BLOCKCHAIN FOR INCENTIVE INSURANCE AND ITS RELATED PAYMENT SERVICES FOR AUTOMATED AND CONNECTED VEHICLES

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech

in

COMPUTER SCIENCE AND ENGINEERING

By

Roshan Singh(Gau-C-15/L-297)
Gwmsrang Muchahary(Gau-C-15/069)
Mridutpal Lahon(Gau-C-15/L-300)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CENTRAL INSTITUTE OF TECHNOLOGY KOKRAJHAR-783370

BONAFIDE CERTIFICATE

This is to certify that the project titled **BLOCKCHAIN FOR INCENTIVE INSURANCE AND ITS RELATED PAYMENT SERVICES FOR AUTOMATED AND CONNECTED VEHICLES** is a bonafide record of the work done by

Roshan Singh(Gau-C-15/L-297)
Gwmsrang Muchahary(Gau-C-15/069)
Mridutpal Lahon(Gau-C-15/L-300)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the **CENTRAL INSTITUTE OF TECHNOLOGY, KOKRAJHAR,** during the year _____.

Pranav Kumar Singh
Project Supervisor

Dr. Pankaj Pratap Singh
Head of the Department

Project Viva-voice held on_____

Examiner _____

Abstract

Providing better experience to the drivers is one of the core objectives of implementing Intelligent Transportation System(ITS) related services in VANETs. ITS aims to provide a range of services for making the life of drivers easier. However, the current approach of implementing these services are centralized which makes the stakeholders trust a central authority and are often prone to single point of failure. Blockchain Technology along with the smart contracts can help address such issues. In this paper we study the scope of Blockchain Technology in implementing ITS related services. We design, implement and evaluate a decentralized approach for Intelligent Parking System and Intelligent Toll Tax Payment System using smart contracts..

Keywords: VANET ,Blockchain,ITS Services, Smart Contracts.

Acknowledgements

We have taken efforts in this project. However, it would not have been possible without the kind support and help of Department of Computer Science and Engineering (Central Institute of Technology, Kokrajhar).

We are highly indebted of (Sir) Pranav Kumar Singh (Asst. Professor) Department of Computer Science and Engineering, for his kind guidance and constant supervision as well as for providing necessary information regarding the project. Sir, has been highly motivating throughout the work and always acted as a torch bearer for leading us to make the project a success.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Vehicular Ad Hoc Network (VANET) is a classification of MANET (Mobile Adhoc Network). A typical VANET consists of vehicles, Road Side Units (RSUs) and the Traffic Authority(TA). The vehicles are the mobile nodes which keeps their position changing over time. RSUs are considered to be the infrastructure deployed by the Traffic Authority. RSUs are responsible for providing the nodes a range of services such as ITS related and infotainment services. TA is the authorized entity which is considered to be the supreme authority who oversees the management as well as the maintenance of the infrastructure. TA is also responsible for setting up the initial configurations of the nodes to enable them to participate in the VANET. As the node in the VANET keeps on changing its position over time it comes in the range of different RSUs. These nodes are equipped with sophisticated hardware device known as On Board Units (OBUs) which let them communicate with other mobile nodes as well as the infrastructure on the network. Based on the location where the operations are performed the working of the VANET can be categorized into two planes:

1.1 Vehicular Plane

The vehicular plane consists of mobile nodes or the vehicles. The network topology at the vehicular plane keeps on changing. At the vehicular plane the vehicles communicate with each other in order to share certain information. The vehicular plane also includes entities other than the vehicles such as a pedestrian or a cyclist. The sharing of the information is achieved by the means of exchange of certain messages. Depending upon the criticality of the exchanged messages the messages can be categorized as

1.1.1 BSM

BSM stands for Basic Safety message. These are the core V2V messages exchanged by the vehicles with their peers to inform them about an urgent prevailing situation around its neighbourhood. An example of a BSM is Forward Collision Warning (FCW) message. FCW is an active safety warning feature that warn the drivers for an imminent frontal collision. The alert is raised when a vehicle comes too close in front of another vehicle.

1.1.2 PSM

PSM stands for Personal Safety Message. These are the messages broadcasted by the Vulnerable Road User (VRU) devices to indicate their presence on the road.PSM helps to avoid accidents where the VRU is beyond the visibility of the driver of an approaching vehicle.

1.2 RSU Plane

The RSU plane consists of the RSUs and the infrastructure deployed by the TA. At the RSU plane each RSU is connected with each other with the help of dedicated lines. The communication among these RSUs are considered to be secure and fast. The main goal of the deployment of these RSUs by the TA is to provide instant access to information to the vehicles operating at the vehicular plane. The propagation of information at this plane is considered to be fast. Also, the TA can directly communicate with the entities on this plane.

1.3 Communications

Depending upon the entities participating in the communication process in the VANETs the entire communication domain can be modelled as:

1.3.1 V2V (Vehicle to Vehicle Communications)

The V2V communication takes place at the Vehicular plane. The entities participating in V2V are the only the vehicles. The vehicles communicate with each other with the help of a standard protool named DSRC.

DSRC or the Dedicated Short Range Communication is a short range to medium range wireless communication channel specially designed for vehicular communications.It works at 5.9 GHz band. The DSRC can operate within the range of 10 - 1000 meters. It provides a

high bandwidth rate suitable for implementation for fast data dissemination at the Vehicular plane.

1.3.2 V2I (Vehicle to Infrastructure)

The V2I communications takes place between the vehicles at the vehicular plane and among the RSUs/Infrastructures working at the RSU plane. Like V2V here also the communication medium is wireless, however the protocol to be implemented varies upon the requirements of the deployer. Now a days communication technologies such as LTE and 5G are being tested for as choice of protocol for V2I communications. An example for V2I communication can be a vehicle requesting for an infotainment service to the RSU.

