Blockchain-based Peer-to-Peer Ride Sharing

Project-II (CS47006) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Somay Chopra

(18CS10052)

Under the supervision of

Professor Shamik Sural

IIT Kharagpur

and

Professor Balaji Palanisamy

University of Pittsburgh

Department of CSE

Indian Institute of Technology Kharagpur

Spring Semester, 2021-22

April 30, 2022

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 30, 2022 (Somay Chopra)

Place: Kharagpur (18CS10052)

DEPARTMENT OF CSE

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

CERTIFICATE

This is to certify that the project report entitled "**Blockchain-based Peer-to-Peer Ride Sharing**" submitted by **Somay Chopra** (Roll No. 18CS10052) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under our supervision and guidance during Spring Semester, 2021-22.

Professor Shamik Sural
Department of CSE
Indian Institute of Technology Kharagpur
and
Professor Balaji Palanisamy
University of Pittsburgh

Date: April 30, 2022
Place: Kharagpur

Abstract

Name of the student: **Somay Chopra** Roll No: **18CS10052**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of CSE**

Thesis title: **Blockchain-based Peer-to-Peer Ride Sharing**

Thesis supervisor: **Professor Shamik Sural** and **Professor Balaji Palanisamy**

Month and year of thesis submission: **April 30, 2022**

An increasing number of vehicles running on the road in recent years has become a major concern due to the adverse effects they have on the environment and the society, such as air pollution, noise pollution, mental stress, etc. Carsharing is one way that has been proved to be effective in solving this problem. It allows users with compatible routes to share a ride, which not only saves the cost of running multiple vehicles but also reduces the negative impact on the environment. Although carsharing services exist, they are typically commercial in nature and are controlled by central authorities. This report discusses an approach to build a non-monetary decentralized platform for the users to interact directly and offer or share rides with other users in the system. The application is executed on a blockchain, thus allowing complete transparency in its operations.

Acknowledgements

I would like to acknowledge and give special thanks to my supervisors Professor Shamik Sural, Department of Computer Science and Engineering, IIT Kharagpur and Professor Balaji Palanisamy, University of Pittsburgh for their guidance and support. Their immense knowledge and valuable advice carried me through all the stages of my B.Tech project work, which wouldn't have been possible without their supervision.

I would also like to give warmest thanks to my family and friends at IIT Kharagpur for their constant support and motivation throughout the course of my work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Carsharing has served as a potential solution to the various problems caused by an evergrowing volume of traffic in recent years (17). It allows a rider to share her ride with other riders having compatible routes, thus saving on the cost of using multiple vehicles (5). For example, it allows to share fuel and parking costs by the riders. More the riders in a carpool, higher is the saving (20). Fewer vehicles on the road has several environmental benefits (15), such as reduced noise and air pollution. The latter is due to less greenhouse gas emissions and improved air quality (9). Carsharing is benefiting for mental health (10) as well, since less driving leads to lower stress level (7) and in addition, riding with people is found to be less stressful than commuting alone.

Various services such as Uber and Lyft provide carsharing, but their primary motive is profit-making. In this model, the drivers are their employees under various forms of contracts, vehicles are owned by the drivers, and the service providers charge their users for every ride undertaken. These services work as part of centralized systems, i.e., all the steps such as ride-matching, transactions and money transfer lie in the hands of the service providers. The work discussed in this report is to build

a non-monetary decentralized carsharing application using the Ethereum Blockchain where all the users are allowed to interact directly with each other for sharing a ride. The incentive for a user to use the platform is not to earn money but to leverage and appreciate the benefits of carsharing (15). Since there is no money exchange involved and the users interact with each other directly, so for transparency and fairness, the ride information (initiation, completion, abortion, route, etc.) is stored in a public ledger and the ride-matching step is executed by smart contracts (4). Thus, it is available to all in a decentralized peer-to-peer network.

The proposed application 1.1 allows a user to share a ride with other users in the system. Likewise, the user can also act as a driver providing his own vehicle to other users looking for a ride and having compatible routes. A user in the system can thus act both as a rider and a driver. Fairness in this system refers to the fact that each user does not only enjoy rides but also offers to drive. The application ensures fairness by introducing tokens. Users are rewarded or penalized with tokens based on their activities. Offering rides increases the token count for a user while availing a ride from another driver deducts some of the acquired tokens.

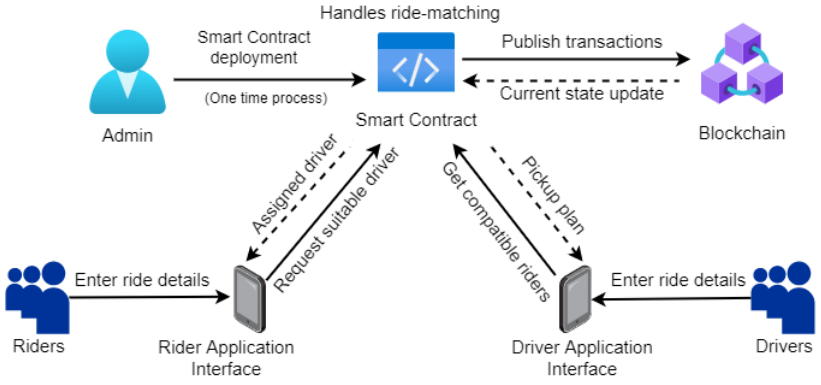

FIGURE 1.1: System Design

The rest of this thesis is divided into five chapters, each dealing with a distinct aspect of the project. Chapter 2 presents some of the previous work done in similar areas

and how the work presented in this thesis is different from those. Chapter 3 discusses the components involved in building the platform and a brief description of them, the ride-matching algorithm developed, and the Ethereum blockchain network on which the entire system is developed. Chapter 4 provides the implementation details of the algorithm using smart contracts and an illustrative example to describe the working of the algorithm. Chapter 5 gives a summary of the thesis and the planned work that is to be done in the future.

