TRUSTED TRAFFIC DATA SHARING WITH CONGESTION CONTROL IN INTERNET OF VEHICLES USING BLOCKCHAIN

A PROJECT REPORT

Submitted by

MAGESHBOOPATHY.V	**(422420104019)**
SANTHOSH.A	**(422420104030)**
PRAVEEN.P	**(422420104306)**

in partial fulfilment for the award of the degree
of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"TRUSTED TRAFFIC DATA SHARING WITH CONGESTION CONTROL IN INTERNET OF VEHICLES USING BLOCKCHAIN"** is the bonafide work of **"MAGESHBOOPATHY V (422420104019), SANTHOSH A (422420104030), PRAVEEN P (422420104306)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. L. Jegatha Deborah, M.E., Ph. D,	Mr. R. Manikandan, M.E., (Ph. D),
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Assistant professor,	Teaching Fellow,
Department of CSE,	Department of CSE,
University College of Engineering,	University College of Engineering,
Tindivanam,	Tindivanam,
Melpakkam – 604 001.	Melpakkam – 604 001.

Submitted for the University Examination held on _____________

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We would like to thank our revered Dean, **Dr.P. Thamizhazhagan, M.E., Ph.D.,** for providing infrastructure facilities and wholehearted encouragement for completing our project successfully.

For mostly, we pay our grateful acknowledgement and extend our sincere gratitude to **Dr.L. JEGATHA DEBORAH, M.E., Ph.D.,** Assistant professor and Head, Department of Computer Science and Engineering, University College of Engineering Tindivanam, for extending the facilities of the department towards our project and for unstinting support.

We express our thanks to our guide, **Mr.R. Manikandan, M.E., (Ph.D.),** Department of Computer Science and Engineering, University College of Engineering Tindivanam, for guiding us for every phase of the project. We appreciate her thoroughness, tolerance and ability to share her knowledge with us. We thank her for being easily approachable and quite thoughtful. We owe her harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could never have it to this extent.

We thank all our teaching and non-teaching faculty members of the department and also our fellow friends for helping us in providing valuable suggestions and timely ideas for the successful completion of the project and also, we extend our immense thanks to our parents and siblings who have been a great source of inspiration and rendered us great support during the course of this project work. We sincerely thank all of them.

ABSTRACT

The Internet of Vehicles (IoV) plays a vital role in data sharing and accidents avoidance in transportation systems. Vehicles share the observed data to the RSUs and other vehicles. However, the data provided by the vehicles cannot be trusted due to the presence of hackers, in which fake data leads to dangerous condition. Channel congestion is occurred when excess data is shared resulting in loss of vital messages. To overcome these issues, a blockchain based trusted data sharing mechanism with congestion control is proposed. There are two phases, data sharing and trust evaluation phase. The mechanism contains two steps. First, Kademlia algorithm-based traffic data forwarding method which is used to control the channel congestion. Second, Cuckoo filter-based traffic data deduplication to avoid repetitive sharing of same data by checking local filters in vehicles and RSUs. Here RSUs act as full nodes and vehicles are light nodes in the blockchain. The trust evaluation is done from the vehicles. Atlast, we develop a trust management prototype system with congestion control which is much effective.

Keywords: Internet of Vehicles, Trust Management, Blockchain, Data sharing, Congestion control.

திட்டப்பணிச்சுருக்கம்

வாகனங்களின் இணையம் (IoV) தரவுப் பகிர்வு மற்றும் போக்குவரத்து அமைப்புகளில் விபத்துகளைத் தவிர்ப்பதில் முக்கிய பங்கு வகிக்கிறது. வாகனங்கள் கவனிக்கப்பட்ட தரவை RSUகள் மற்றும் பிற வாகனங்களுடன் பகிர்ந்து கொள்கின்றன. இருப்பினும், ஹேக்கர்கள் இருப்பதால் வாகனங்கள் வழங்கும் தரவுகளை நம்ப முடியாது, இதில் போலியான தரவு ஆபத்தான நிலைக்கு இட்டுச் செல்கிறது. அதிகப்படியான தரவு பகிரப்படும்போது சேனல் நெரிசல் ஏற்படுகிறது, இதன் விளைவாக முக்கிய செய்திகள் இழக்கப்படுகின்றன. இந்தச் சிக்கல்களைச் சமாளிக்க, நெரிசலைக் கட்டுப்படுத்தும் பிளாக்செயின் அடிப்படையிலான நம்பகமான தரவுப் பகிர்வு பொறிமுறை முன்மொழியப்பட்டது. தரவு பகிர்வு மற்றும் நம்பிக்கை மதிப்பீடு கட்டம் என இரண்டு கட்டங்கள் உள்ளன. பொறிமுறையானது இரண்டு படிகளைக் கொண்டுள்ளது. முதலில், காடெம்லியா அல்காரிதம் அடிப்படையிலான டிராஃபிக் டேட்டா ஃபார்வர்டிங் முறை, இது சேனல் நெரிசலைக் கட்டுப்படுத்தப் பயன்படுகிறது. இரண்டாவதாக, வாகனங்கள் மற்றும் RSU களில் உள்ள உள்ளூர் வடிப்பான்களைச் சரிபார்ப்பதன் மூலம் ஒரே தரவை மீண்டும் மீண்டும் பகிர்வதைத் தவிர்க்க குக்கூ வடிகட்டி அடிப்படையிலான போக்குவரத்து தரவுக் குறைப்பு. இங்கே RSU கள் முழு முனைகளாக செயல்படுகின்றன மற்றும் வாகனங்கள் பிளாக்செயினில் ஒளி முனைகளாக உள்ளன. வாகனங்களில் இருந்து நம்பிக்கை மதிப்பீடு செய்யப்படுகிறது. அட்லாஸ்ட், நெரிசலைக் கட்டுப்படுத்தும் நம்பிக்கை மேலாண்மை முன்மாதிரி அமைப்பை நாங்கள் உருவாக்குகிறோம், இது மிகவும் பயனுள்ளதாக இருக்கும்.

முக்கிய வார்த்தைகள்: வாகனங்களின் இணையம், நம்பிக்கை மேலாண்மை, பிளாக்செயின், தரவு பகிர்வு, நெரிசல் கட்டுப்பாடு.

TABLE OF CONTENTS

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF ABBREVATIONS

S. NO.	ABBREVATIONS	DEFINITIONS
1	IoV	Internet of Vehicles
2	ITS	Intelligent Transportation System
3	IoT	Internet of Things
4	RSU	Road Side Unit
5	CSP	Cloud Service Provider
6	CF	Cuckoo Filter
7	UML	Unified Modelling Language
8	DFD	Data Flow Diagram

CHAPTER 1
INTRODUCTION

The advent of the Internet of Vehicles (IoV) has ushered in a new era of intelligent transportation systems, revolutionizing the way vehicles communicate and share data on the road. In the IoV ecosystem, vehicles, infrastructure, and various sensors collaborate to provide real-time information that has the potential to enhance road safety, reduce traffic congestion, and pave the way for more efficient and sustainable transportation. However, as this interconnected web of vehicles and data sources grows, ensuring the trust, security, and integrity of the information exchanged becomes an increasingly complex challenge. This paper presents a novel framework designed to address the critical issues of trust and security in data sharing within the Internet of Vehicles. It leverages blockchain technology to create a decentralized and tamper-resistant ledger for the IoV, offering a solution to the inherent vulnerabilities associated with data transmission in a highly dynamic and potentially adversarial environment. Moreover, this model integrates congestion control mechanisms to optimize data sharing, mitigating traffic bottlenecks, and facilitating efficient and reliable communication among vehicles. By introducing this scheme, this paper strives to contribute to the burgeoning field of Internet of Vehicles and, more broadly, to the intersection of blockchain technology, trust, and data sharing in a dynamically evolving transportation landscape. It is our aim to provide a foundation for secure, efficient, and trusted data sharing within the IoV ecosystem, thereby enhancing the safety, convenience, and sustainability of future mobility systems.

1.1 Internet of vehicles

The term "IoV" stands for the "Internet of Vehicles." It is a concept that is part of the broader Internet of Things (IoT) ecosystem. The IoV refers to the interconnection of vehicles, infrastructure, and various sensors through the internet and other communication networks. The primary goal of the IoV is to enable vehicles to communicate with each other, with infrastructure such as traffic lights and road signs, and with centralized systems to share real-time data and information.

Traffic Management: IoV can contribute to more efficient traffic management. For example, traffic data from vehicles can be used to identify congestion, accidents, or roadwork, and this information can be shared with other drivers and traffic management centres to optimize traffic flow.

1.2 Trust management

Trust management in the context of blockchain technology is a critical aspect of ensuring the reliability and security of blockchain networks and applications. Trust management in blockchain encompasses various mechanisms and strategies to establish, assess, and maintain trust in a decentralized and transparent environment.

Smart Contracts: Smart contracts are self-executing contracts with the terms of the agreement directly written into code. Trust in smart contracts is established through code transparency and deterministic execution. Users trust that the smart contract will execute as specified.

1.3 Congestion control

Congestion control in data sharing within transportation systems, particularly in the context of the Internet of Vehicles (IoV), is a crucial consideration. Congestion control aims to manage and mitigate the potential overload of data traffic within the system, ensuring efficient and reliable communication while avoiding network congestion.

1.4 Blockchain

Blockchain technology is an advanced database mechanism that allows transparent information sharing within a business network. A blockchain database stores data in blocks that are linked together in a chain. The data is chronologically consistent because you cannot delete or modify the chain without consensus from the network. As a result, you can use blockchain technology to create an unalterable or immutable ledger for tracking orders, payments, accounts, and other transactions.