1.3.3 V2X (Vehicle to Anything)

The V2X communication takes place between the vehicles on the road and any other non transportation entity such as a pedestrian. V2X communication can include communication between vehicle and driver smart phone etc.

Due to its long term prospects VANET is turning out to be a hot topic among the research community as well as in the industrial domain. The core objective for implementing VANETs is to make the vehicles communicate with each other in order to reduce the number of road accidents, improve the traffic efficiency and to provide drivers a better driving experience. The current approaches of providing the ITS related services are neither efficient nor reliable.

With this in mind, a blockchain based approach for Decentralized Intelligent Parking System and Decentralized Intelligent Toll tax payment is developed in this work. The rest of the work is organized as follows. We first define our problem statement. We then discuss the related work carried out in the domain till date. We continue with giving a brief introduction to blockchain technology the underlying technology used in our work and describing our proposed mechanism, the evaluation of the results of the experiments performed on our developed model. And finally conclude with discussing future prospects with our developed model.

1.4 Motivation and Objectives

The objective of the work performed in this project is motivated by the need to have a decentralized and stable ITS related services scheme for VANETs. The earlier approaches are mainly node centric where a single authority was responsible for providing the services and managing the happening in order to provide that service in VANET. Such schemes make the entire system dependent on a single entity which is not monitored by any other entity. This can lead to an ample scope of discrepancies that could happen in the system if that centralized authority becomes un available or does not provide the services.

Our work tries to address this challenge by proposing a blockchain based decentralized approach for implementing ITS related services in VANETs. In our work we implement a use case of an ITS related service to be used by the vehicles.

1.5 Blockchain

Blockchain can be considered as one of the cutting edge technology of 21st century. Blockchain is a new technology which is based upon some core principles in Computer Science and Mathematics that are in existence from past hundreds of years such as cryptography. A blockchain as its name suggest is essentially a sequence of blocks where each block is linked with its previous block with the help of hashes. A block is formed of zero or more data elements known as transactions. The transactions are signed by the entity executing it. PKI is used for signing in and verifying the transactions on the blockchain. A blockchain is maintained by a set of nodes where each node maintain its own local copy of the chain. These nodes communicate with each other with the help of P2P connections. The blockchain provides some key features which makes it a bit unique from other technologies in Computer Science.One of them is decentralization a blockchain can never be owned or can be dominated by a single entity. Theoretically, it is possible but practically it is not. To own a blockchain one has to get control over at least 51 percent of the nodes maintaining the chain. Second key feature is immutability data once written on the chain is immutable and cannot be erased. Blockchain can be used to establish trust among a set of untrusted nodes. The blockchain assumes the nodes in the network to be untrusted. It utilises a mechanism to make all these untrusted nodes to agree on a same state. This mechanism is known as consensus. The consensus mechanism may vary depending upon the type of blockchain to be considered. In case of public blockchain where anyone can come and be a part of the network the common consensus mechanism used is Proof of Work (PoW). There exists

certain special nodes on the network which collects the transactions coming from various clients on the network and bundle them together to form a block. These miners need to perform a computationally complex operation to get their block on to the chain. This computationally complex problem is termed as PoW. It is difficult to generate but easy to check. Since the miners invest their computational resources for mining the blocks they also get incentivised by the network in terms of mining rewards. Depending upon the type of the blockchain the consensus algorithm to be used varies. It is suggested to use challenge response based consensus algorithms such as PoW for public blockchains where the entities are unknown and unauthenticated. Consensus algorithms such as PBFT and PoA can be used in case of private blockchains. Each consensus algorithm have their own advantages and drawbacks. Such as the PoW algorithm can scale well in terms of the number of nodes, however it provides a very low throughput in terms of transaction processing. In the bitcoin blockchain the average throughput is 7 tx/s. On the other hand algorithms like PBFT can scale pretty well in terms of transaction throughput however it fails miserably when coming to scalability in terms of number of nodes.

Based upon the read and write permissions on the chain a blockchain can be categorised into four categories:

1.5.1 Public Blockchain:

A public blockchain is a blockchain which is open to everyone. Anyone can read the data on the chain. Anyone can join the network and can download full copy of the entire network and can run node in their local devices. Anyone in the world can send transaction to anyone and can see transactions details in public explorer.
Examples: Bitcoin, Litecoin, Monero etc.

1.5.2 Permissionless blockchain:

A permissionless blockchain is a blockchain which does not have any restriction on writing data onto it. Anyone is allow to create their own address and can begin interact with the network by submitting a transaction on it.

1.5.3 Private Blockchain:

A private blockchain only permits a set of users to view the data present on the chain. A private blockchain is run by specific the member of the consortiums and companies

1.5.4 Permissioned Blockchain:

A permissioned blockchain is a blockchain which puts a set of restrictions in order to decide which set of entities will be able to write data onto the chain.

The first application of the blockchain technology was the cryptocurrency Bitcoin [1]. Bitcoin is a peer to peer cryptocurrency which lets user transact on a peer to peer network in an unrestricted manner. Bitcoin is a public, permissionless blockchain which allows anyone to join and transact on it. Blockchain platforms such as Bitcoin are meant for dedicated purpose that means they can be used only for those tasks for which they are made. Like the Bitcoin blockchain only supports the financial transactions which involves transferring of bitcoins from one account to another. And such platforms also doesn't provide much programming flexibility in terms of writing and executing turing complete scripts.

However, there exists blockchain platforms such as Ethereum and Hyperledger which permits users to write and deploy their own turing complete codes.

