Decentralized Ride Hailing System using Blockchain and IPFS

Om Naik
Department of Information Technology
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
omnaik54@gmail.com

Nimai Patel
Department of Information Technology
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
nimai.m.patel@gmail.com

Sabir Ali Baba
Department of Information Technology
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
sabirali567@gmail.com

Harshal Dalvi
Assistant Professor
Department of Information Technology
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
harshal.dalvi@djsce.ac.in

Abstract— **Ride-hailing applications like Uber and Ola have gained immense popularity in India and across the globe as convenient alternatives to traditional modes of travel such as private vehicles which can be unaffordable many times due to rising fuel prices and maintenance cost etc., buses and trains which are crowded; and taxis and auto rickshaws which can deny you service at the driver's whim. The advancement in internet technology and affordable smartphones have made it convenient to book a ride through our smartphones in just a few clicks. However, the service has also brought with it a slew of other challenges like unpredictable surge-pricing and high intermediary fees. The current research into decentralized systems for vehicles is accelerating rapidly. Even then, implementing a decentralized ride-hailing platform proves to be a difficult challenge due to the inherently centralized nature of traditional ride-hailing systems. Blockchain and other similar decentralized technologies are an attractive choice for this architecture due to their immutability, transparency and fault tolerance. If implemented successfully, decentralization using blockchain technology would remove roadblocks along the way such as intermediary fees and surge charges by third parties as it would offer more transparency. This paper proposes a framework for developing a decentralized ride-hailing architecture implemented on the InterPlanetary File System (IPFS) and Ethereum blockchain platform.**

Keywords— Blockchain, Decentralization, Ride-Hailing, Ethereum, IPFS

I. Introduction

Today, we have access to many cab-hailing services like Uber, Ola and Lyft through smart phones. These apps have become synonymous with the ridesharing and the taxicab booking industry in nearly the entire world. These services have been offering every possible conveyance system to lure their customers and increase their market share. However, all these services operate using a centralized approach to manage the operations of the system and thus have many loopholes in the current ecosystem. Like other industries, blockchain is all set to bring disruption to Uber and other cab service platforms.

The on-demand transportation business is expected to grow eightfold by 2030, mostly because of increased smartphone and internet access. In 2017, almost 5 billion rides were completed in the United States alone. Despite the high demand for taxi services, the present systems have numerous flaws. As a result, establishing a decentralized platform for drivers and riders can assist both parties in connecting with one another without the need for mediators [1].
With the introduction of cryptocurrencies like Bitcoin we now realize how decentralized technology has the potential to transform numerous sectors of the industry. The decentralized revolution is about democratizing control, taking control away from the centre, a group of people, and returning delegated power to the individuals. While the impact of decentralized technologies has been felt in the music and art industry with Non-Fungible Tokens, transportation systems in general have not experienced any decentralized revolutions yet. Our project aims at using decentralization to relay communication and performing transactions between all parties in the ride-hailing process without the intervention of a middleman or at the very least, minimizing it as much as possible.

Decentralization, in software terms, boils down to not relaying any communication through a centralized server. However, as we progressed through the project, we realized that because of the inherent limitations of Android OS and smartphone platforms in general, we will have other problems to tackle as well. Additionally, blockchain and IPFS are still emerging technologies and so are not as well researched and tested compared to their Web2 counterparts. To add to the difficulty, our aim is to develop a mobile application while most material and examples available focus primarily on web-based browser-backed user experiences.

II. Literature Review

A. Existing Systems

1) RideX: RideX is a community-based Taxi Booking app which is built on the Ethereum Blockchain. The app includes a bidding system which is used by the drivers to 'bid' on various ride requests. RideX uses KyberSwap, a token swapping service, to convert the crypto payment tokens to stable currency like Ethereum, Dai, etc. Data such as details of financial transactions, driver ratings, driver and user details are stored on the blockchain but data about bidding, and general- purpose processing are stored off the blockchain to reduce load on the blockchain. Because of this dual way of storing necessary data, privacy is taken care of [2].

2) PeerPool: PeerPool is a decentralized ride sharing application whose main features are minimal middleman fees, and trustless smart contracts. All the user data involved is stored using the IPFS protocol so that it is decentralized, preventing any external sources from accessing the private user information [3].

3) Deris: DeRiS aims to use a public blockchain that enables drivers to offer ride-sharing services eliminating the need for a third party. The project uses the Ethereum Blockchain for deploying the smart contracts [4]. The system's basic design can be summarised as follows:

- To request a ride, the rider must post the start and end locations of his or her trip, as well as his or her Ethereum account address.
- The rider's request is recorded on the blockchain.
- All ride requests are now visible to the driver.
- The driver chooses his or her favoured rider.
- When the ride is finished, the rider pays the driver.