CHAPTER 2

LITERATURE SURVEY

2.1 A blockchain-based roadside unit-assisted authentication and key agreement protocol for Internet of Vehicles

Author: Z. Xu, W. Liang, K.-C. Li, J. Xu, H. Jin

Year: 2021

These are infrastructure components deployed along roadways in an IoV ecosystem. RSUs act as communication hubs and are equipped with sensors and processing capabilities. They assist in authenticating vehicles and facilitating secure communication. The decision should be made based on a thorough assessment of the specific needs and goals of the IoV ecosystem and the willingness of stakeholders to embrace this technology.

Merits: Blockchain technology is known for its strong security features. It can enhance the overall security of IoV systems by providing a tamper-resistant and transparent ledger of transactions. Unauthorized access and data breaches are more challenging on a blockchain**.** Blockchain networks demand significant computational resources and energy. Resource-constrained IoV devices may struggle to meet these requirements. Blockchain can simplify and automate key management processes, making them more efficient and reliable.

Demerits: Blockchain systems, particularly public ones like Bitcoin and Ethereum, can suffer from scalability issues. In a rapidly changing environment like IoV, this may lead to delays in transaction processing. Blockchain networks demand significant computational resources and energy. Resource-constrained IoV devices may struggle to meet these requirements. While blockchain ensures data security, it may not fully address privacy concerns, particularly in the context of personal and vehicle-related data.

2.2 Multitype highway mobility analytics for efficient learning model design: A case of station traffic prediction

Author: Duan

Year: 2022

The process of using various types of data and analytics to develop effective predictive models for understanding and forecasting traffic conditions and patterns at specific highway stations. It is a specialized application of data analytics and machine learning in the domain of transportation and traffic management. Successful implementation depends on a careful consideration of these factors and the development of solutions that address these challenges effectively.

Merits: Multitype highway mobility analytics can provide a more accurate representation of traffic patterns. By considering various types of vehicles, such as cars, buses, and trucks, the model can better capture the complex interactions on the road, leading to more precise traffic predictions. Station operators and transportation agencies can use the insights derived from this approach to allocate resources more effectively. Effective traffic prediction through multitype analytics can help authorities and businesses allocate resources, such as traffic management personnel and emergency services, more efficiently.

Demerits: Multitype highway mobility analytics require a diverse and detailed dataset, including information on different vehicle types. Gathering and maintaining such data can be challenging and costly. Analysing multitype mobility data can be computationally intensive. This might require more powerful hardware and longer processing times, which could be a limitation in real-time traffic prediction systems. Developing and maintaining machine learning models capable of handling multitype data can be complex and may require ongoing adjustments and updates. This complexity can increase the need for specialized expertise.

2.3 A blockchain-based solution for reputation management in IoV

Author: S. Abbes, S. Rekhis

Year: 2021

A blockchain-based solution for reputation management in the IoV aims to foster trust, reliability, and accountability among participants by providing a transparent and immutable record of their behaviour and interactions. This, in turn, enhances the safety and efficiency of IoV operations, such as autonomous driving, traffic management, and transportation services. The decision should be made based on a thorough assessment of the specific needs and goals of the IoV ecosystem and the willingness of stakeholders to embrace this technology.

Merits: Blockchain technology ensures the security and immutability of reputation data. Once reputation information is recorded on the blockchain, it is nearly impossible to alter or tamper with, enhancing the trustworthiness of the system. The transparent nature of the blockchain allows all participants in the IoV ecosystem to verify the reputation of others. Vehicles and other entities in the IoV can independently verify the reputation of other participants, making informed decisions regarding interactions and transactions.

Demerits: Blockchain networks can face scalability issues, and as IoV data can be extensive, this can lead to slow transaction processing and higher costs. Recording reputation data on a public blockchain can raise privacy concerns, as participants might be uncomfortable with having their interactions and behaviors openly accessible. Convincing vehicle manufacturers, infrastructure providers, and regulatory bodies to adopt blockchain technology for reputation management may be challenging due to inertia and resistance to change in the transportation sector. While blockchain ensures data security, privacy concerns may persist, particularly as reputation data can expose sensitive information about individuals and vehicles.

2.4 LVBS: Lightweight vehicular blockchain for secure data sharing in disaster rescue

Author: Z. Su, Y. Wang, Q. Xu, N. Zhang

Year: 2022

It refers to a blockchain-based system designed to provide a secure and efficient way to share data in the context of disaster rescue operations. The system is intended for use in vehicles, such as emergency response vehicles, to enable real-time data sharing and coordination among different entities involved in disaster response. The term "lightweight" suggests that the blockchain is optimized for performance and resource efficiency, which is crucial in fast-paced and resource-constrained disaster scenarios. This technology can enhance communication and data sharing among rescue teams, first responders, and other relevant parties, ultimately improving the effectiveness of disaster relief efforts.

Merits: LVBS provides a secure platform for data sharing, which is crucial in disaster rescue operations. Data stored on the blockchain is inherently secure and tamper-resistant, reducing the risk of unauthorized access or data manipulation. Blockchain technology is decentralized, meaning there is no central authority or single point of failure. This can improve the reliability of communication and data sharing, even in chaotic environments. The blockchain records are immutable, which means once data is added, it cannot be altered.

Demerits: Implementing and maintaining a blockchain system, even a lightweight one, can be complex and may require specialized technical knowledge. This could be a challenge for some disaster response teams. Traditional blockchains may face scalability issues when a large number of transactions are processed simultaneously. This could affect the performance of LVBS in highly congested disaster areas. Blockchain systems typically require an internet connection to function, which may not always be reliable or available in disaster-stricken areas. This dependency can be a limitation.

2.5 Blockchain based trustworthy energy dispatching approach for high renewable energy penetrated power systems

Author: Y. Xu, Z. Liu, C. Zhang, J. Ren, Y. Zhang, X. Shen

Year: 2022

A system and methodology that utilizes blockchain technology to enhance the dispatching and management of energy in power systems with a substantial integration of renewable energy sources. This approach is designed to address the unique challenges and requirements associated with a grid that relies heavily on renewable energy generation, such as wind and solar power. It addresses the challenges associated with renewable energy variability and seeks to ensure the reliable and trustworthy operation of the power grid.

Merits: Blockchain provides an immutable and transparent ledger of energy transactions, making it easy to trace the source and flow of energy. This transparency can enhance accountability and trust among stakeholders. The decentralized and cryptographic nature of blockchain ensures high levels of security, reducing the risk of data tampering, fraud, and cyberattacks. It can be especially important when managing critical infrastructure like power systems. Automation through smart contracts can minimize human errors in energy dispatching, leading to more accurate and reliable operations.

Demerits: Implementing a blockchain-based system can be complex and may require a high level of technical expertise. This complexity can be a barrier to adoption and may lead to higher development costs. As the number of transactions on the blockchain increases, scalability can become a challenge. High transaction volumes could lead to delays and increased computational demands. While blockchain offers security, it can also pose data privacy concerns. It's important to manage sensitive energy data while maintaining transparency.

2.6 Blockchain based decentralized trust management in vehicular networks

Author: Z. Yang, K. Yang, L. Lei, K. Zheng, V. C. M. Leung

Year: 2019

Blockchain-based decentralized trust management in vehicular networks refers to a system and methodology that utilizes blockchain technology to establish and maintain trust among vehicles and entities participating in vehicular communication and cooperation. In this context, vehicular networks involve vehicles communicating with each other and with infrastructure components for various purposes, such as improving road safety, traffic management, and transportation efficiency. It enhances security, transparency, and trustworthiness in vehicular communication, which is vital for ensuring the safety and efficiency of future transportation systems.

Merits: Blockchain technology provides a high level of security through cryptographic methods and decentralized consensus mechanisms. This reduces the risk of unauthorized access, fraud, and cyberattacks in vehicular networks. The transparent nature of blockchain ensures that all participants can view the data and trust-related transactions, promoting accountability and reducing the likelihood of disputes. The absence of a central authority in trust management can reduce the risk of single points of failure and enhance the overall reliability of vehicular networks.

Demerits: As vehicular networks grow and generate a high volume of transactions, the scalability of the blockchain can become a challenge, leading to potential delays and increased computational demands. The blockchain's consensus mechanisms, such as proof-of-work or proof-of-stake, can introduce latency into trust verification processes, which may not be suitable for applications requiring real-time decision-making, like autonomous vehicles. Designing and implementing a blockchain-based trust management system in vehicular networks can be complex and require specialized technical expertise.

CHAPTER 3

PROBLEM STATEMENT

- In Internet of Vehicles, the data shared by the vehicles are not always trustable due to presence of attackers.
- In existing literatures, centralized trust management systems are proposed. They have issues with single point of failure and being tampered.
- Communication overhead occurs between nodes during trust evaluation process.

3.1 OBJECTIVES

- To ensure the traffic data security and accident avoidance.
- To introduce decentralized trust management system in vehicular network using blockchain.
- To reduce congestion in data transfer using Kademlia algorithm and cuckoo filter.

3.2 SCOPE

- Internet of Vehicles
- Improving Traffic efficiency
- Accidents avoidance
- Traffic data security

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1. SYSTEM REQUIREMENTS

The software requirements specification is acquired at the completion of analysis phase. The function and performance allocated to software in system engineering is elaborated by complete information description of software, performance evaluation, design constraints and appropriate validation criteria.

4.1.1 Functional requirements

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process. It lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specifications is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

Hardware Requirements

Hard Disk: 320 GB and above

RAM: 8 GB and Above

Processor: Intel(R) Core (TM) i5-7020U CPU

Software Requirements

IDE: Visual Studio and Jupiter

Operating System: Windows 11Coding

Language: Python, HTML, CSS

4.1.2 Non-functional requirements

Describe user-visible aspects of the system that are not directly related with the functional behaviour of the system. Non-functional requirements include quantitative constraints, such as response time (i.e., how fat the system reacts to user commands) or accuracy (i.e., how precise are the systems numerical answers).