1.6 Smart Contracts

The smart contracts [2] are a bunch of self-executable code sitting on top of a blockchain. A smart contract consists of a well defined set of rules/condition and a set of actions. The conditions specify the actions to be performed when it turns out to be true. The property of self execution makes a smart contract much more unique than any other programming construct. A smart contract works on the data present on the blockchain. A smart contract is also immutable that means once deployed on the chain the rules of the contract cannot be changed. This property of smart contract ensures bias free decisions and helps to minimize the need of trust. It can be used as a tool for automating decision making process without the need of an intermediary in a blockchain environment.

1.7 Ethereum Framework

Ethereum [3] is a public blockchain platform. Unlike Bitcoin blockchain Ethereum supports smart contracts. That means snippets of code can be written and deployed on Ethereum blockchain. The supportability of smart contracts makes the platform ideal for developers who want to implement their smart contract for certain problem specific use case. The

Ethereum blockchain is based upon a State Transition based model, where each node executes a set of transactions in the same order to reach a common consensus. Ether is a cryptocurrency, which is used as a medium of exchange on the Ethereum blockchain. The platform uses the notion of GAS(a unit of measuring the computational complexity) and GAS LIMIT while processing the transactions.The idea of GAS is used to curb the menace of an attacker who can embed an infinite loop in his smart contract and executes a transaction. In the absence of GAS and GAS LIMIT it will lead to a halting problem in the blockchain and may result in a crash of the chain. The Ethereum blockchain supports two types of accounts:

1.7.1 Externally Owned Accounts:

These are the accounts owned by the users on the blockchain. A externally owned account is associated with a set of public/private key pairs. These are used by the user to sign and verify the transactions.

1.7.2 Contract Accounts:

These are the accounts where the code of the smart contract resides. An externally owned account can send and receive ether to and from other externally owned accounts or contract accounts. The contract accounts cannot do anything of its own. It depends upon the transactions executed from the externally owned accounts which triggers the execution of the code in present in the contract account.

Chapter 2

Related Work

There exists some work related in the field of Intelligent Transportation Systems [4][4] [5] [6]. In[7] authors propose a Machine Learning based , Histogram Oriented Gradient in Intelligent Transportation System approach for alerting drivers about frontal presence of obstacles as well as pedestrian. A J2EE based architecture GIS application is proposed in [8] for fast access of web pages in VANETs.

However, approaches using the decentralized technologies such as blockchain is very much deficient in this domain. We could only find a paper [9] where the authors utilized a decentralized technology for implementing an ITS related services. However, their system was not fully automated and was dependent on manual intervention under certain scenarios. We could not find any resource specific to our problem domain. We consider that much more contributions are needed in this domain using such decentralized technologies. To, this end our approach will be one such effort.

Chapter 3

Proposed mechanism

3.1 System Architecture:

The proposed system architecture to facilitate our blockchain based strategy for implementing incentive and ITS related services is shown in Figure 3.1. We have considered two major plane of the architecture those are the Vehicular plane and the RSU plane. The Vehicular plane consists of smart vehicles. The RSU plane includes Traffic Authority (TA), RSUs and ITS services.

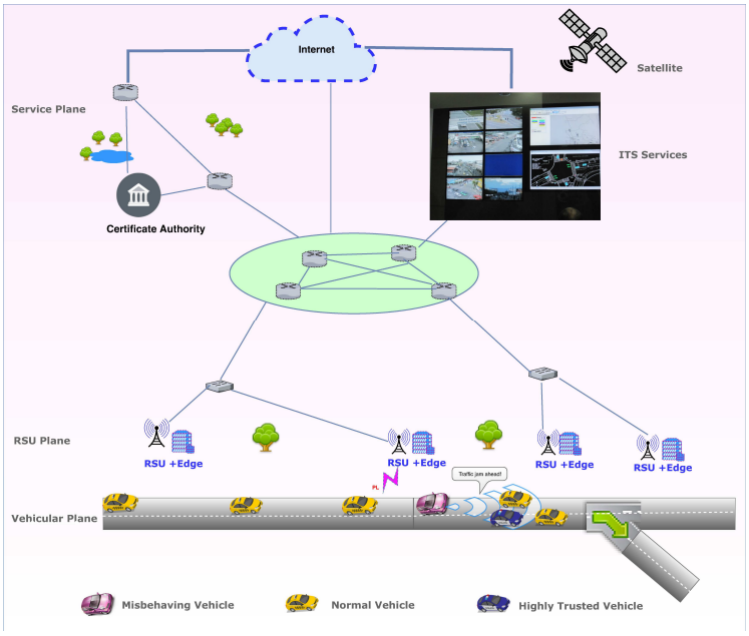


Figure 3.1: Proposed System Architecture

3.2 Framework

The system framework of our implemented system is as shown as in Figure 3.2. The TA deploys the contract on the blockchain. RSUs interact with the contract via Command Line Interface. RSU uses Node.js API for communicating with the blockchain. On the other