4) PEBERS: PEBERS is a Ridesharing System based on Consortium Blockchain. This system utilizes several Fog Computing Nodes which have computation, communication and network capabilities. They are used as semi-trusted nodes in the network. Each of these nodes hold a copy of the blockchain transaction ledger. The PEBERS system consists of four basic components: the Trusted Authority for initializing the system and maintaining the FOG computing nodes, the FOG computing nodes which form a consortium blockchain and server as agents to match passengers with the drivers, the riders and the drivers. In the system, the riders and the drivers register with the system. Each of the riders and drivers are authorized by the Trusted Authority after some sort of identity authentication. After authorization, the ride process begins when a rider requests a ride. This ride is first authenticated by the FOG nodes and then they broadcast the ride details to the drivers. The driver smart contract is deployed. This phase is followed by a matching protocol conducted by the FOG nodes, and they match the driver and the rider. A consensus mechanism is used to ensure the consistency of the blockchain ledger. After the ride is completed, the payment is made using cryptocurrency and the details of that transaction are written on the blockchain [5].

5) B-Ride: B-Ride is a blockchain-based ridesharing system which uses smart-contracts to mitigate single point of failure issues presented in classical client-server architectures. It is a privacy preserving Blockchain-based Ride sharing service. The main aim of B Ride is to remove any intermediaries between riders and drivers and thus pioneer the decentralization of the ridesharing industry [6]. The main proposals mentioned by the authors include:

- Privacy protection using cloaking, which involves hiding of the exact pickup and drop-off locations of the rider. The ride, along with the cloaked locations, is broadcast and if a driver is interested, the system uses an off-line matching technique to determine the location of the rider, thus preserving privacy.
- A time-locked deposit protocol based on a zero-knowledge membership.
- A system for ensuring a fair payment made in an absolutely trustless manner between the rider and driver.
- A reputation system for drivers based on different parameters taking prior ride experiences into consideration. B-Ride appears to be practicable in terms of both on-chain and off-chain overheads, according to the tests undertaken by the authors. Furthermore, it demonstrates the feasibility of resolving two primary objectives in the use-case of decentralized ride sharing on public blockchain: one between transparency and privacy, and the other between system users' accountability and anonymity.

B. Observations for Existing Systems

Almost all of the projects that we came across were built for web browsers and not smartphones. This meant that we had little reference that we could use for our own project which was being built for mobile platforms.

PEBERS used the novel idea of FOG computing nodes with the use of consortium blockchains which completely alleviated the need for a centralized relay server necessary for some of the functionality [7]. However, because of the immense hardware requirements it also introduced a huge risk. The PeerPool project had the unique idea of using the Interplanetary File System to store large amounts of data decoupled from the actual blockchain. This way, we can reduce the amount of storage we are using on the blockchain which reduces the ever-soaring Ethereum gas fee. Additionally, because of the IPFS's decentralized and immutable nature we do not lose out on those features in our overall system as well. Huge data can be stored on the IPFS and the immutable hash of the stored data can be tracked using our Solidity smart contract running on the Ethereum blockchain. Other FOSS solutions involve a generic mix of using the Ethereum-Solidity ecosystem to keep track of driver details, rider details and ride details but most fail to provide solutions to the high Ethereum gas fee. Many of the solutions also naively include a relay server which in some cases also use full-fledged databases to tackle the high Ethereum gas fee. This, obviously, defeats the entire purpose of decentralization and poses similar middle-man issues since the database is controlled by the application developer at the end of the day [8].

C. Review of the Development Stack

1) Flutter: Flutter is User Interface toolkit developed by Google for building, natively compiled cross-platform applications. Another option was to use React Native -- a popular JavaScript-based mobile app framework that allows you to build natively rendered mobile apps for iOS and Android. Our first idea was in fact, to use React Native. However, as we started the project, we realized that many of the libraries for communicating with the Ethereum blockchain were only available for React on the browser, namely the web3 library. This led us to using Flutter as our UI framework.

2) Solidity: Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state. Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features. Most of the execution time of transactions depends on the validation method used.

Different blockchain networks employ various validation methods. If all transactions are validated, the new block gets "chained" onto the blockchain. After that, the new current state of the ledger is broadcast to the network. This whole process can be completed in 3-10 seconds which is a reasonable amount of minimum time for trying to find a driver and making payments [9].

3) IPFS: IPFS or InterPlanetary File System uses content addressing to identify files. It uses a hash of the file as its content ID. It uses Merkle Directed Acyclic Graphs for identification of files. To find which peers are hosting the content you're after (discovery), IPFS uses a distributed hash table, or DHT [10].