Scalability

- Program with larger number of lines can be obfuscated.

Supportability

- The developer can use any platform for converting the assembly code to executable.

Performance Characteristics

- Speed, throughput and execution time depends on the number of instructions obfuscated.

Security

- Provides security of the code from any unknown intruders.

Flexibility

- The system shows great flexibility in compile the visual studio code.

CHAPTER 5

DESIGNS

5.1 System architecture

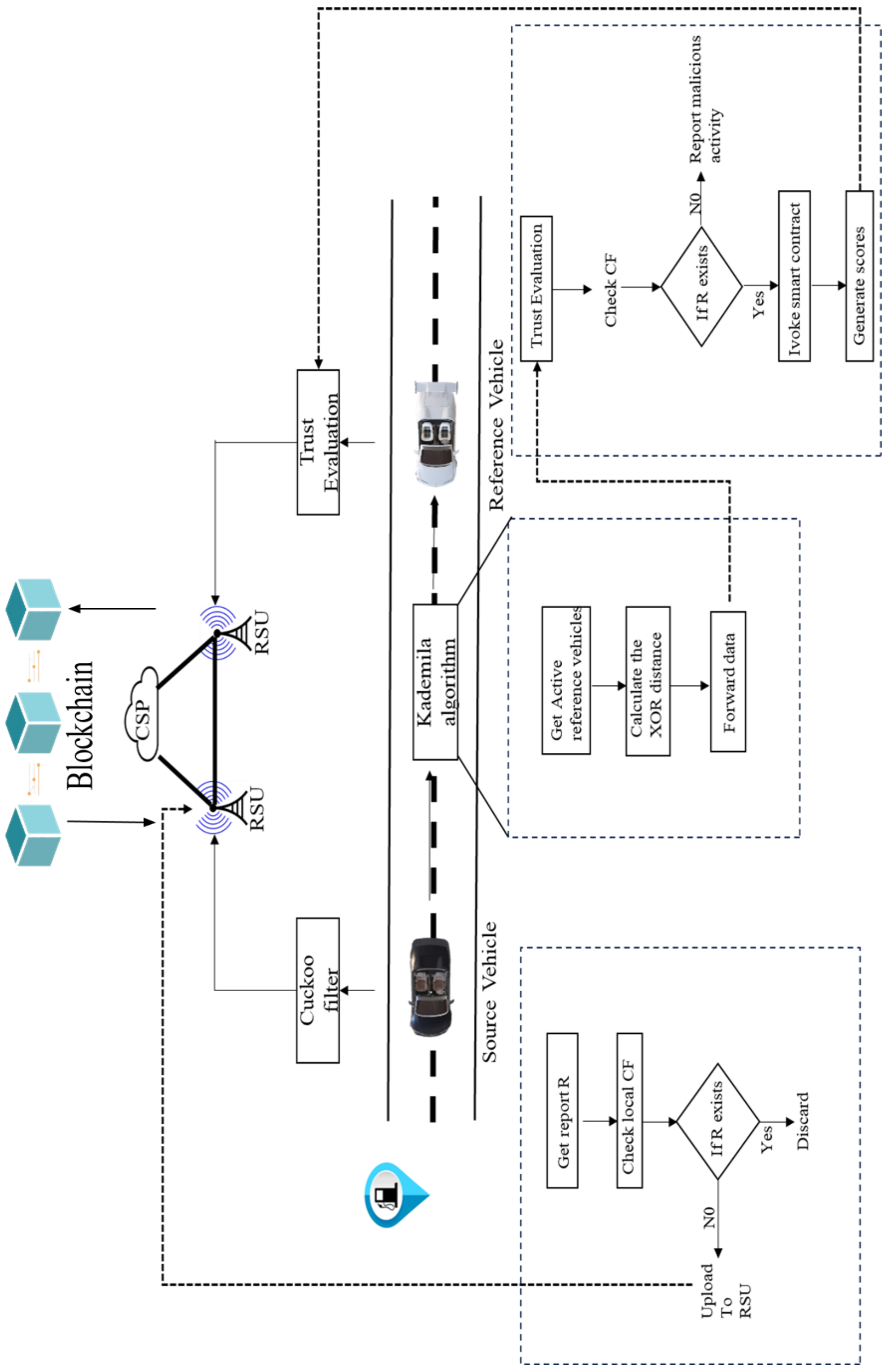


Fig. 5.1 System Architecture

5.2 MODULE DESCRIPTION

The system architecture consists of the three main modules which are given below.

- Congestion Control
- Data Sharing Phase
- Trust Evaluation Phase

5.2.1 Congestion Control

Congestion control in the system involves the use of algorithms like Kademlia and cuckoo filters to manage data flow, prevent network congestion, and reduce communication overhead in the Internet of Vehicles. These mechanisms contribute to efficient data sharing, improved network performance, and enhanced trustworthiness of shared data in the IoV ecosystem.

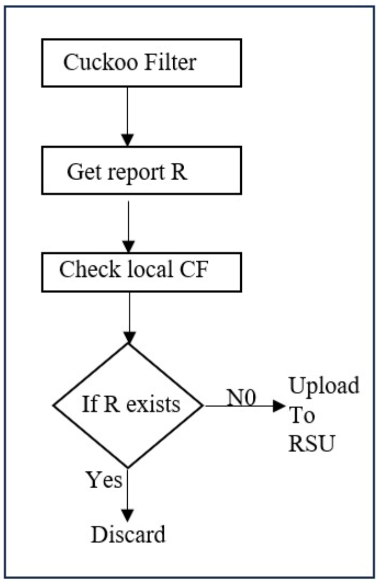


Fig. 5.2 Congestion control

The adoption of cuckoo filters on vehicles and RSUs enables the detection and elimination of duplicate data reports. By checking local filters, vehicles and RSUs can verify if the current data report has already been shared, thereby avoiding redundant data transmission and reducing congestion on the network.

Deduplicated traffic reports uploading (Cuckoo Filter)

1. Iterate over each traffic report r in the set Ri.
2. Generate a fingerprint f for the traffic report.
3. Compute two hash values i1 and i2 based on the traffic report.
4. Check if the local filter CFk contains the traffic report r.
    - If it does, update the repeat times in CFk and continue.
    - If it doesn't, add the fingerprint and a counter to CFk and send a request to a Road Side Unit (RSU).
5. RSU decrypts the request to get the traffic report.
6. If the cuckoo filter CFk contains the fingerprint f, update the corresponding entry.
7. If the buckets associated with the hash values have empty entries, insert the fingerprint.
8. If none of the above conditions are met, return failure.
9. Repeat this process for each traffic report in the set Ri.
10. Return the final cuckoo filter CFk.

1. Algorithm (Data Deduplication using Cuckoo Filter)

```
Input: Traffic report set $R_i$, Trust values $T_i$
Output: Cuckoo filter $CF_k$
1: for $r \in R_i$, $i \in \{1, 2\ldots, n\}$ do
2: $f$=fingerprint($r$)
3: $i_1$=hash($r$), $i_2 = i_1 \oplus$hash($f$)
4: if local filter $CF'_k$ contains $r$ then
5: update repeat times $t_r$ in $CF'_k$
6: continue
7: else
```

```
8: Add <$H(r)$ || 1> to $CF'_k$
9: Send $Req_i=E_{pk_{RSU_j}}(r||VIN_i)$ to $RSU_j$
10: **end if**
11: $RSU_j$ decrypts $Req_i$ to get r
12: **if** $CF_k$ contains f **then**
13: update <$H(r)$ || $t_r$> to <$H(r)$||$t_r$+1>
14: **else if** bucket[$i_1$] or bucket[$i_2$] has empty entry **then**
15: select a entry to insert <$H(r)$ || 1>
16: **else**
17: **return** Failure
18: **end if**
19: **end for**
20: **return** $CF_k$
```

5.2.2 Data Sharing Phase

Our proposed scheme relies on the close cooperation between vehicles, RSUs, and CSP. Vehicles are equipped with multiple sensors, such as cameras, millimetre wave radar and lidar, etc.

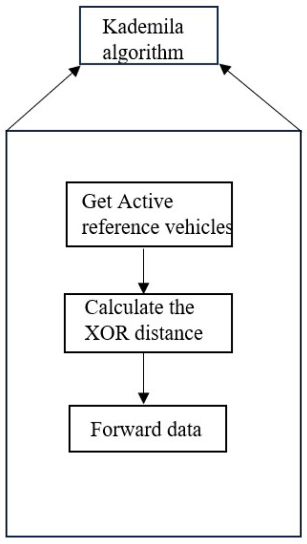


Fig. 5.3 Data Forwarding using Kademlia

Their perceived original data are then processed by the edge computing equipment using fusion methods mentioned. After performing a standardization process, the fused data can become a traffic event report. The standardized traffic data reports will be uploaded to RSUs so as to make better decisions to optimize the global traffic situation. However, chances are that vehicles in area-k generate the same reports, which brings unnecessary communication overhead. To avoid this kind of situation, RSUs in area-k will create a cuckoo filter to filter out the duplicate data. The data sharing phase in the system involves the collection, processing, and sharing of traffic data among vehicles, RSUs, and the CSP.