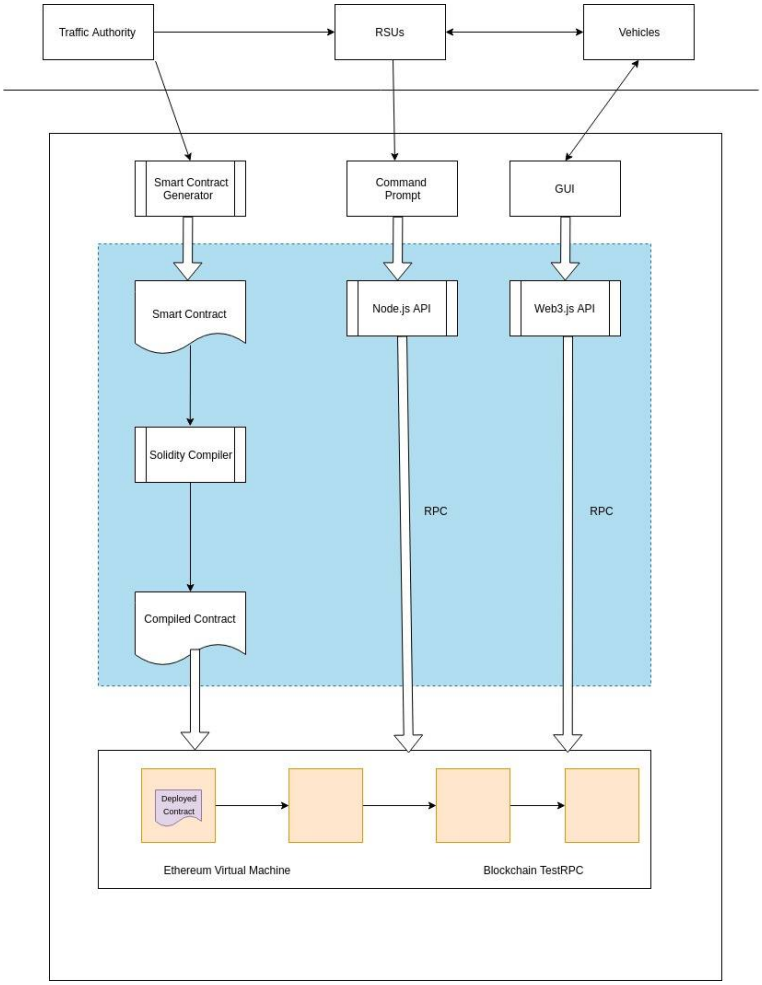


Figure 3.2: System framework of our implemented system

hand vehicles are provided with Graphical User Interface. They communicate with the blockchain with the help of Web3.js API.

3.3 Implementation

3.3.1 Decentralized Intelligent Parking System (DIPS)

To fulfill the objective of implementing a DIPS, our smart contract stores the details of available parking zones in an area. A vehicle can search for available parking zone in a particular area by using the PIN/ZIP code associated with that particular area. We have used the PIN code as an unique identifier here.

```
function configureParkingZone(address parkingZoneID,uint totalCapacity)
    onlyTrafficAuthority public {
        PZ[parkingZoneID].rate = 1 ether;
        PZ[parkingZoneID].rateValidUpto = now + 1000;
        PZ[parkingZoneID].totalCapacity = totalCapacity;
        PZ[parkingZoneID].availableLots = totalCapacity;
        PZ[parkingZoneID].cancellationFees = 5;
        PZ[parkingZoneID].parking_Zone_Manager = parkingZoneID;
}
```

Figure 3.3: Code Snippet for configureParkingZone

The Traffic Authority (TA) adds a Parking Zone into the blockchain by calling the method *configureParkingZone* as shown in figure 3.3 . A vehicle can poll the blockchain for getting the list of available parking zones available in a particular area. Our contract provides a getter function called *checkAvailiability* for this.

The method also displays the rate of parking charged by each of these parking zones. A vehicle can book a lot in the parking zone by calling the function *bookMyLot* as shown in Figure 3.4. Figure 3.5 shows the process flow involved in booking a lot.

On being called method results in a successful transaction if there exists enough parking lots available in the parking zone, otherwise the transaction gets reverted. In case of a successful transaction the vehicle is allocated with a lot identifier determining the parking lot allocated to it in a given parking zone. The smart contract maintains a counter to specify the duration of time the parking lot was occupied by the vehicle. Each parking lot is equipped with sensing devices which senses the ID of the vehicle trying to park in the lot. If the address of the parking vehicle matches with the address of the vehicle which has booked the lot then only the vehicle is allowed to park itself in the lot otherwise not. The

```
function bookMyLot(address parkingZoneID) public{
    if(VR[msg.sender].hasBooked == false){
    for( uint i=0; i<PZ[parkingZoneID].totalCapacity;i++){
        if(Status[parkingZoneID][i].isOccupied == false){
            Status[parkingZoneID][i].reservedBy = msg.sender;
            VR[msg.sender].hasBooked = true;
            VR[msg.sender].bookedPZAddress = parkingZoneID;
            VR[msg.sender].booked_Lot_No = i;
            VR[msg.sender].bookedAtTimestamp = now;
            PZ[parkingZoneID].availableLots = PZ[parkingZoneID].availableLots - 1;
            Status[parkingZoneID][i].isOccupied = true;
            VR[msg.sender].cancellationFees = PZ[parkingZoneID].cancellationFees;
            i = PZ[parkingZoneID].totalCapacity + 5;
        }
    }
}else{
    revert();
}
}
```

Figure 3.4: Code Snippet for bookMyLot

counter starts when the vehicle books a lot and gets over when the vehicle exists the lot. The vehicle can exit a lot by calling the function *exitParking* as shown in Figure 3.6

Upon resulting in a successful transaction the counter associated with the vehicle gets off and the parking lot occupied by the vehicle in the contract gets free. The vehicle can now pay the parking bill. The vehicle first checks the bill incured for parking by calling the function *parkingBill* as shown in Figure 3.7. It displays the amount of ether to be paid as a bill. The parking bill is calculated with the formula:

$$ParkingBill = Durationof Parking * Rateof Parking \tag{3.1}$$

After getting the parking bill the vehicle can pay the bill by calling the method *payMyBill* as the amount to be paid displayed by the function. It is to be noted that if a vehicle exists a parking lot and does not pays the bill the vehicle will be unable to book further more slots in some later point of time.