4) OrbitDB: OrbitDB is a peer-to-peer database for the decentralized web. IPFS generates the hash of the data saved into OrbitDB and sends it to the blockchain database.

III. Methodology and Approach

A. Scope

The smartphone application will be built for the Android operating system and involve interfaces keeping in mind two kinds of users: the rider and the driver. The primary aim of the application is to pair the riders who want to go to a destination with drivers who are willing to take them there. Login and registration features will have to be built for each type of entity. For the rider's side specifically, pages for being able to select start and end destinations which provide map view of both, pages for viewing profiles and shortlisting drivers from list of prospective drivers and pages for facilitating the payment to the driver and rating them at the end of the ride will have to be programmed. For the driver's side, pages for confirming ride requests from riders, map view of start and end locations and payment screens along with rating screens will have to be programmed.

B. Assumptions

The Flutter platform has libraries for developing Decentralized Applications or dApps, but they are still mostly in the alpha stage of their development which means a lot of the programming had to be done by ourselves too that should have been at the API level.

The Solidity language being used for developing smart contracts and the Dart programming language used for programming the user interface in Flutter don't have types that match exactly to each other. For example, one might have only a single integer type while the other has separate types for BigIntegers. This requires a lot of care on the developer's part as it could cause involuntary type casts leading to unpredictable behaviours.

Due to the eventually-consistent nature of the IPFS, sometimes there might be delays in updating databases and smart contracts leading to discrepancies. Although this does not cause any logical errors in the ride hailing process, it might mean some riders and drivers might miss out on opportunities if their device is not fast enough to execute the smart contracts and/or cannot make requests as quickly to the server running the IPFS daemon. In the worst cases, this might lead to a form of starvation.

Since the system is running on the Android operating system as compared to a full-fledged desktop operating system like Windows, Linux or MacOS it cannot run its own IPFS daemon. Even if it could, from anecdotal experience we have noticed serious performance degradation from running it even on proper desktops. Because of this, the smartphone will instead make a request to an external server that is running the IPFS daemon instead of running a daemon itself. It should be noted at this stage that this will indeed introduce a degree of centralization, but it is necessary to keep performance and operational costs in check.

C. Implementation

1) Signup System: The signup system will be included for both the rider and driver side of the application. It is relatively simplistic as it is assumed both the rider and driver have Ethereum wallets. Therefore, their private and public keys can be used to uniquely identify the entities. The signup process will involve typical signup pages for the users and their details will be stored on the IPFS through the OrbitDB abstraction layer API. The hash for these details will then be stored in the deployed smart contract which exists on the Ethereum blockchain.

2) Login System: The login system will also involve typical user interfaces for both the driver and rider. In both cases, for the login and the signup, the private key of the user will act as secrets like the password which they will have to maintain. The login system will look up their details on the IPFS-OrbitDB database

3) Driver and Rider Pairing: This is the crux of the system. The details of the driver are intermittently updated and polled for on the IPFS database. This means when the rider requests for a ride they get details as new as efficiently possible. The prospective drives list is generated using a spherical law of cosines rule which eliminates all drivers beyond a certain threshold distance.

4) Ethereum transfer: After the ride is completed, an easy interface is created to transfer the Ethereum tokens from the rider's crypto wallet to the driver's crypto wallet automatically. The cost is calculated by a function that depends on the distance of travel and the time it took for the journey to complete.

5) Driver and Rider Rating System: After the payment is completed, the driver and rider are both provided with pages to rate each other. The details are accepted, averaged out with existing values on the database and then updated on the database [11].