Traffic Reports Forwarding (Kademlia Algorithm)

1. Iterate over each vehicle vj in a set {1, 2, ..., m}.
2. Calculate the distance between the current vehicle vj and a reference vehicle vi denoted by d (i, j) using an XOR operation on their coordinates.
3. If the calculated distance is greater than or equal to a threshold d, discard the current vehicle vj from the traffic report set RTi associated with the reference vehicle vi, and continue to the next vehicle.
4. Find an index x that satisfies a specific range condition based on the distance d (i, j).
5. If the k-bucket at index x is not full, insert the current vehicle vj into it.
6. If the k-bucket at index x is full, remove an invalid node from it and then insert the current vehicle vj.
7. Transmit the traffic report set Ri from the reference vehicle vi to the current vehicle vj in the set RTi.
8. Repeat this process for all vehicles in the set.

2. Algorithm (Data Forwarding)

Input: Vehicle coordinate λj

Output: Traffic report set Ri

```
1: for vj, j ∈ {1, 2…, m} do
2: d (i, j) =$\lambda i \oplus \lambda j$
3: if d (i, j) ≥d then
```

```
4: vi discard vj in RTi and **continue**
5: **end if**
6: find index x satisfying d (i, j) ∈[2x,2x+1)
7: **if** k -bucket[x] is not full **then**
8: insert vj to k -bucket[x]
9: **else**
10: remove invalid node in k -bucket[x]
11: insert vj to k -bucket[x]
12: **end if**
13: vi transmits Ri to vj in RTi
14: **end for**
15: **return** Ri
```

5.2.3 Trust Evaluation Phase

Since the vehicles in our scheme are not trustworthy, they may transmit fake messages to RSUs due to hardware damages or malicious purposes.

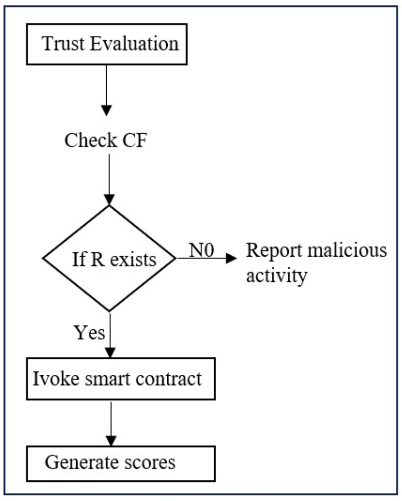


Fig. 5.4 Trust Evaluation

To make vehicles behave honestly and reduce communication overheads, we propose a unique trust management mechanism. RSUs will initiate the trust evaluation process occasionally. Vehicles are divided into source vehicles and reference vehicles.

Source vehicle needs to transmit report set uploaded in the last time period to other vehicles. At first, source vehicle needs to decide which reference vehicles to forward these reports to. Secondly, reference vehicle gives a score to source vehicle and invoke smart contracts to update trust value. Specifically, source vehicle completes the data forwarding phase using Kademlia algorithm. Source vehicle sends a request to the nearest RSU to fetch the active reference vehicles list information. The list contains ID and their coordinate. The XOR distance is calculated and the data is shared to the reference vehicles.

Trust Evaluation

1. Iterate over each vehicle vj in a set {1, 2, ..., m}.
2. Calculate the intersection of the report sets Ri and Rj.
3. For each report r in the intersection of Ri and Rj:
    - ➢ If the hash of the report r is not contained in the cuckoo filter CFk, update the trust value T'i of vehicle vi based on some formula.
    - ➢ Otherwise, if vehicle vj thinks the report r is correct, update a counter N(c) representing correct reports; otherwise, update $N(\bar{c})$ representing incorrect reports.
4. Calculate the score Sj→i representing the trust score from vj to vi.
5. Get the average score Si based on scores from all vehicles vj.
6. If the average score Si is greater than or equal to the original trust value Ti, update the trust value T'i of vi with a positive adjustment factor ρ.
7. If Si is less than Ti, update T'i with a negative adjustment factor σ.
8. Return the updated trust value T'i.

3. Algorithm (Trust Evaluation)

Input: Report set Ri and Rj, Original trust value Ti

Output: Trust value T′i

```
1: for vj, j ∈ {1,2…m} do
2: Φ (Ri, Rj) =Ri∩Rj
3: for report r ∈Φ (Ri, Rj) do
```

```
4: **if** $H(r)$ is not contained in $CF_k$ **then**
5: update trust value $T'_i=T_i\cdot(1-\varepsilon)\ \mu$
6: **else if** $v_j$ thinks $r$ is correct **then**
7: update $N(c)=N(c)+1$
8: **else**
9: update $N(\bar{c}) = N(\bar{c}) + 1$
10: **end if**
11: **end for**
12: calculate $S_{j\to i}=N(c)/(N(c)+N(\bar{c}))$
13: **end for**
14: get average score $S_i=\sum_{j=1}^{m} T_j\cdot S_{j\to i}/m$
15: **if** $S_i\geq T_i$ **then**
16: update trust value $T'_i=T_i+(S_i-T_i)\ \rho$
17: **else**
18: update trust value $T'_i=T_i-(T_i-S_i)\ \sigma$
19: **end if**
20: **return** $T'_i$
```

Roadside Unit (RSU)

A Roadside Unit (RSU) is a device used in Intelligent Transportation System and connected vehicle environments. It is typically installed alongside roads or highways to facilitate communication between vehicles and the transportation infrastructure.

Cloud Service Provider (CSP)

A CSP (cloud service provider) is a third-party company that provides scalable computing resources that businesses can access on demand over a network, including cloud-based compute, storage, platform, and application services.

5.3 UML DIAGRAMS

5.3.1 Use case Diagram

A use case diagram is a graphical representation of the interactions between a system and its users or actors. It depicts the functionality of a system from the user's perspective by showing the different use cases or tasks that the system can perform.

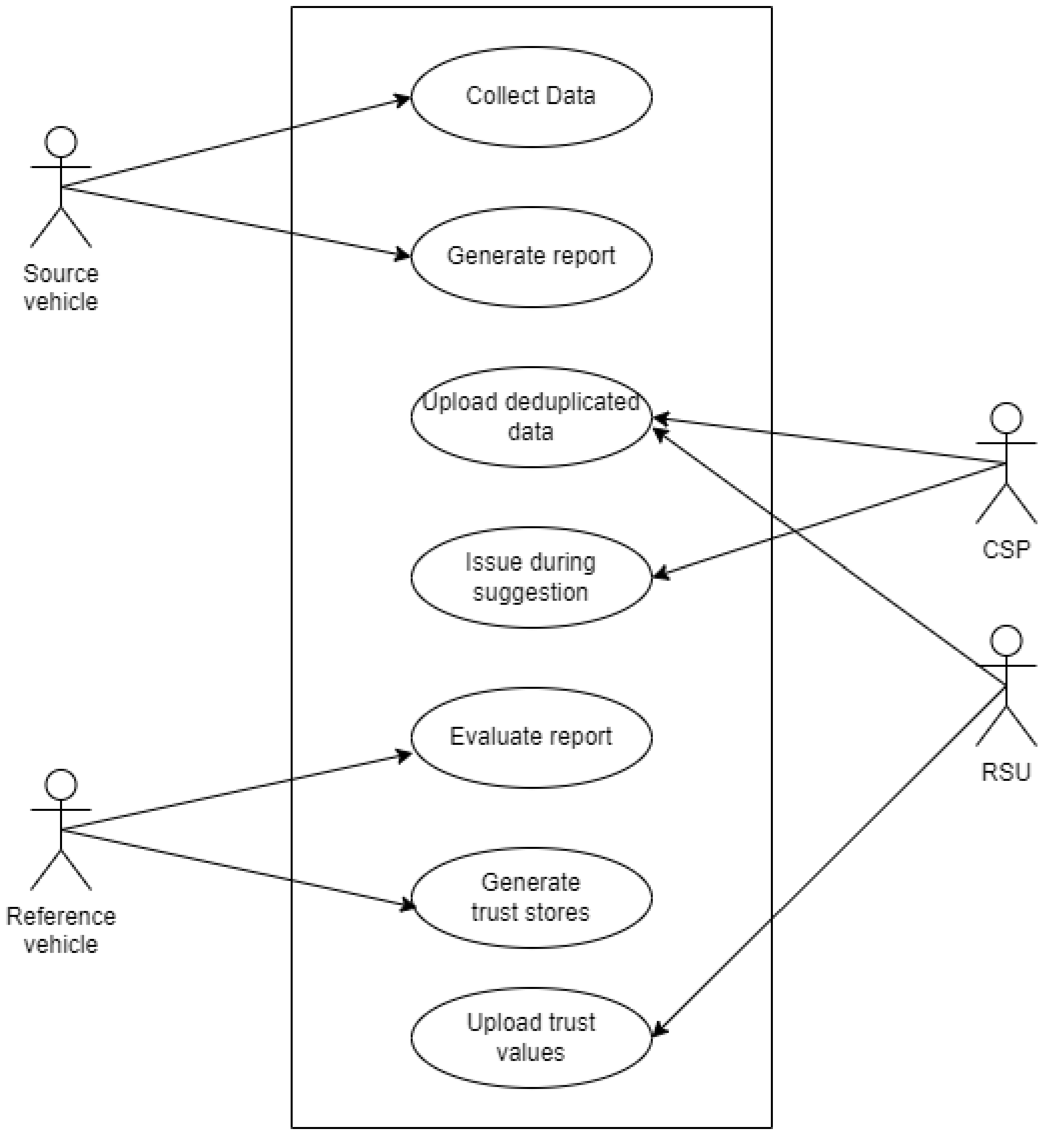


Fig. 5.5 Use Case Diagram

5.3.2 Class Diagram

Class diagrams depict the structure of a system by showing the classes, their attributes, and the relationships between them. They can help developers to understand the relationships between different objects in the system and how data flows between them.