Chapter 2

Related Work

It is well-known that services like Uber, Lyft and Ola allow their customers to share rides with other people, which not only benefits the individuals by saving money but also leads to less number of cabs on the road. However, these services have started to control the cab service markets and are purely money driven. They are centralized, meaning that the prices and the operations lie in the hands of the service provider itself and not with the users. We feel that a potential solution to the problem is to use blockchains, so that the workings and transactions are transparent to the users and they can interact directly with each other without the need for a central agency.

It needs to be mentioned that some models for such decentralized (blockchain based) carsharing systems have been proposed before. For example, Pal et al. (18) propose a decentralized version of currently existing centralized cab services. Unlike centralized systems, it is implemented using blockchain. The blockchain ledger is available to all the users in the network, thus ensuring fairness of the ride. Sathya et al. (1) present a decentralized application (DApp) for ride-sharing, aiming to get rid of the convenience fees and the payment gateway charges that come with third parties due to centralized approach. It uses smart contracts to carry out all the transactions in

the system automatically in a secured and safe manner. Tripathy et al. (23) present a decentralized casharing consortium for the car owners and renters to use their cars conveniently. The primary reason for the use of blockchain is to ensure a secure and transparent carsharing system, where the owner and the renter can directly interact without the need for a third party to be involved. Vazquez et al. (24) discuss a blockchain-based solution to ride-sharing for eliminating the problems due to existing centralized approaches such as data privacy and community involvement. Since all the data and processing takes place in a central system controlled by a single body, blockchain can help to create an autonomous system using data from drivers and passengers securely to process transactions. The paper proposes to create a system to implement a safe and efficient relationship between passenger and driver while still allowing to use data for predicting common patterns and frequent trips.

Saurabh et al. (21) in their paper present an interesting approach of building a social media platform for carsharing, acknowledging the growing potential of social media applications in coming generation connectivity. In contrast to the centralized social media platforms existing today, it presents a blockchain based social media platform approach to a carsharing decentralized application (DApp) use case. Some of the previous work has also looked into the issue of fairness in different areas due to introduction of decentralization in ride sharing systems. For example Sid et al. (3) address the solution to the problem that in a decentralized system, commuters are not particularly inclined towards a decentralized system, if no individual benefit is involved. The authors discuss matching mechanisms and fair cost sharing mechanisms, to make achieve a social optimality along with the maximizing the matching of drivers with other passengers. Sánchez et al. (22) provide a possible solution to the problem of lack of trust among the users in a ride-sharing system. The authors discuss a decentralized reputation management protocol which ensures that the users find no incentives to deviate from the prescribed protocols based on the

notion of co-utility.

The decentralized approach in carsharing has been implemented in the market by some startups as well. One such startup named Darenta Crypto Carsharing is a decentralized autonomous car rental service. Everything on this platform is executed using smart contracts without any executives involved. HireGo is another example of blockchain in the carsharing market. It provides a decentralized structure and user control. The platform uses its own tokens for transactions in smart contracts, which can be converted to other currencies.

The model proposed in this thesis is not incentivized by money earning or digital currency, but rather motivated by the fundamental benefits of carsharing. Users are allowed to share as well as provide rides to other users in the system, which will save the cost of using privately owned vehicles every time, and thus lead to a fewer number of vehicles on road. Since the execution is done on a blockchain, the working and the transactions are transparent to the users, who will be able to communicate directly without the need for a third party. To maintain fairness in the system, it needs to be ensured that every user not only requests for a ride but is also required to provide rides to other users from time to time as well.

Chapter 3

Proposed Methodology

In this chapter, we discuss the main modeling components of the proposed ride sharing platform. First, we discuss the Ethereum Blockchain on which the platform runs and then the smart contracts using which the back-end execution of the ride matching algorithm takes place. Finally, the ride matching algorithm is discussed, which forms the most important component of any carsharing system.

3.1 Ethereum Blockchain and Smart Contracts

Blockchain was introduced as part of a proposal for bitcoin (16). It is a type of distributed database that stores data in the form of blocks which are chained to each other cryptographically (14). The transactions and blocks are verified by mutual consensus among the peers of the system. Blockchain technology is defined as a decentralized, distributed ledger that records the transaction history and their details, where the public ledger is distributed among the peers (13). Every executed transaction is available to all the users to verify. This fact that it is open to all the peers makes blockchain free from malpractices (6).

Ethereum (2) is an open-source blockchain featuring smart contracts. It is a generalized technology to build the concept of transaction-based state machines and has its own digital asset (cryptocurrency) (19) known as Ether which is used for inter-account payments as well as to pay transaction fees. Decentralized applications (DApps) can be easily created using Ethereum because of the rich functionality that it offers such as smart contracts.

Smart contracts are self-enforcing agreements embedded in computer code managed by a blockchain. In other words, it is a program (collection of code and data) that runs on the Ethereum blockchain. They are a type of Ethereum account, which has a balance and can send transactions over the network but are not controlled by any user. The contracts are deployed to the network and run as programmed. Once the contracts are deployed, user accounts can then interact with them by submitting transactions that execute a function defined on the smart contract. Smart contracts can define rules, similar to regular contracts, and automatically enforce them via the code.