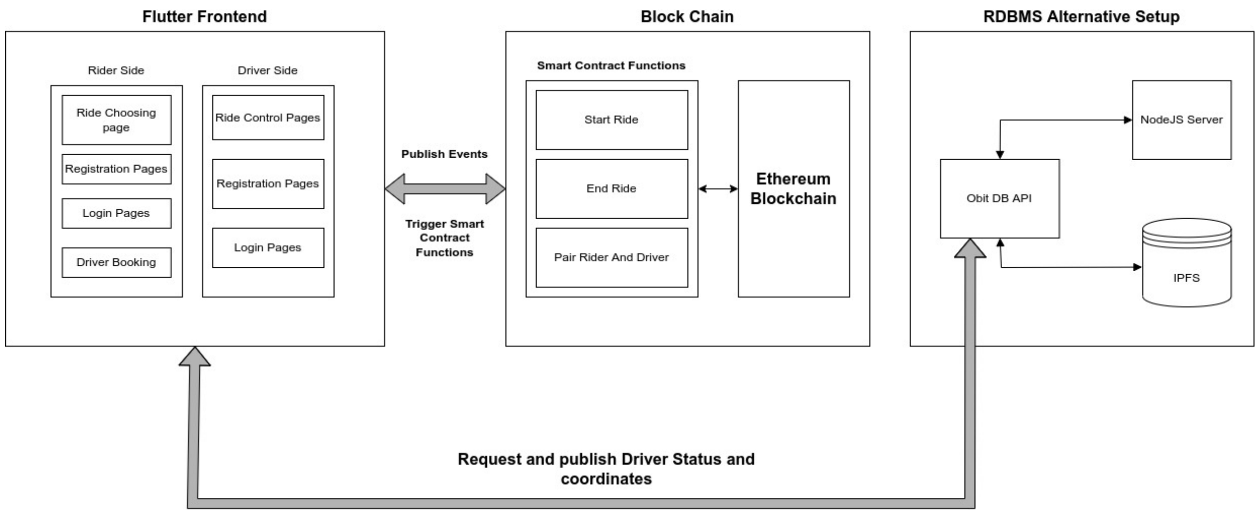


Fig 1: Architecture

D. Benefits of Proposed Implementation

This implementation specification lets us enjoy some interesting benefits. Firstly, the system reduces the influence of a middleman to the greatest degree possible while being operational on smartphone platforms like Android. Also, the system does not require special technical knowledge from the users' side, i.e., the riders and drivers. If they have used systems like Uber and Ola then they can easily try out and transition to using our system. As long as they know how to open an Ethereum wallet using many of the available apps in the play store they will have a streamlined experience. Also, no specialized hardware is needed for the system to run for either the user or the development team. Lastly, using Flutter means that the application is hypothetically easy to port to other platforms apart from Android like iOS and the web.

IV. System Architecture

Fig. 1 portrays a graphical representation of the system architecture. Its principal components are as follows:

A. Frontend

The front end is built on top of the Flutter framework using the Dart programming language. It includes pages to signup, login for both the rider and the driver. Pages to select prospective drivers, pages to accept or reject requesting riders, pages to select journey endpoints and route, pages for rating the driver and the rider and pages for making the final transaction.

B. Blockchain

We are using the Ethereum blockchain to deploy our smart contracts. The smart contracts themselves are written in the solidity programming language which is an object-oriented language for defining the structure and methods involved in an application. Our application involves methods for starting a ride, ending a ride, initializing a ride, pairing a rider to a driver by making a request to our IPFS server, et cetera.

C. Alternative-to-DBMS

Since we cannot use a database for keeping records of application transactions, we use the IPFS instead. The IPFS however, in itself is only a file system like EXT4 or FAT32. It is only capable of storing files like on a disk. To complete the requirement of being a full-fledged database we use OrbitDB which is a complete database built on top of IPFS. OrbitDB is a serverless, distributed, peer-to-peer database. It uses IPFS as its data storage and IPFS Pubsub to automatically sync databases with peers. It's an eventually consistent database that uses Conflict-free Replicated Data Types (CRDTs) for conflict-free database merges, making OrbitDB an excellent choice for dApps, blockchain applications and offline-first web applications [12].

V. Implementation

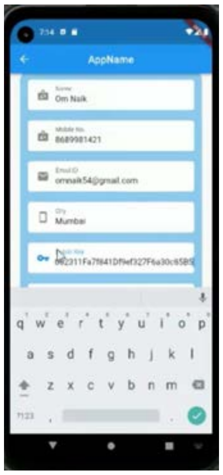
The finally developed system has the following flow of execution:

1. The User logs in or signs into the application. Their details are verified with data from OrbitDB on IPFS or they are stored there if not already present. The immutable hash of this data is stored on the smart contract.

2. The rider enters details of the start and end position of their ride. These are used to semi-initialize a new block on the smart contract. These details are then sent to the relay server.
3. At the relay server, we have continuously polled data of the locations of all active drivers. The spherical law of cosines is used to create a list of qualifying drivers using the data on IPFS. This list is then sent back to the user.
4. The user selects one of the riders and this is added to the smart contract.
5. The rider receives notification for a ride and they can accept it. This then triggers the ride block.
6. The driver reaches the pickup location and picks up the rider. Both mark this event in their app.
7. At the destination, both mark that they have reached the destination in their app as well. This triggers the final event for the smart contract and the ether is transferred from the rider's wallet to the driver's wallet.
8. They are both presented with a page to rate the other. This rating is averaged with existing ratings on the IPFS and updated over there.