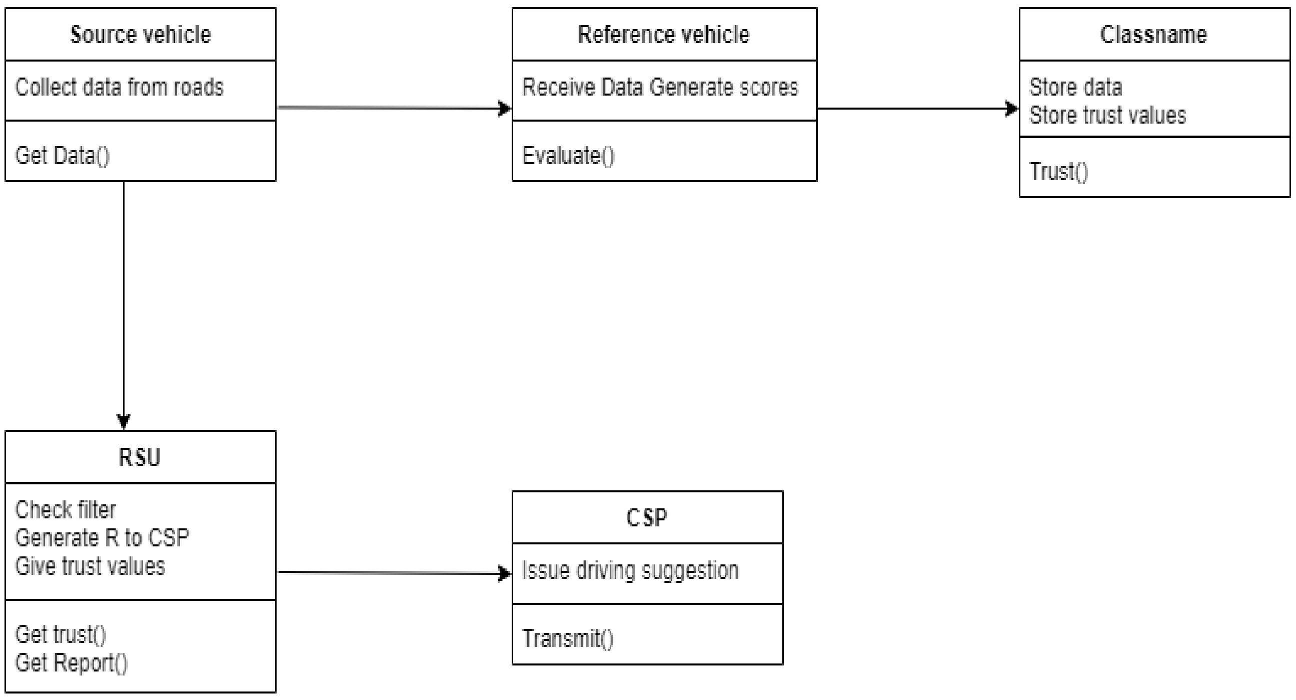


Fig. 5.6 Class Diagram

5.3.3 Sequence Diagram

Sequence diagrams depict the interactions between objects in a system over time, showing the order in which messages are sent and received. They can help developers to understand the behaviour of the system during different scenarios and identify potential issues.

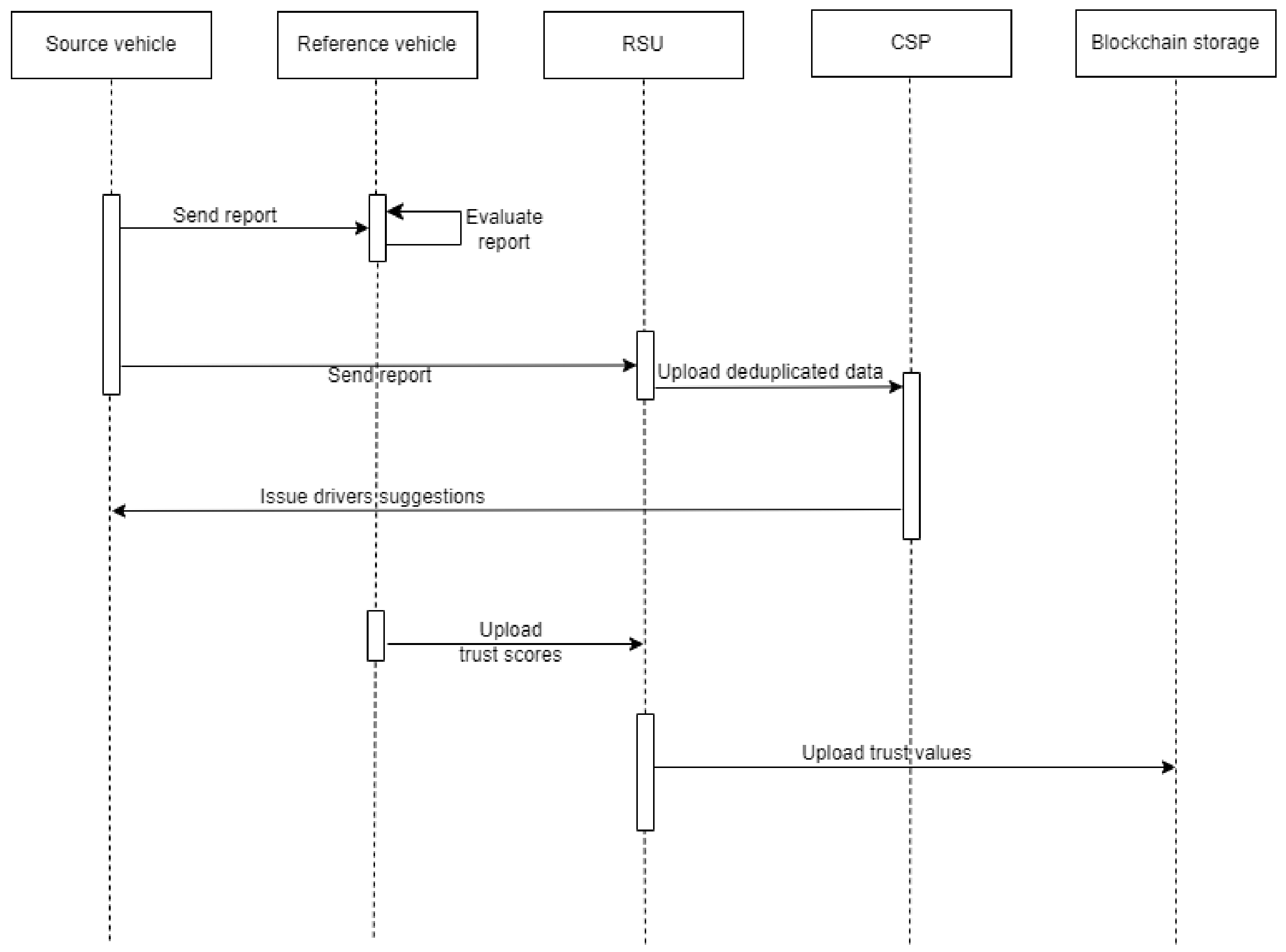


Fig. 5.7 Sequence Diagram

5.3.4 Activity Diagram

Activity diagrams depict the flow of control in a system by showing the activities and decisions involved in a process. They can help developers to understand the various steps involved in a process and identify potential bottlenecks or inefficiencies.

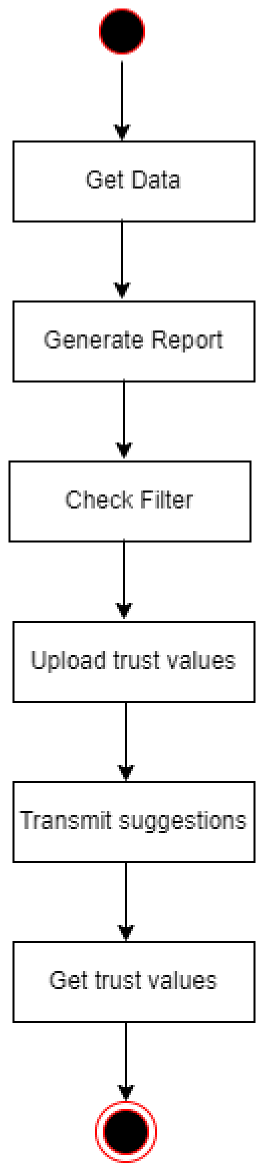


Fig. 5.8 Activity Diagram

5.3.5 State-chart Diagram

State diagrams depict the states and transitions of objects in a system, showing how objects behave in response to events. They can help developers to understand the behaviour of the system during different states and identify potential issues.