Smart contract in blockchain has revolutionized several platforms as it provides speed, transparency and fair accounts of events to all the parties involved. Therefore, smart contracts find their application in various fields (12) such as payments in banking and finance, agreements in real estate, claim adjudication in insurance, etc.

In this project, we use smart contracts for implementing a decentralized carsharing service. The central component of this service, which is the ride-matching algorithm is described next.

3.2 Ride-Matching Algorithm

Ride-Matching is an integral part of the carsharing application. Once the users offering to drive and the users who are willing to ride have submitted their requests in the system, the ride-sharing algorithm handles the part of matching the driver with the riders having compatible routes. The algorithm takes the drivers and riders at a given point of time as input along with other details. For every driver, the ride matching algorithm outputs a plan for the driver from his starting point to his destination while picking up the riders. It is done along the way so as not to increase the original route of the driver significantly and also ensuring to pick as many riders as possible.

After the users who are offering to drive and the users requesting a ride have entered the details in the system, the ride-matching algorithm starts its execution, matching the drivers with the riders that have compatible routes. A rider's route is compatible with the driver if the extra distance covered by the driver in picking the riders is within a percentage of the original (i.e., with no riders to pick) distance of the driver.

The algorithm requires the driver and rider details at a point of time as input. For every driver, the algorithm outputs a path plan for the driver to follow, starting from the start point of the driver, picking up the riders along the way that the algorithm has matched, and finally ending at the destination point of the driver.

Fairness in the system is maintained by introducing a token system, enforcing which ensures that a user does not always requests for rides but is also required to give others users a ride, thus from time to time acting as a driver too. This can be ensured by maintaining *tokens*. Each user has a number of tokens associated with his account. Every time a user gives a ride, the number of tokens increase, and they decrease if the user takes a ride from another user. To avail a ride, *tokens* for a user must be greater than a threshold $token_{th}$.

The input, output and the algorithm steps are as follows

Input:

- Graph G(V,E), where

  V = Vertices are the junctions existing in the roadmap

  E = Edges, an edge e ∈ E for every road between pair of vertices in V

- Drivers: D_1, D_2, ..., D_d, where

  d = Number of Drivers

  D_i = <start, end, seats, tokens>

  start = Start node of the driver D_i

  end = End node of the driver D_i

  seats = Number of seats available to add rider in D_i's vehicle

  tokens = Number of tokens available in the account of user D_i

- Riders: R_1, R_2, ..., R_r, where

  r = Number of Riders

  R_i = <start, end, tokens>

  start = Start node of the driver R_i

  end = End node of the driver R_i

  tokens = Number of tokens available in the account of user R_i

Output:

- Paths Assigned: PD_1, PD_2, ..., PD_d, where

  d = Number of Drivers

  PD_i = {N_1, N_2, ..., N_n}, where

  N_i = <nodeId, riderId>

  nodeId = Node Id of the ith node in the plan path

  riderId = Rider to pickup or drop-off at that node

Algorithm:

Consider a rider only if R_i.tokens is greater than or equal to the $token_{th}$.

Sort the drivers are sorted in decreasing order of the time they entered their travel.

Let SP(u,v) represent the shortest path function which returns the shortest path from node u to node v in the Graph G and Dis(Path) returns the distance of a Path.

The Algorithm is as follows:

Algorithm 1 Ride-Matching Algorithm

for D_i *in Drivers* **do**
  $Short_P \leftarrow SP(D_i.start, D_i.end)$ ▷ Calculate the shortest path for the driver
  $MaxDis \leftarrow 1.2 * Dis(SP(D_i.start, D_i.end))$
  $MinDis \leftarrow Dis(SPL(D_i.start, D_i.end))$
  $current \leftarrow D_i.start$
  $k \leftarrow (MaxDis - MinDis)$
  while $D_i.seats \geq 1$ *and* $current \neq D_i.end$ **do**
    for R_i *in Riders* **do**
      if $Dis(SP(current, R_i.start)) \leq k$ **then**
        $WithinK.add(R_i$
      end if
    end for
    for R_j *in* $WithinK$ **do**
      $PermuteNodes \leftarrow [current$, *unvisited nodes in* PD_i, $R_j.start$, $R_j.end$, $D_i.end]$
      $NewPath \leftarrow FindPossiblePath(PermuteNodes)$
      if $Dis(NewPath) \leq MaxDis$ **then**
        $Eligible \leftarrow R_j$
      end if
    end for
    if $Eligible \neq empty$ **then**
      $current \leftarrow Eligible.start$
      $k \leftarrow (MaxDis - Dis(NewPath))$
      $D_i.seats \leftarrow D_i.seats - 1$
      $D_i.tokens \leftarrow D_i.tokens + 1$
      $Eligible.tokens \leftarrow Eligible.tokens - 1$
    else
      $current \leftarrow current.next$
    end if
  end while
end for

Helper Functions assumed for the above algorithm are as follows:

1. *SP(u,v)*: returns the shortest path between starting from node v and ending at node v.

2. *Dis(Path)*: returns the length of the *Path*.

3. *FindPossiblePath(Nodes)*: returns the shortest possible covering all the nodes in *Nodes*, among all the permutations of the nodes, keeping the start and end node fixed.

Chapter 4

System Implementation and Results

In this chapter, the implementation details of the ride-matching algorithm in the smart contract are discussed. We also show a sample run of the algorithm under example scenario to understand its working.