Our developed smart contract for DIPS also provides the facility of cancelling a lot booked on the blockchain. It provides a method called *cancelBooking* as shown in Figure 3.8 for this. However, to deal with the notorious vehicles trying to book and cancel the lots uselessly we have taken certain preventive measures.

- A booked slot can only be cancelled by a vehicle who has booked it. We specify a threshold time that specify the duration during which the cancellation of the booked

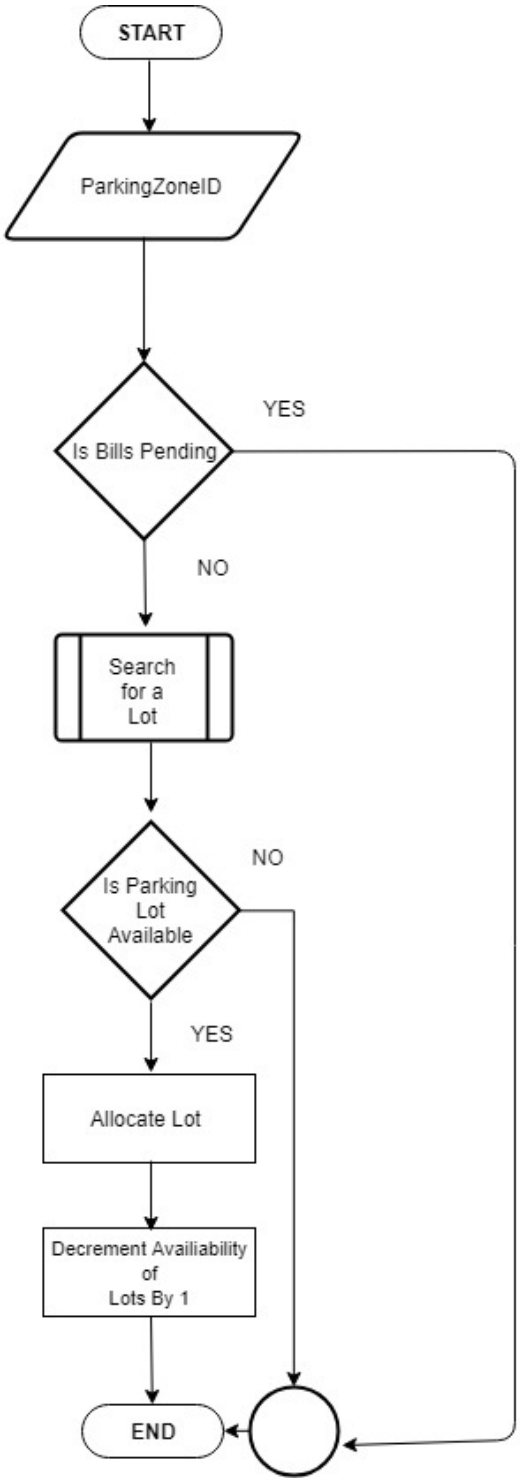


Figure 3.5: Flowchart for booking a parking lot

```
function exitParking() hasBooked payable public {
    VR[msg.sender].parkingFees = (now - VR[msg.sender].bookedAtTimestamp)*
        ((PZ[VR[msg.sender].bookedPZAddress].rate)/10000000000000000);
    Status[VR[msg.sender].bookedPZAddress][VR[msg.sender].booked_Lot_No].isOccupied = false;
    PZ[VR[msg.sender].bookedPZAddress].availableLots = PZ[VR[msg.sender].bookedPZAddress].availableLots + 1;
}
```

Figure 3.6: Code Snippet for exitParking

```
function payMyBill() payable public {
    assert(msg.value == (VR[msg.sender].parkingFees));
    if(msg.value != (VR[msg.sender].parkingFees)){
        revert();
    }
    address receiving_ac = VR[msg.sender].bookedPZAddress;
    receiving_ac.transfer(msg.value);
    VR[msg.sender].hasBooked = false;
}
```

Figure 3.7: Code Snippet for payMyBill

lot can be made. There exists no scope of cancelling a booked lot at some later point of time once the threshold duration has passed.

- To discourage any vehicle for frequent cancelling of lots we also imposed a cancellation charge.

Our DIPS smart contract also facilitate the Parking Lot Manager to adjust the parking charges. However, to curb the menace of any greedy Parking Lot Manager who frequently increase as well as decrease the rates, we have taken certain preventive measures here.

- To discourage any vehicle for frequent cancelling of lots we also imposed a cancellation charge.

A Parking Lot Manager has to notify the changes in the rates from a particular date before implementing the changes. If the Parking Lot Manager has not notified earlier he will not be able to implement the changes.

We developed a Web based application for the vehicles to interact with the DIPS. Figure c shows the web based interface at the vehicle end for booking a parking lot. The drivers can book a slot by using the Web based interface. A web based real time portal was also developed for the Parking Lot Manager to view the status of its parking lot.

```
function cancelBooking() isCancellationTimeExpired isBooked payable public {
    assert(msg.value == VR[msg.sender].cancellationFees);
    if(msg.value != VR[msg.sender].cancellationFees){
        revert();
    }
    address receiving_ac = VR[msg.sender].bookedPZAddress;
    receiving_ac.transfer(msg.value);
    VR[msg.sender].hasBooked = false;
    Status[VR[msg.sender].bookedPZAddress][VR[msg.sender].booked_Lot_No].isOccupied = false;
    PZ[VR[msg.sender].bookedPZAddress].availableLots = PZ[VR[msg.sender].bookedPZAddress].availableLots + 1;
}
```

Figure 3.8: Code Snippet for cancelBookings

3.3.2 Decentralized Intelligent Tolltax Payment System (DITPS)

We also implemented a decentralized toll tax payment system, a ITS related service . Private blockchain handle data flow based on the smart contract or authorized policies set up by the TA. Like our previous implementation here also we use the cryptocurrency ether which is provided in the core of the ethereum client as a medium of exchange.Incentives earned by the smart vehicles can be redeemed as ether which can be utilized by them for accessing various ITS related services. The procedure for our implemented ITS service is shown in Figure 3.11. The vehicle identity is fetched from the vehicle with some sensors present at the toll crossing. The device installed at toll crossing checks the block-chain for assuring that the vehicle has paid the tax, if yes the indicator installed at the toll crossing go green indicating the vehicle to cross.