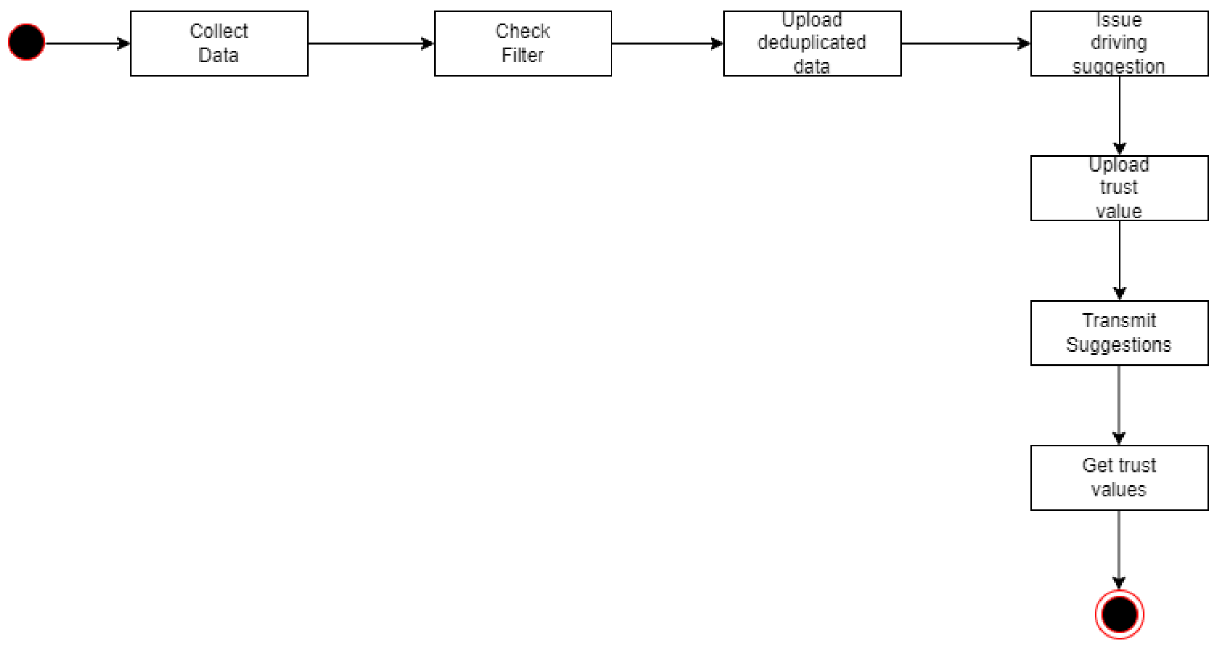


Fig. 5.9 State-chart Diagram

5.3.6 Component Diagram

Component diagrams depict the physical and logical components of a system and the relationships between them. They can help developers to understand how the various components of the system work together and how changes to one component.

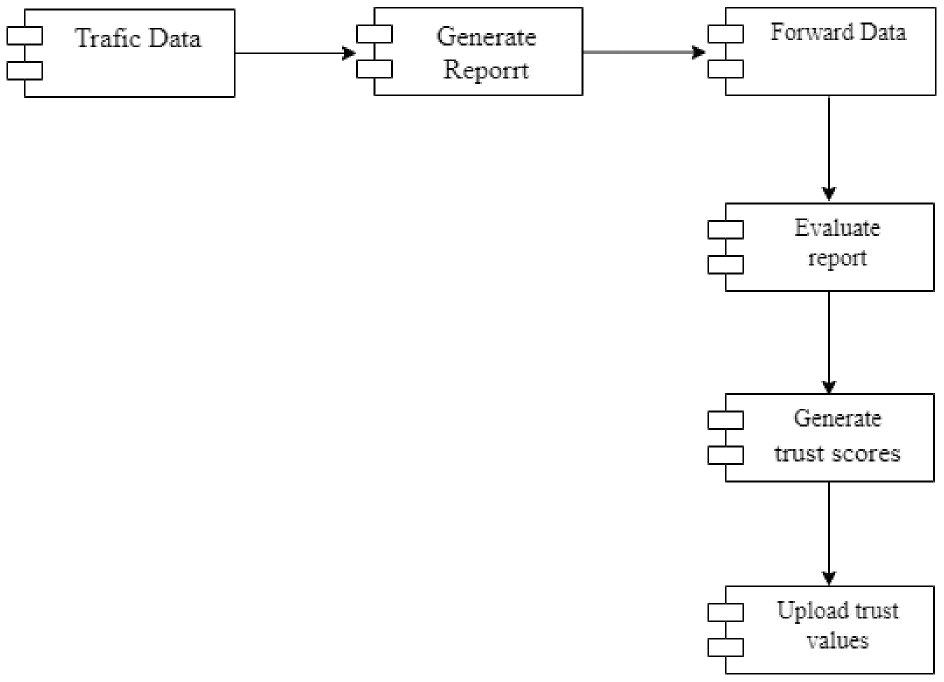


Fig. 5.10 Component Diagram

5.3.8 Deployment Diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modelling the physical aspects of an object-oriented system. They are often used to model the static deployment view of a system.

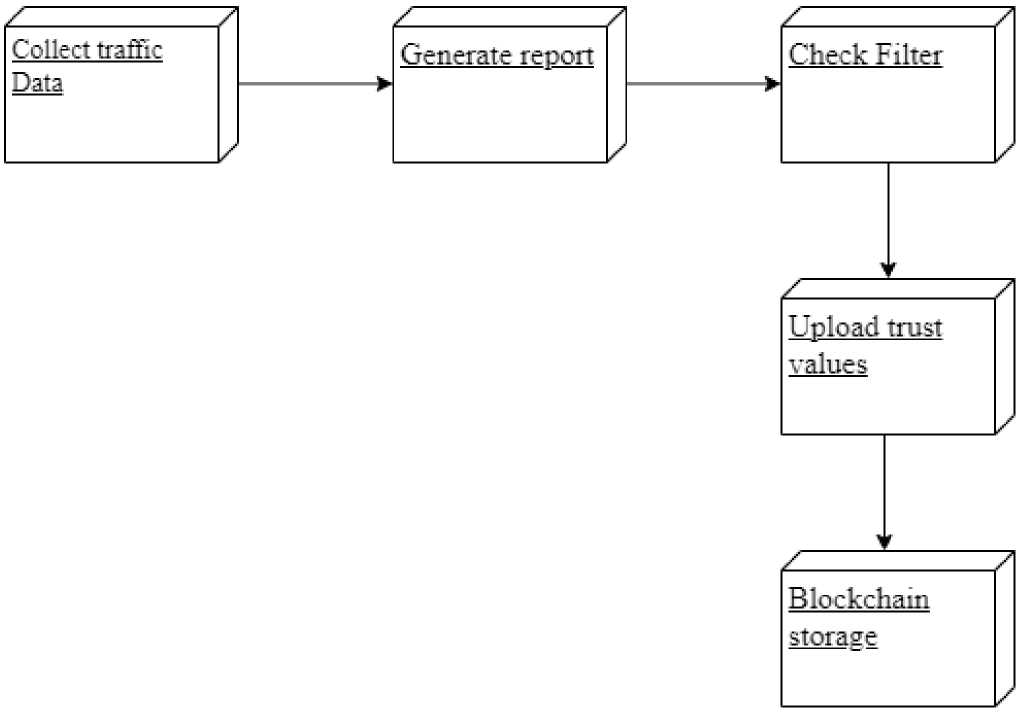


Fig. 5.11 Deployment Diagram

5.4 Data Flow Diagram

A data flow diagram is a graphical representation of the “flow “of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts.