4.1 Implementation Details

The complete system is executed on the Ethereum Blockchain. The ride-matching algorithm is implemented in a Smart Contract (a useful functionality of Ethereum Blockchain). The Smart Contract code is written using Solidity, which is an object-oriented programming language for writing smart contract (25). For testing and compiling the smart contract, the Remix Ethereum online editor was used. Remix IDE allows developing, deploying and administering smart contracts for Ethereum like blockchains.

The deployment and execution of transactions on Ethereum main network requires Ethers which are expensive to own, so instead Goerli Test Network was used, which provides the complete functionality of the main network and issues some Ethers to work with for free.

The code snippet 4.1 gives an overview of the main functions and data structures in the smart contract.

.

```
contract CarSharing {
  //Data Structure used for the implementation
  struct Drivers {
      uint256 start;
      uint256 end;
      uint256 seats;
      uint256 tokens;
  }
  struct Riders {
      uint256 start;
      uint256 end;
      uint256 tokens;
  }
  struct PlanNode {
      uint256 nodeId;
      int256 riderId;
  }
  struct Plans {
      PlanNode[] planNode;
      uint256[] allRiders;
  }
  Drivers[] drivers; //Users offering to ride
  Riders[] riders; //Users requesting for a ride
  Plans[] plans; //Output path plan for each driver
```

```
//Main functions
function dijkstra(uint256 src, uint256 end) public
            returns(uint256[] memory){}
function permute(uint256[] memory a, uint256 l, uint256 r,
            uint256 start, uint256 end) public{}
function findPossiblePath(uint256[] memory tspnodes) public {}
function rideMatch() public {}
```

Code Snippet 4.1: Smart Contract main functions and data structures.

The shortest path between two nodes in a graph is found using Dijkstra's Algorithm (8), which is implemented in the dijkstra() function. The permute() function is a helper function to get all the permutations of a set of nodes, keeping the start and end nodes fixed. The findPossiblePath() function calls the permute() function and finds the optimal path possible. The rideMatch() function is the main function which finds the path plan for each driver in Drivers[]. For each driver, it iterates through the Riders[] to look for eligible riders by checking for the eligibility conditions and calling the findPossiblePath() function.

The testing of the smart contract was done on a subset of the map of Northwest Atlanta region of Georgia (Figure 4.1). The map was extracted via Google Maps, and the distance between the nodes was measured using the Measure Distance functionality of Google Maps.

The figure contains 20 nodes, each junction/end of road marked as a separate node, and the distance between labelled in meters. The smart contract was tested with 8 users, 4 users as drivers and 4 users as riders. The maximum distance to be travelled by a driver was fixed to be twice the minimum distance.

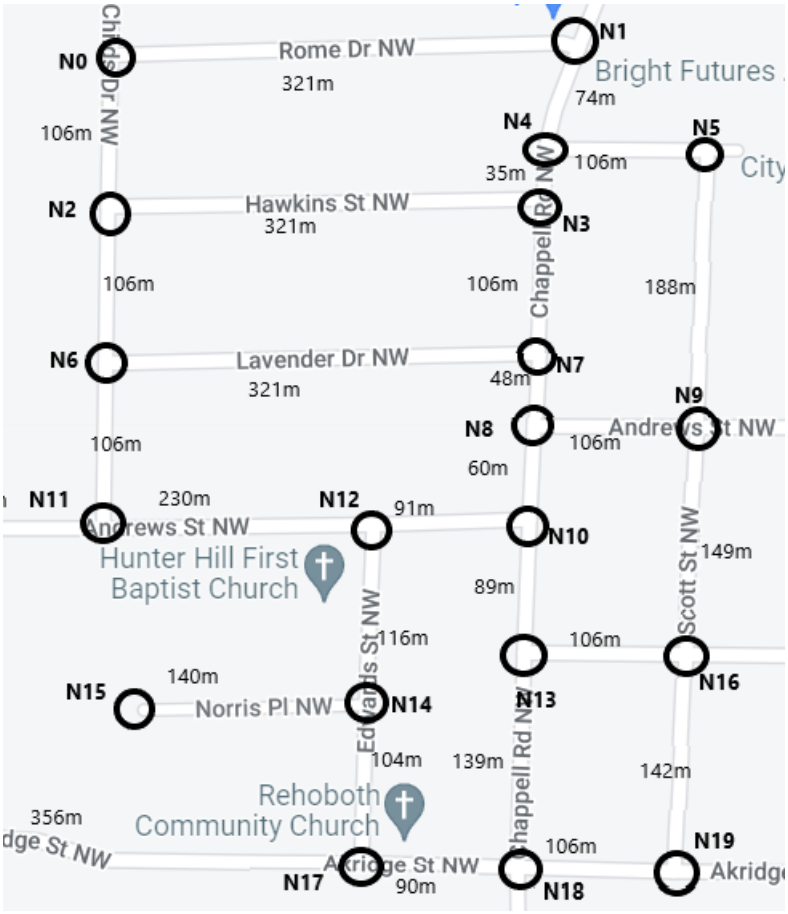

FIGURE 4.1: Map for Smart Contract Testing

4.2 Examples and Results

This section provides an illustrative example to show the working of the algorithm and the output obtained from the smart contract implemented algorithm.

Fig. 4.2 shows the sample graph used for the example, with the nodes marked and the distance between them labeled on the edges.

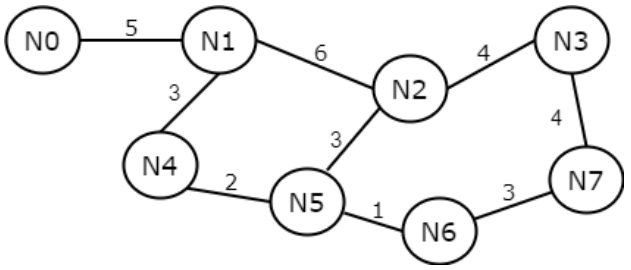

FIGURE 4.2: Sample Graph

Table 4.1 shows the sample users considered for the example.