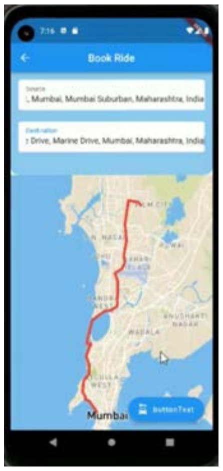
VI. Results

Here we present some screenshots from our app. Fig 2 shows our user registration page while Fig 3. shows how a user can select their pickup and destination which generates appropriate polylines to provide visual feedback. Similarly, pages for confirming the ride, making the payment and rating the users are also present.



Name
Om Naik
Mobile No.
8689981421
Email ID
omnaik54@gmail.com
City
Mumbai
Public Key
0x2311Fa7f841Df9ef327F6a30c65B5

Fig 2: Registration Page



Source
, Mumbai, Mumbai Suburban, Maharashtra, India
Destination
: Drive, Marine Drive, Mumbai, Maharashtra, India

Fig 3: Journey Selection

VII. Future Scope

Based on some limitations that our system suffers from; we have realized the following aspects which might need further improvement. Firstly, we need to completely strip out the need for a NodeJS relay server by figuring out a method to efficiently operate an IPFS daemon on smartphones or at least be able to operate a remote daemon without using a relay server. We also need to further refine the blockchain connection and the contract linking with the blockchain. Finally, the existing naive rating system which uses averages has to be replaced with a heuristic-based reputation model for the drivers.

VIII. Conclusion

With our research on various prevalent ride hailing services, we have realized that these services have a lot of potential for data loss or data theft due to their centralized way of storing private data. Also, the involvement of the companies as a middleman is very high. In order to reduce this overinvolvement of the middleman, we have made this system. This project is a complete overhaul of the current ride hailing services. During the process of developing this application, we chanced upon various existing works in this domain. We learnt a lot from these works and came up with our own system. We deliver a completely decentralized ride hailing system which eliminates the need of a middleman in the ridesharing industry. This app allows riders to choose a pickup and drop-off location through the user interface and allows them to book rides. We store this ride data on the blockchain. We also use the InterPlanetary File System or IPFS as a secondary storage medium for data which is trivial for storage on the blockchain. This is a feature of our application which makes us stand out from the existing technologies.

References

[1] Blockchain to Disrupt Uber: Entering Ridesharing Industry, LeewayHertz. Retrieved on 1st June, 2022 from https://www.leewayhertz.com/blockchain-disrupting-uber-platform/
[2] RideX | Decentralized taxi experience on Ethereum Blockchain, UX Planet. Retrieved 1st June, 2022 from https://uxplanet.org/ridex-taxi-service-on-ethereum-blockchain-fecee1879a23
[3] PeerPool: A Decentralized Ride Sharing Application. Retrieved 1st June, 2022 from https://github.com/adityakeerthi/PeerPool
[4] DeRiS: Blockchain based Ride Sharing. Retrieved 1st June, 2022 from https://github.com/Nirvan66/DeRiS
[5] Kudva, Sowmya, Renat Norderhaug, Shahriar Badsha, Shamik Sengupta, and A. S. M. Kayes. "Pebers: Practical ethereum blockchain based efficient ride hailing service." In 2020 ieee international conference on informatics, iot, and enabling technologies (iciot), pp. 422-428. IEEE, 2020.
[6] Baza, Mohamed, Noureddine Lasla, Mohamed MEA Mahmoud, Gautam Srivastava, and Mohamed Abdallah. "B-ride: Ride sharing with privacy-preservation, trust and fair payment atop public blockchain." IEEE Transactions on Network Science and Engineering 8, no. 2 (2019): 1214-1229.
[7] Omar Dib, Kei-Léo Brousmiche, Antoine Durand, Eric Thea, Elyes Ben Hamida. Consortium blockchains: Overview, applications and challenges. International Journal on Advances in Telecommunications, IARIA, 2018.
[8] V. Buterin et al., "Ethereum white paper: a next generation smart contract & decentralized application platform," First version, 2014.
[9] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," Ieee Access, vol. 4, pp. 2292–2303, 2016.
[10] Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. ArXiv, abs/1407.3561.
[11] M. Y. Alam, S. Saurav, R. Mandal, S. Saha, S. Nandi and S. Chakraborty, "A fair and effective driver rating system for developing regions," 2017 9th International Conference on Communication Systems and Networks (COMSNETS), 2017, pp. 454-459, doi: 10.1109/COMSNETS.2017.7945432.
[12] OrbitDB: Peer-to-Peer Databases for the Decentralized Web. Retrieved 1st June, 2022 from https://github.com/orbitdb/research