The system architecture of our DTPS is as shown in Figure.

3.4 Programming Constructs

One of the characteristics of a smart contract is its immutable nature. It's impossible to make any kind of modification in a deployed contract. This fact is often overlooked by many smart contract developers which in turn results in heavy financial losses. In our case also management of incentives can be considered as management of valuable asset which are valuable for the vehicles as far as their credit score is concerned.

In order to make our smart contract bug free as well as secure we have developed the contract as per as the guidelines provided in the official documentation of the language used. We have also considered core principles of best practices in the development of smart contracts.

The code of the contract has been tested a number of times. A major portion of the project time line was devoted for this purpose. Both static as well as run time analysis were

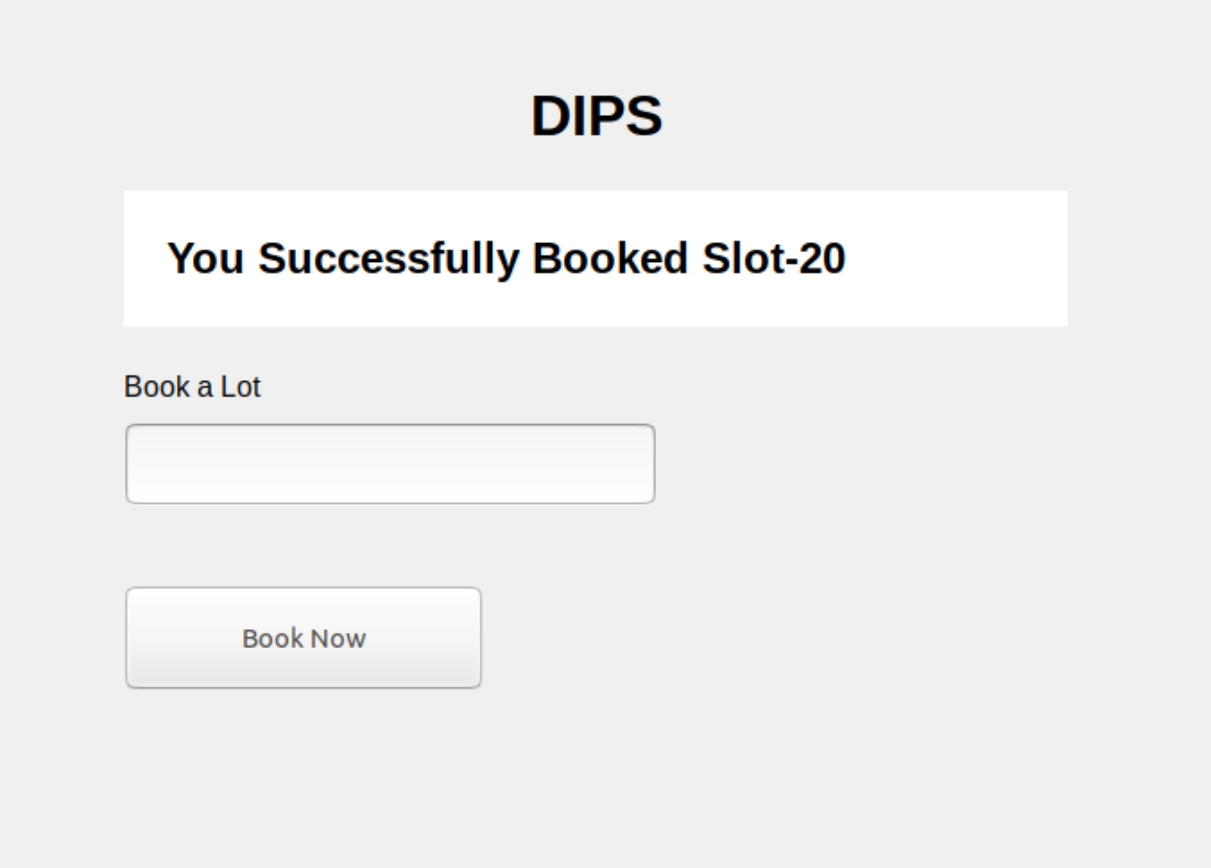


Figure 3.9: Web Interface for Parking lot booking

done. The detected bugs were fixed and appropriate access restrictions were kept in place where necessary. Some of the modifiers used in our smart contract are as follows :

For instance the modifier shown in Figure 3.12 specifies that only a Traffic Authority can execute a function where the modifier onlyTrafficAuthority is imposed.