User	**Role**	**Start**	**End**	**Tokens**	**Seats**
A	Driver	Node 0	Node 3	1	2
B	Rider	Node 4	Node 5	4	-
C	Rider	Node 6	Node 7	5	-

TABLE 4.1: Sample Users for the Example

Table 4.2 shows the path plan for 3 cases: Case 1, when only User A is in the system at the time of running of the algorithm, Case 2 shows the output for User A & B in the system, and finally Case 3 shows the output for all the 3 users in the system. The output path is in the form of (Node, Rider), where Node is the path node and Rider is the user to pickup or drop-off at that node.

In Case 1, as there are no riders to pick (Fig. 4.3), the algorithm shows the shortest path possible for the driver, although the driver is free to choose any path in this case since there are no commitments to any rider.

In Case 2, the driver starts from Node 0 with original path as Case 1 and length 15 units, the maximum distance that the driver can travel after a possible re-route is 1.2*15 = 18 units. The value of k (the radius for potential rider lookup, as mentioned in Chapter 3), is 15*1.2 - 15 = 3. At Node 0 (Fig. 4.3), there is no rider within k distance, so the algorithm follows the initial path and proceeds to Node 1. At Node

1 (Fig. 4.4), the Rider B is at distance 3 which is within k. If this rider is picked, the new path will be through Nodes 4, 5 and 2, and distance 17 units, which is less than the maximum allowed distance, making Rider B eligible to be picked up. The plan for the driver is hence updated to this new path (Fig. 4.5).

In Case 3, the algorithm works as Case 2 till Node 5. At Node 5 (Fig. 4.6), Rider C is eligible to be picked up and allows a possible re-route, hence changing the path of the driver through Nodes 6 and 7 (Fig. 4.7).

Figures below show the path for the driver while at a particular node, which is highlighted, the nodes in red is the current node being considered in the algorithm along with the value k at that node.

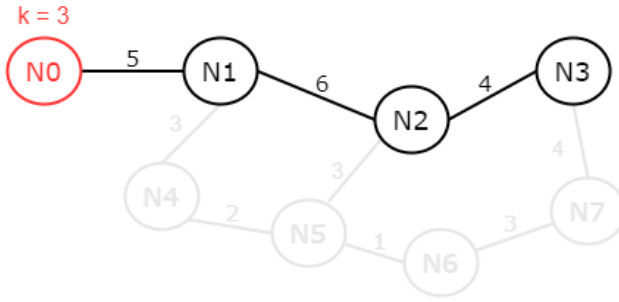

FIGURE 4.3: The initial path is highlighted, no eligible rider to be picked

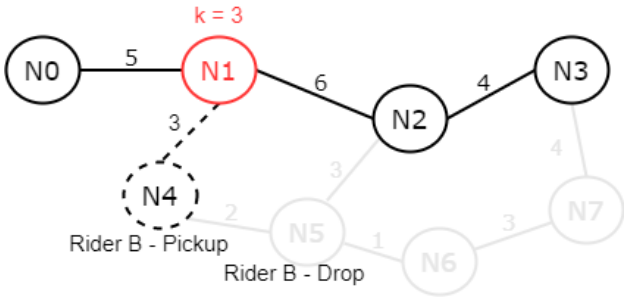

FIGURE 4.4: At Node 1, eligible Rider at Node 4

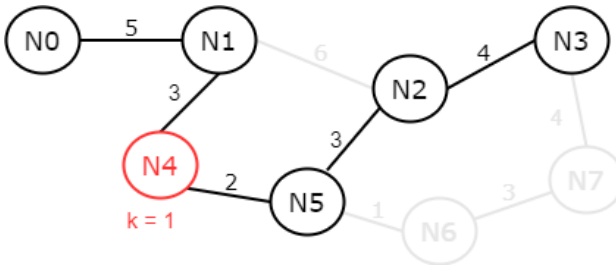

FIGURE 4.5: Path updated to pick Rider at Node 4

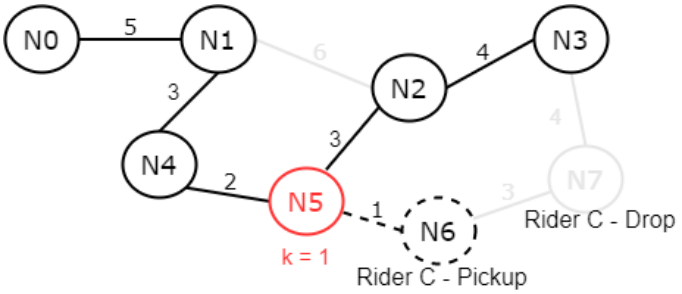

FIGURE 4.6: At Node 5, eligible Rider at Node 6

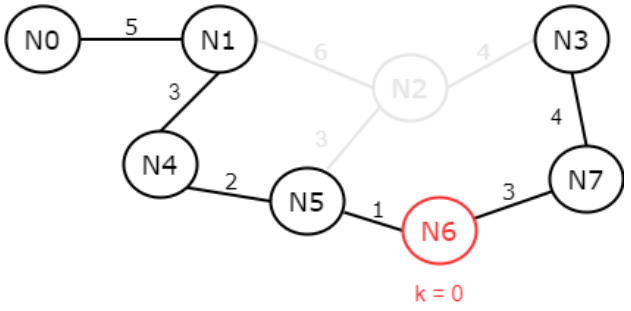

FIGURE 4.7: Path update for Rider C, final path picking up rider B, C

Users	**Output Path**
A	(Node 0,-)→(Node 1,-)→(Node 2,-)→(Node 3,-)
A, B	(Node 0,-)→(Node 1,-)→(Node 4,B)→(Node 5,B) →(Node 2,-)→(Node 3,-)
A, B, C	(Node 0,-)→(Node 1,-)→(Node 4,B)→(Node 5,B) →(Node 6,C)→(Node 7,C)→(Node 3,-)

