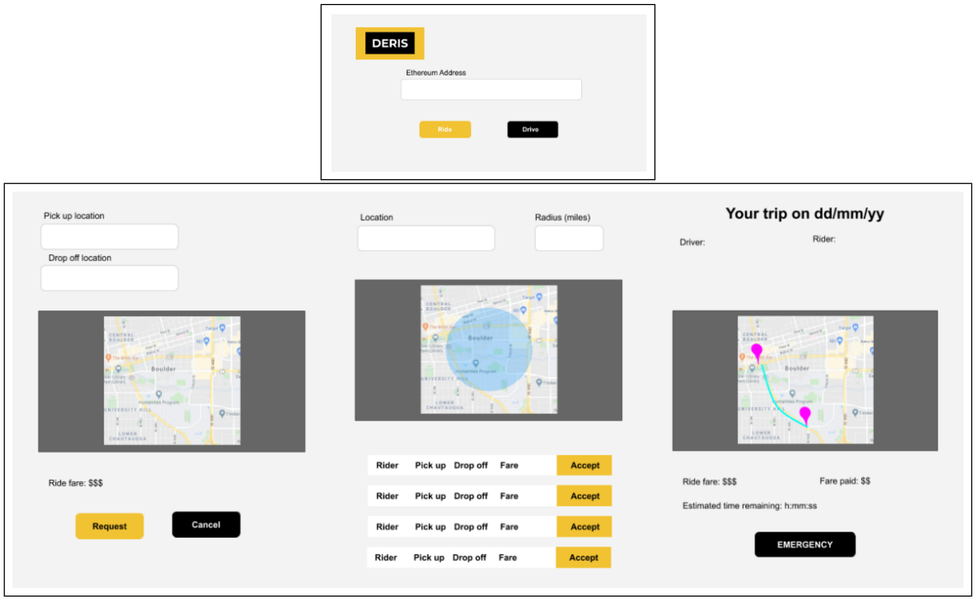

Figure 2: Rudimentary application design. On the top is the login page. On the left is the Rider's location selection page, in the middle is the driver's rider selection page and on the right is the trip in progress page.

- If trip is cut short, payment stops.

The application to interact with smart contract will be written in HTML/JS with the web3js (9) package being used for interfacing with the smart contract (see figure 2). The first page is the login page which requires the user to input his/her/them Ethereum account address. This address will be used to identify the user and initiate transactions between users. The user has to then select if he/she/they is a rider or a driver. Depending on this selection they will be directed to the rider or the driver page. In the ride page the user has to select the start and end location of the trip to request a ride. The ride requested will show up on the driver page. The driver can view rides within a certain radius of the start location. The driver can then accept a ride which leads both the rider and driver to the trip in progress page. The in progress page outlines trip details as it completes. The pay as you ride algorithm as mentioned above deducts money from the rider and pays the driver as the trip is completed.

Evaluation

A fair evaluation of a prototype for this application would be the ability to set up an account as a driver or a rider, the ability to book a ride, and the ability to send a location for pick and drop off. These initial goals would prove the feasibility of the project. A final, full fledged application would have extra features including payments (potentially in the form of cryptocurrency), route planning, and other features to be competitive with current ridesharing applications.

References

[1] Ride coin. [Online]. Available: https://www.fairride.com

[2] Lazooz. [Online]. Available: http://lazooz.org/

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger."

[4] Chasyr. [Online]. Available: https://www.chasyr.com/

[5] Dascee. [Online]. Available: https://dacsee.com/#/

[6] Tada. [Online]. Available: https://tada.global/

[7] M. Baza, N. Lasla, M. Mahmoud, G. Srivastava, and M. Abdallah, “B-ride: Ride sharing with privacy-preservation, trust and fair payment atop public blockchain,” *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2019.

[8] Á. Hajdu and D. Jovanovic, “solc-verify: A modular verifier for solidity smart contracts,” *CoRR*, vol. abs/1907.04262, 2019. [Online]. Available: http://arxiv.org/abs/1907.04262

[9] web3js developer page. [Online]. Available: https://web3js.readthedocs.io/en/v1.2.6/