3.5 Our approach v/s Centralized approach

- Transparency and Immutability: All the proceedings either it may be booking a parking lot or paying the toll tax are commited on the blockchain as in the form of mined transactions. With, our developed scheme, it is impossible for the vehicle to cross the toll without making a payment.Similarly, it is also not possible for the Parking Lot Manager to deny the fact that a vehicle has not paid for parking or to charge higher rates of parking. All the code in the smart contract are visible publicly. So anyone can check it.

```
function payTollTax(address TollBooth) smartVehicle public payable{
    if(now - TollTaxRegister[TollBooth][msg.sender].timestamp > 1000 ||
        TollTaxRegister[TollBooth][msg.sender].paid == false){
            assert(msg.value == Toll[TollBooth].charge)
            if(msg.value != Toll[TollBooth].charge){
                revert();
            }
            TollTaxRegister[TollBooth][msg.sender].timestamp = now;
            TollTaxRegister[TollBooth][msg.sender].paid = true;
    }
}
```

Figure 3.10: Code snippet for payTollTax

- Availability: As compared to centralized approach our approach always guarantee of a availability as long as a full node exist in the system. Traditional centralized approaches does not guarantee this characteristic. This ensures that one can always do a transaction using our scheme.
- Trust : In traditional approaches the parties involved in a transaction needed to trust each other. These parties are often highly untrusted and tries to cheat each other. Like the Parking Lot Manager always aims to charge a higher rate of parking to increase his profit. We minimize this trust relationship to a maximum extent. Saying, truly we have just removed the need of trust here.

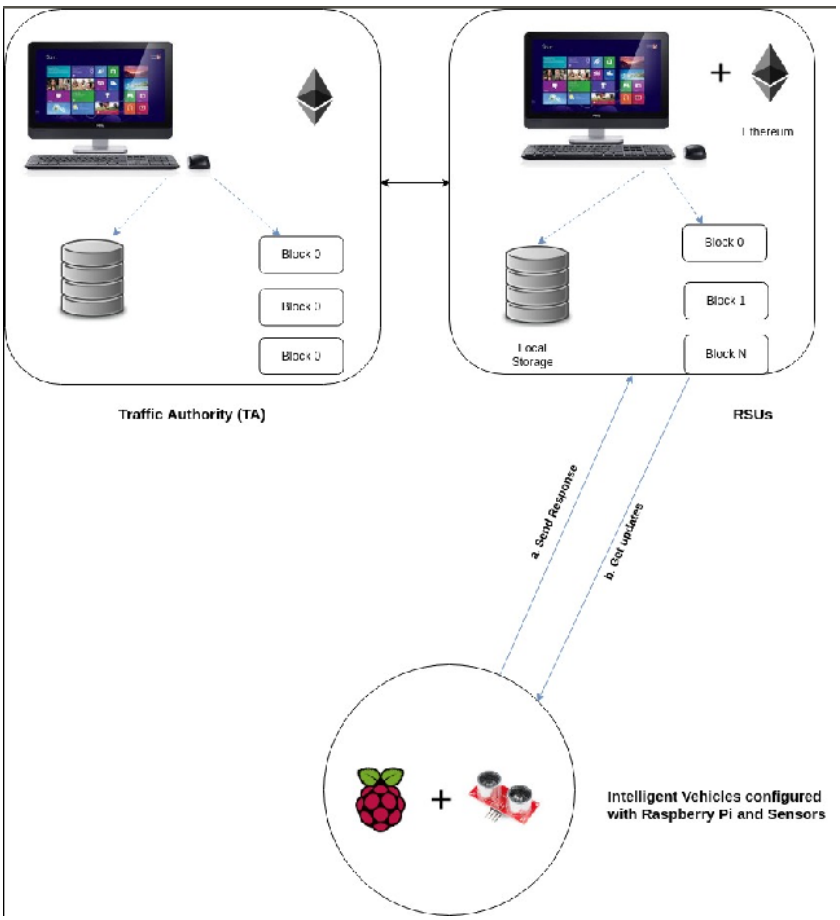


Figure 3.11: Caption

```
modifier onlyTrafficAuthority{
    bool auth = false;
    if(msg.sender == trafficAuthority_Address){
        auth = true;
    }
        require(auth == true,"You are not the Traffic Authority");
        _;
}
```

Figure 3.12: A sample access modifier

Chapter 4

Prototype Testbed

To implement our decentralized approach for DIPS and DITPS with the help of smart contract, we use a private permissioned Ethereum blockchain. The backbone connectivity between the RSUs are wired network. The connectivity between Vehicles and RSUs are wireless. For writing and compiling the contract, we used the Remix integrated development environment(IDE) and for Solidity a browser-based IDE. We performed experiments where the Raspberry Pi 3 node (vehicle) sends a set of the transaction of type bookMyLot and payTollTax to our private blockchain platform in an asynchronous manner, i.e., all transactions are sent without waiting for a response from the blockchain. The number of requests has been set to 1, 100, 500, 750 and 1000. The results of the experiment averaged over 3 independent runs. The testbed details are as follows:

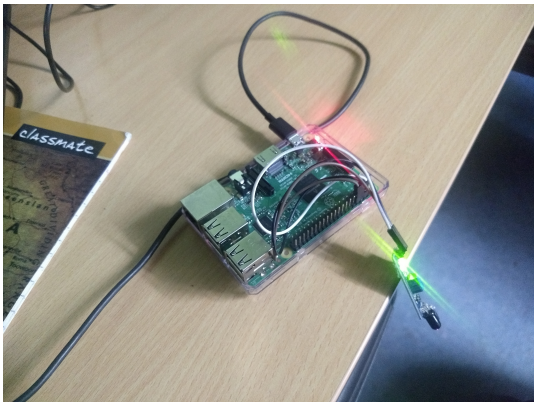


Figure 4.1: Vehicle OBU

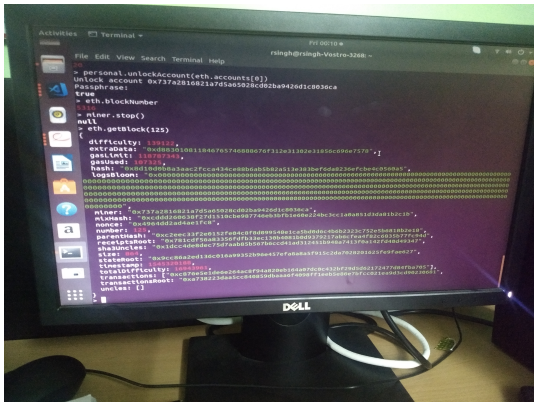


Figure 4.2: RSU

4.0.1 RSU Configuration:

- Intel CoreTM i7-7700 CPU 3.60GHz x 8, 8 GB RAM
- 1 TB hard drive
- Ubuntu 18.04.1 LTS Operating System
- Geth Version : geth 1.8.17-stable