TABLE 4.2: Output for the Example

Chapter 5

Ride Matching Algorithm Simulation

In this chapter, we present an overview of the GTMobiSIM simulator using which the ride-matching algorithm was tested. Then we discuss and analyze the results of the simulation.

5.1 GTMobiSIM Simulator

GTMobiSIM is a simulator tool for generating mobility traces and query traces for large numbers of mobile agents moving in a road network. The simulator takes a configuration file as input in which we can specify the parameters such as number of agents, duration of the simulation, maximum and minimum speed allowed on a road segment, type of location distribution etc. Each agent starts from a starting location allocated to it and then progressively changing the destination location and the new route accordingly till the simulation lasts.

For our purpose, to test the algorithm, we have modified the simulator codebase.

We integrated the ride-matching algorithm that is originally written in the smart contract to set the routes of the drivers accordingly, and the agents are divided into drivers and riders.

5.2 Environment and Setup

We use the GT Mobile simulator to generate a trace of the drivers and riders moving on a real-world road network, which is obtained from maps available at the National Mapping Division of the USGS. By default the simulator uses the map of Northwest Atlanta region of Georgia that has 6,831 road intersections. The simulator tool is written in Java language and the compilation and testing was done using Eclipse IDE.

5.3 Results and Analysis

In this section we discuss and analyze the results and various metrics for the ride sharing algorithm.

1. We start by changing the number of agents running in the system (X-axis). On the Y-axis we measure the waiting time of each rider (Figure 5.1) i.e. how much each rider has to wait before getting picked up by the designated driver. For each run we keep the number of drivers equal to the number of riders. The maximum allowed distance to travel by a driver is kept constant and is equal to 2 times the shortest distance from start to destination of a driver. The number of credits are initialized randomly between -10 to 10. After sorting the users in the ascending order of the credits, top 50 percent of the users are chosen to be the drivers and the rest act as riders in the system.

It can be observed that increasing the number of agents decreases the average waiting time of the riders, hence proving that the algorithm works well with more number of users in the system.

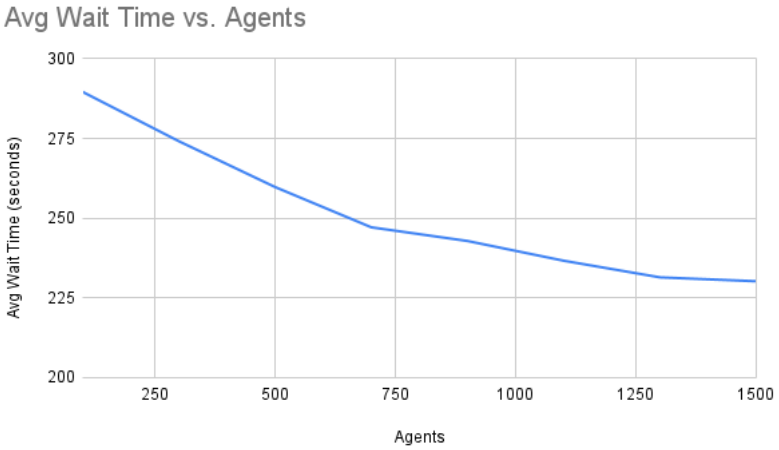

FIGURE 5.1: Average Waiting Time Vs. Number of Agents

2. Percentage of riders picked up by the ride matching algorithm is a good criteria to check whether the algorithms works appropriately. For this, we change the number of agents (X-axis) , and on the Y-axis we measure the percentage of riders that were picked up (were allocated a driver) out of the total eligible riders (Figure 5.2).

   We keep the number of drivers equal to the number of riders on each run. The maximum distance a driver is permitted to travel is fixed at 2 times the shortest distance between his or her starting point and destination. The number of credits is set at a random value between -10 and 10. The top 50% of users are picked to be the drivers, while the rest operate as riders in the system, after sorting the users in ascending order of credit.

   It can be observed that increasing the number of agents increases the percentage of riders getting picked up, which again proves that the algorithm works well with more number of users in the system, hence proving the credibility of the algorithm.

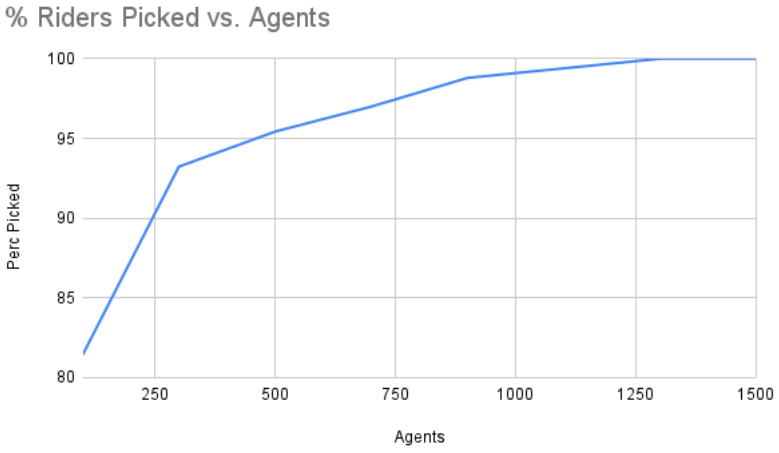

FIGURE 5.2: Percentage Riders Picked Vs. Number of Agents

Percentage Riders Picked Vs. Maximum Distance Travelled

3. In the Figure 5.3, we vary the maximum distance allowed to travel by a driver (in terms of multiple of the shortest/minimum distance between a driver's start and end location) on the X-axis. On the Y-axis we measure the percentage riders picked up. This measure is calculated for different number of agents in the system, represented by different coloured lines in the graph.