5.4.1 DFD Level 0

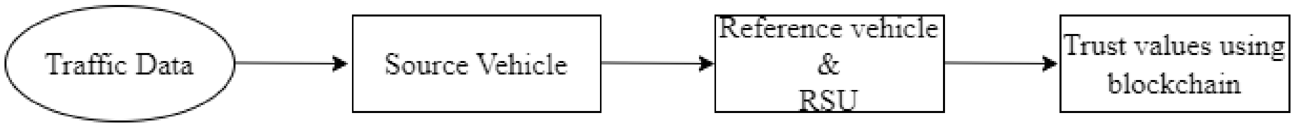


Fig. 5.12 DFD Level 0

5.4.2 DFD Level 2

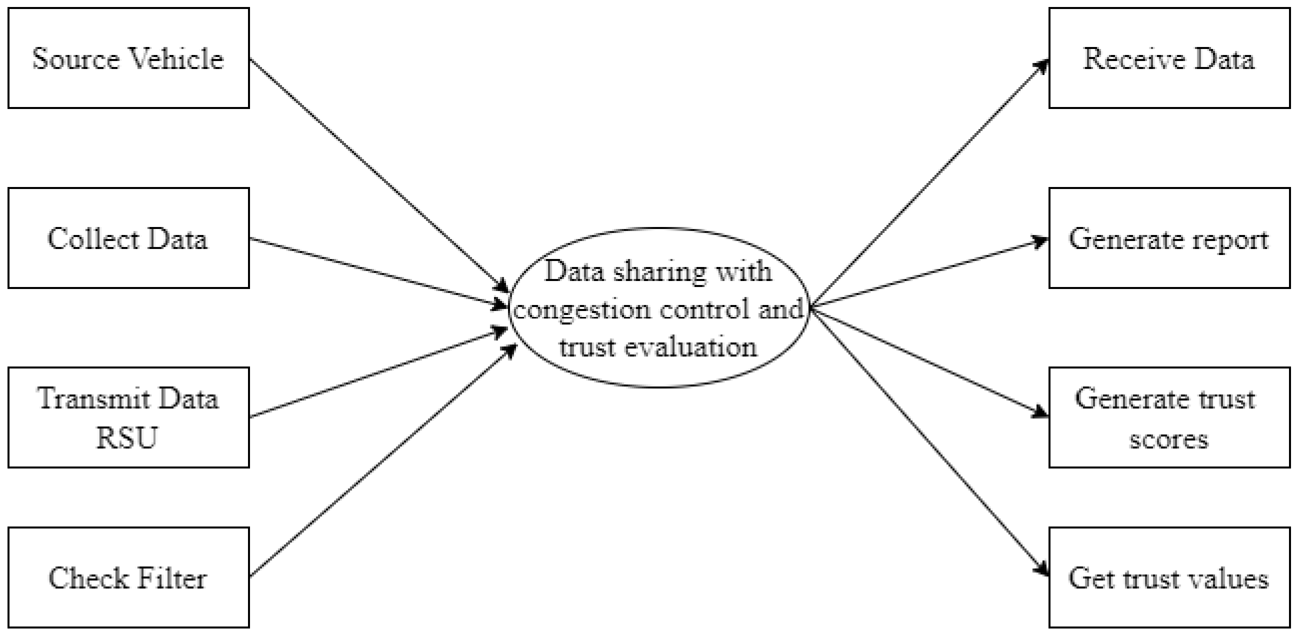


Fig. 5.13 DFD Level 2

5.4.3 DFD Level 3

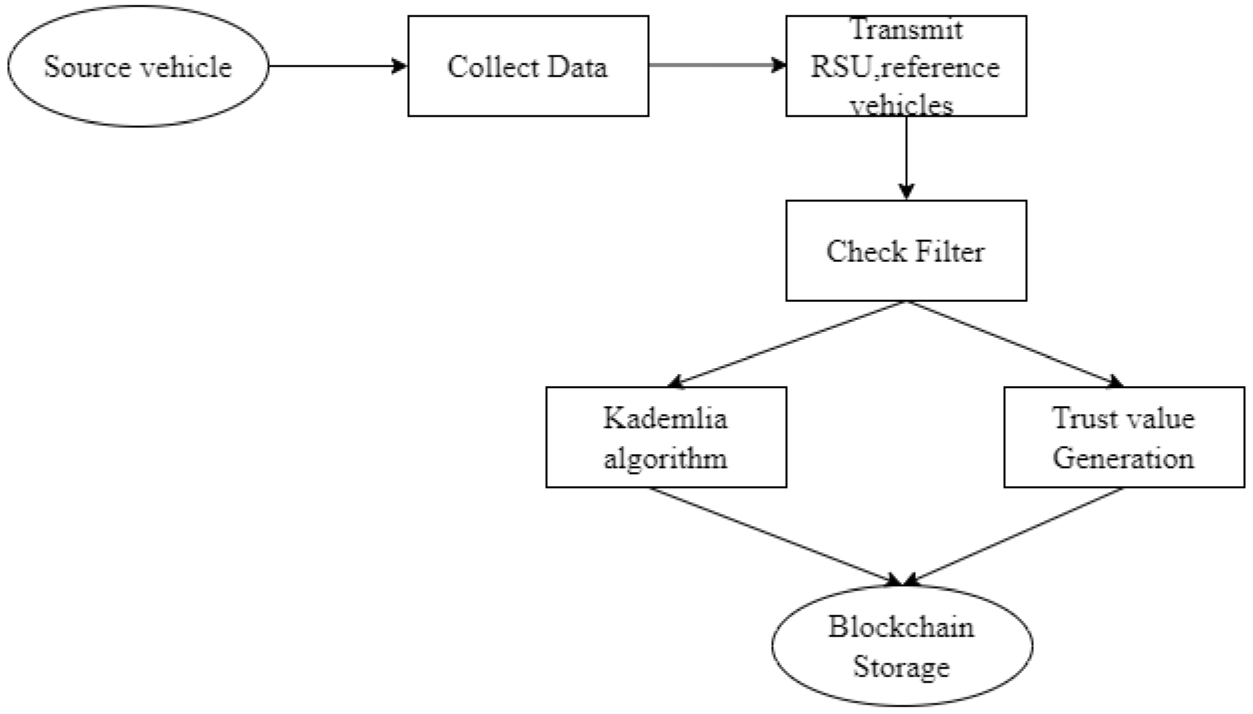


Fig. 5.14 DFD Level 3

CHAPTER 6

IMPLEMENTATION

6.1 Technologies used

The project can be done successfully by using various technologies for different functions or operations.

6.1.1 Python

Python is a popular high-level, interpreted programming language known for its simplicity, readability, and versatility. The codes are all executed in the python programming language.

6.1.2 Blockchain

Blockchain is a decentralized digital ledger technology that allows multiple parties to maintain a shared and immutable record of transactions or data in a secure and transparent manner. It is the underlying technology behind cryptocurrencies like Bitcoin, but its applications go beyond just digital currencies. At its core, a blockchain consists of a chain of blocks, where each block contains a list of transactions or data.

6.1.3 Private Ethereum Blockchain

Ethereum is a network of computers all over the world that follow a set of rules called the Ethereum protocol. The Ethereum network acts as the foundation for communities, applications, organizations and digital assets that anyone can build and use. You can create an Ethereum account from anywhere, at any time, and explore a world of apps or build your own. The core innovation is that you can do all this without trusting a central authority that could change the rules or restrict your access.

6.1.4 Kademlia

Kademlia uses a "distance" calculation between two nodes. This distance is computed as the exclusive or (XOR) of the two node IDs, taking the result as an unsigned integer number. Keys and node IDs have the same format and length, so distance can be calculated among them in exactly the same way. The node ID is typically a large random number that is chosen with the goal of being unique for a particular node (see UUID).

6.1.5 Cuckoo Filter

A cuckoo filter is a probabilistic data structure that is used to test whether an element is a member of a set or not. It is similar to Bloom filter. It can show false positives but not false negatives. It can also delete existing items. It is a hash table that uses cuckoo hashing to resolve collisions. It decreases the space used by only maintaining a fingerprint of the value to be stored in the set. It uses a small f-bit fingerprint to store the data. The value of f is decided by the ideal false positive rate.

6.2 Snapshots

Geth installation

Geth – Go Ethereum Blockchain

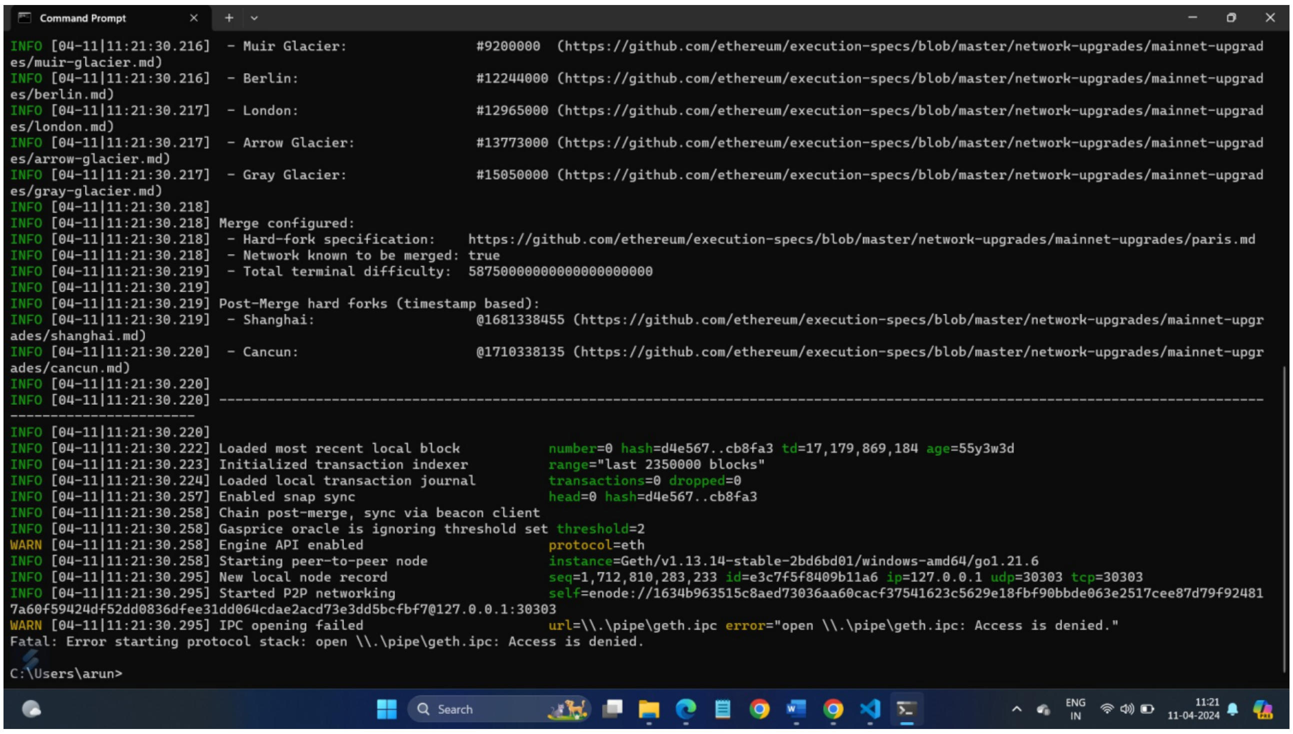
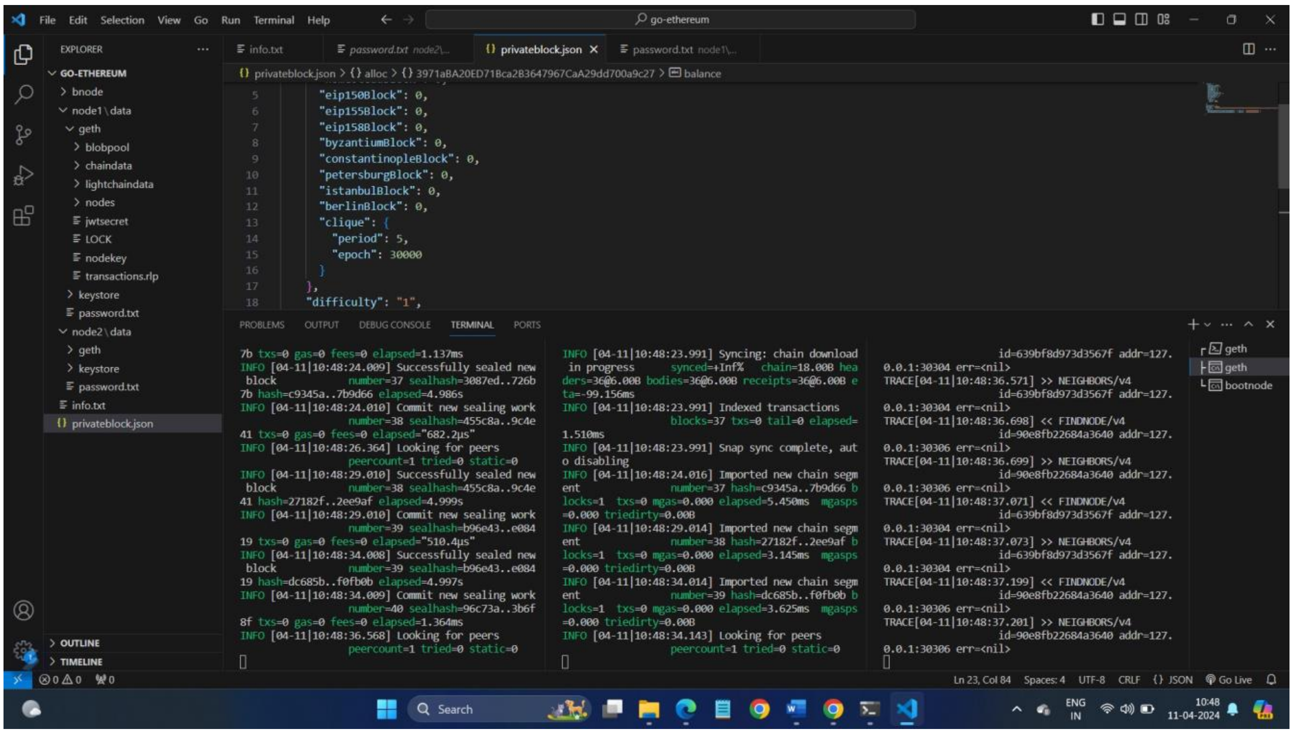

```
INFO [04-11|11:21:30.216]  - Muir Glacier:                #9200000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrad
es/muir-glacier.md)
INFO [04-11|11:21:30.216]  - Berlin:                      #12244000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrad
es/berlin.md)
INFO [04-11|11:21:30.217]  - London:                      #12965000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrad
es/london.md)
INFO [04-11|11:21:30.217]  - Arrow Glacier:               #13773000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrad
es/arrow-glacier.md)
INFO [04-11|11:21:30.217]  - Gray Glacier:                #15050000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrad
es/gray-glacier.md)
INFO [04-11|11:21:30.218]
INFO [04-11|11:21:30.218] Merge configured:
INFO [04-11|11:21:30.218]  - Hard-fork specification:    https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris.md
INFO [04-11|11:21:30.218]  - Network known to be merged: true
INFO [04-11|11:21:30.219]  - Total terminal difficulty:  58750000000000000000000
INFO [04-11|11:21:30.219]
INFO [04-11|11:21:30.219] Post-Merge hard forks (timestamp based):
INFO [04-11|11:21:30.219]  - Shanghai:                    @1681338455 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgr
ades/shanghai.md)
INFO [04-11|11:21:30.220]  - Cancun:                      @1710338135 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgr
ades/cancun.md)
INFO [04-11|11:21:30.220]
INFO [04-11|11:21:30.220] ---------------------------------------------------------------------------------------------------------------------------
----------------------
INFO [04-11|11:21:30.220]
INFO [04-11|11:21:30.222] Loaded most recent local block           number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=55y3w3d
INFO [04-11|11:21:30.223] Initialized transaction indexer          range="last 2350000 blocks"
INFO [04-11|11:21:30.224] Loaded local transaction journal         transactions=0 dropped=0
INFO [04-11|11:21:30.257] Enabled snap sync                        head=0 hash=d4e567..cb8fa3
INFO [04-11|11:21:30.258] Chain post-merge, sync via beacon client
INFO [04-11|11:21:30.258] Gasprice oracle is ignoring threshold set threshold=2
WARN [04-11|11:21:30.258] Engine API enabled                       protocol=eth
INFO [04-11|11:21:30.258] Starting peer-to-peer node               instance=Geth/v1.13.14-stable-2bd6bd01/windows-amd64/go1.21.6
INFO [04-11|11:21:30.295] New local node record                    seq=1,712,810,283,233 id=e3c7f5f8409b11a6 ip=127.0.0.1 udp=30303 tcp=30303
INFO [04-11|11:21:30.295] Started P2P networking                   self=enode://1634b963515c8aed73036aa60cacf37541623c5629e18fbf90bbde063e2517cee87d79f92481
7a60f59424df52dd0836dfee31dd064cdae2acd73e3dd5bcfbf7@127.0.0.1:30303
WARN [04-11|11:21:30.295] IPC opening failed                       url=\\.\pipe\geth.ipc error="open \\.\pipe\geth.ipc: Access is denied."
Fatal: Error starting protocol stack: open \\.\pipe\geth.ipc: Access is denied.

C:\Users\arun>
```