4.0.2 Vehicle OBU Configuration:

- Raspberry Pi 3.3
- Raspbian Operating System
- Geth Version : geth 1.8.18 ARMv7 stable release

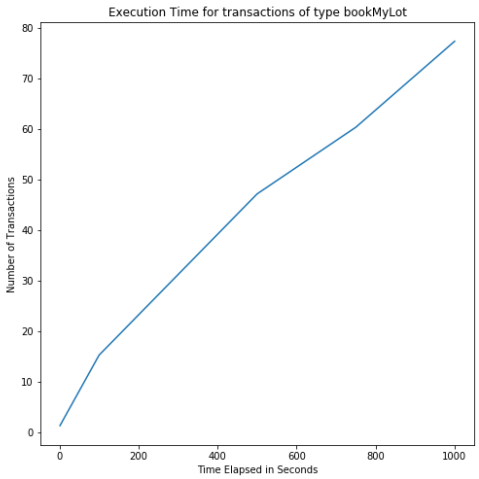
Chapter 5

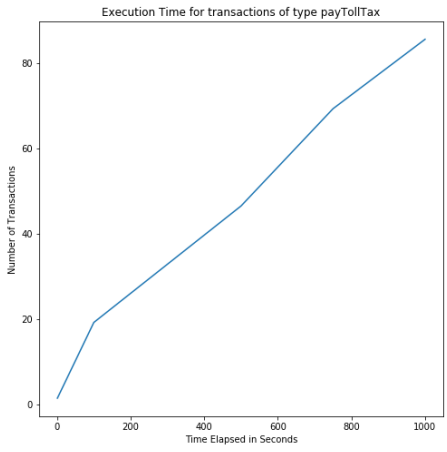
Results and Discussion

We choose transaction execution time as the parameter for the evaluating our set up private blockchain.

- Execution Time: The execution time is the total amount of time (number of seconds) during which the blockchain took to execute and confirm all transactions in the dataset. It is the duration of time elapsed, when the first transaction was deployed to the time when the last transaction is mined. Average execution time for a set of transaction is the execution time averaged over certain independent run.

5.0.1 Comparing Average Execution Time:





We compare the differences in execution time of varying transactions, we executed the transactions in the batch of 1 100 500 750 and 1000 transactions. The execution time grows

as the number of transaction in the dataset increases. For a batch of 1000 transactions the blockchain took 78.33 seconds to execute transactions of type *bookMyLot* and 79.6 for the transactions of type *payTollTax*. The execution time plot for*bookMyLot* is as shown in Figure 5.0.1 and the execution time plot for *payTollTax* is as shown in Figure. 5.0.1

As we can see from the average execution time plot increases exponentially for larger set of transactions. This is a drawback of the used blockchain platform. The Ethereum blockchain fails to grow linearly for a large set of transactions with its default Ethash PoW based Algorithm.

Chapter 6

Conclusion

In this work we have successfully implemented two ITS related services namely DIPS DITPS in a decentralized manner using Blockchain Technology. We tested the implementation and has found it working work. However, due to the consensus algorithm used in our work the performance of the approaches in terms of execution time is not suitable for public grade implementation. This limitation can be resolved by using better and efficient consensus algorithm or different blockchain platform.

Our future work will include implementation of Incentive Mechanism and Insurance processes in VANET using blockchain

Bibliography

[1] S. Nakamoto, Bitcoin : A peer to peer electronic cash system.

[2] N. Szabo, Smart contracts: Building blocks for digital markets.

[3] V. Buterin, A next generation smart contract decentralized application platform.

[4] L. H. Jiang Meiying, N. Lede, The evaluation studiesn of regional transportation accessibility based on intelligent transportation system.

[5] S. Fernandez, I. Takayuki Ito, member, Driver classification for intelligent transport system using fuzzy logic.

[6] J. y. Liu Yang, Big data technology and its analysis of application in urban intelligent transportation system (2018 International Conference on Intelligent Transportation, Big Data Smart City).

[7] LiFei, LiDengLin, Pedestrian detection based on histogram of oriented gradient in intelligent transport system.

[8] X. Lu, Web based public participation gis service for intelligent transportation information collection.

[9] P. L. Yuichi Hanada Luke Hsiao, Smart contracts for machine-to-machine communication: Possibilities and limitations.

Appendix

DATASET FOR EXECUTION TIME OF TRANSACTION SET OF TYPE *bookMyLot*		
Number of transactions	Execution Time in Seconds	Average Execution Time in Seconds
1	1.247	1.242
	1.254	
	1.245	
100	15.271	15.262
	15.311	
	15.197	
500	47.097	47.108
	47.116	
	47.113	
700	60.411	60.326
	60.217	
	60.113	
1000	77.416	77.327
	77.255	
	77.311	

DATASET FOR EXECUTION TIME OF TRANSACTION SET OF TYPE *payTollTax*		
1	1.431	1.422
	1.421	
	1.416	
100	17.912	17.944
	18.031	
	17.890	
500	46.501	46.521
	46.577	
	46.470	
700	69.375	69.329
	69.311	
	69.302	
1000	85.716	85.658
	85.601	
	85.659	