   For each run the number of drivers are kept equal to the number of riders. Initialization of the credits is randomized between -10 to 10. After sorting the users in the ascending order of the credits, top 50 percent of the users are chosen to be the drivers and the rest act as riders in the system.

   From the graph it can be observed that increasing the maximum allowed distance increases the percentage of riders picked for any number of agents, since with large distance driver will be able to pick more number of riders. Also comparing the graph among varying number of agents shows that the percentage pickup is more for large number of agents as observed in result no. 2 above.

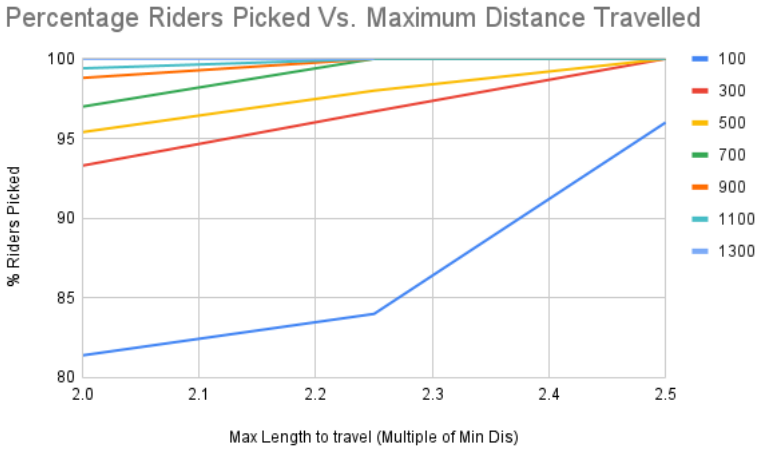

FIGURE 5.3: Percentage Riders Picked Vs. Maximum Distance Travelled

4. Average distance driven by the drivers in the system, and the average driving time of drivers is also a viable metric to judge the algorithm. In the Figure 5.4 we change the number of agents on the X-axis and measure the average driving time of all the drivers in the system. In the figure 5.5 we measure the average driving distance of all the drivers in the system.

   For both the graphs the parameters are kept the same i.e. number of riders equal to number of drivers, the maximum allowed distance equal to the shortest distance of a driver, and the tokens randomly initialized between -10 to 10.

   It can be observed that both the driving distance and driving time decreases as the number of agents increase in the system. Also after a point the rate of decrease, decreases.

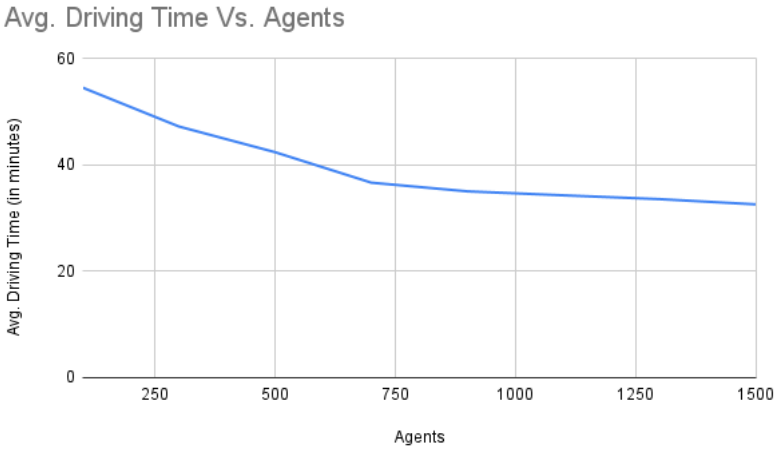

FIGURE 5.4: Average Driving Time Vs. Number of Agents

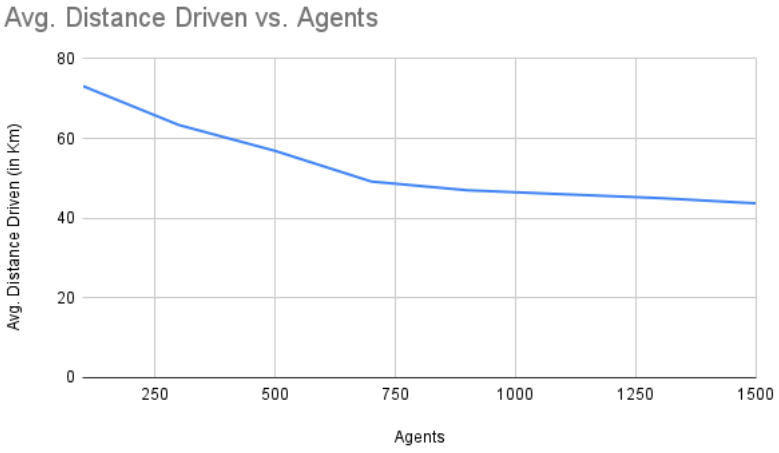

FIGURE 5.5: Average Distance Driven Vs. Number of Agents

Chapter 6

Summary and Future Work

In this chapter, we present a summary of the system described in this project. We also discuss scope for further improvements and optimization that can be done in the current work along with the additional work that we propose in the future.

6.1 Summary

In the thesis, we have proposed a non-monetary decentralized carsharing application using Ethereum Blockchain. It allows the users in the system to share a ride and provide ride to others as well. The system is implemented on Blockchain and hence, all the working and implementation is transparent to its users. The benefits of carsharing would motivate the users to use this platform. We discussed the ride-matching algorithm developed to match the drivers with the potential riders, in order to pickup as many riders as possible and keeping the travelling distance of the driver within a limit. Finally, we showed the implementation done using smart contract and an illustrative example to understand the working of the algorithm.

6.2 Future Scope

The deployment of the smart contract and the contract interactions (transactions) cost the account some gas, which is to be paid in Ethers. The ride-matching algorithm can be optimized further in certain areas to reduce the time and memory complexity of the code, which will reduce the gas cost of the transaction executions (11).

In the future, we plan to develop the proposed work into a Decentralized Application (DApp), providing front-end to the currently existing framework. Developing a user friendly interface will allow users to enter their profile for giving and getting rides and to interact with other users in the system. The DApp will be using a location API to get the physical location of the users, and dynamically execute the ride-matching algorithm accordingly.

Publications

B. Palanisamy S. Chopra and S. Sural. Credit-based peer-to-peer ride sharing using smart contracts. ***IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*****, 2022.**

Bibliography

[1] Ar, S. and Banik, B. (2021). Implementation of a secure ride-sharing dapp using smart contracts on ethereum blockchain. *International Journal of Safety and Security Engineering*, 11:167–173.

[2] Buterin, V. (2013). Ethereum: A next-generation smart contract and decentralized application platform. `https://github.com/ethereum/wiki/wiki/White-Paper`.

[3] Chau, S., Shen, S., and Zhou, Y. (2020). Decentralized ride-sharing and vehicle-pooling based on fair cost-sharing mechanisms. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–11.

[4] De Filippi, P., Wray, C., and Sileno, G. (2021). Smart contracts. *Internet Policy Review*, 10.

[5] Duncan, M. (2010). The cost saving potential of carsharing in a us context. *Transportation*, 38:363–382.

[6] Golosova, J. and Romanovs, A. (2018). The advantages and disadvantages of the blockchain technology. pages 1–6.

[7] Gulian, E., Glendon, I., Matthews, G., Davies, D., and Debney, L. (1990). The stress of driving: A diary study. *Work and Stress - WORK STRESS*, 4:7–16.

[8] Javaid, A. (2013). Understanding dijkstra algorithm. *SSRN Electronic Journal*.

[9] Javits, J., Mary, W., Bracken, B., VanTassel-Baska, J., Bland, L., Stambaugh, T., Gregory, V., Crawford, E., Sutton, E., and Reintjes, C. (2021). *The Greenhouse Effect*, pages 103–108.

[10] Kent, J. (2014). Carsharing as active transport: What are the potential health benefits? *Journal of Transport Health*, 1:54–62.

[11] Khan, M., Sarwar, H., and Awais, M. (2021a). Gas consumption analysis of ethereum blockchain transactions. *Concurrency and Computation: Practice and Experience*.

[12] Khan, S., Loukil, F., Ghedira, C., Benkhelifa, E., and Bani-Hani, A. (2021b). Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-Peer Networking and Applications*, 14.

[13] Lenz, R. (2019). Managing distributed ledgers: Blockchain and beyond. *SSRN Electronic Journal*.

[14] LO, S., Wang, Y., and Chuen, D. (2021). *Cryptography and Blockchain Technology*, pages 1–72.

[15] Migliore, M., D'Orso, G., and Caminiti, D. (2020). The environmental benefits of carsharing: the case study of palermo. *Transportation Research Procedia*, 48:2127–2139.

[16] Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*.

[17] Nansubuga, B. and Kowalkowski, C. (2021). Carsharing: a systematic literature review and research agenda. *Journal of Service Management*, 32:55–91.

[18] Pal, P. and Ruj, S. (2019). Blockv: A blockchain enabled peer-peer ride sharing service. pages 463–468.

[19] Pernice, I. and Scott, B. (2021). Cryptocurrency. *Internet Policy Review*, 10.

[20] Ruch, C., Lu, C., Sieber, L., and Frazzoli, E. (2020). Quantifying the efficiency of ride sharing. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–6.

[21] Saurabh, N., Rubia, C., Palanisamy, A., Koulouzis, S., Sefidanoski, M., Chakravorty, A., Zhao, Z., Karadimce, A., and Prodan, R. (2021). The articonf approach to decentralized car-sharing. *Blockchain: Research and Applications*, page 100013.

[22] Sánchez, D., Martínez, S., and Domingo-Ferrer, J. (2016). Co-utile p2p ridesharing via decentralization and reputation management. *Transportation Research Part C: Emerging Technologies*, 73.

[23] Tripathy, S., Aggarwal, M., and Chakraborty, S. (2021). Beyond uber and lyft: A decentralized cab consortium over blockchains. pages 97–102.

[24] Vázquez, E. and Landa-Silva, D. (2021). Towards blockchain-based ride-sharing systems. In *ICORES*, pages 446–452.

[25] Zheng, G., Gao, L., Huang, L., and Guan, J. (2021). *Ethereum Smart Contract Development in Solidity*. Springer.