Fig. 6.1 Geth – Go Ethereum Blockchain

Private Ethereum Blockchain setup



privateblock.json > alloc > 3971aBA20ED71Bca2B3647967CaA29dd700a9c27 > balance

```
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "berlinBlock": 0,
    "clique": {
      "period": 5,
      "epoch": 30000
    }
  },
  "difficulty": "1",
```

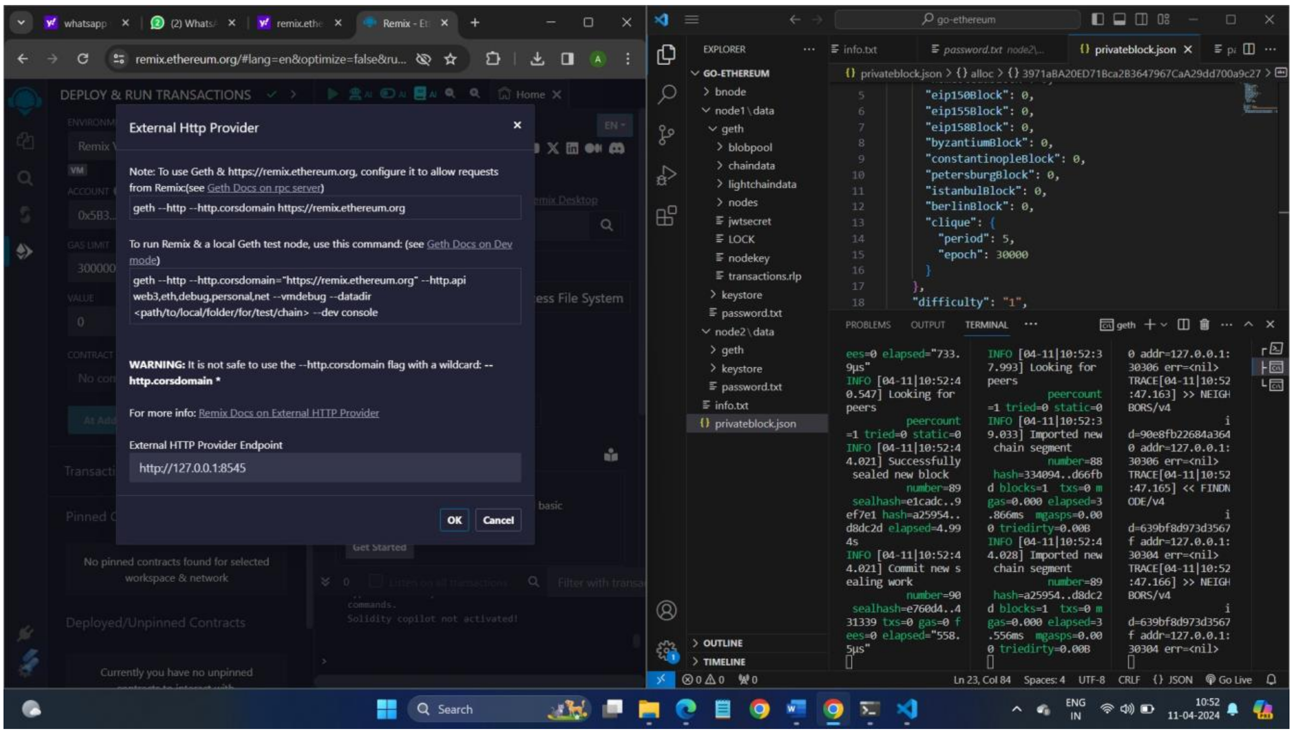
```
7b txs=0 gas=0 fees=0 elapsed=1.137ms
INFO [04-11|10:48:24.009] Successfully sealed new
 block                 number=37 sealhash=3087ed..726b
7b hash=c9345a..7b9d66 elapsed=4.986s
INFO [04-11|10:48:24.010] Commit new sealing work
                       number=38 sealhash=455c8a..9c4e
41 txs=0 gas=0 fees=0 elapsed="682.2µs"
INFO [04-11|10:48:26.364] Looking for peers
                       peercount=1 tried=0 static=0
INFO [04-11|10:48:29.010] Successfully sealed new
 block                 number=38 sealhash=455c8a..9c4e
41 hash=27182f..2ee9af elapsed=4.999s
INFO [04-11|10:48:29.010] Commit new sealing work
                       number=39 sealhash=b96e43..e084
19 txs=0 gas=0 fees=0 elapsed="510.4µs"
INFO [04-11|10:48:34.008] Successfully sealed new
 block                 number=39 sealhash=b96e43..e084
19 hash=dc685b..f0fb0b elapsed=4.997s
INFO [04-11|10:48:34.009] Commit new sealing work
                       number=40 sealhash=96c73a..3b6f
8f txs=0 gas=0 fees=0 elapsed=1.364ms
INFO [04-11|10:48:36.568] Looking for peers
                       peercount=1 tried=0 static=0
```

```
INFO [04-11|10:48:23.991] Syncing: chain download
 in progress       synced=+Inf%   chain=18.00B hea
ders=36@6.00B bodies=36@6.00B receipts=36@6.00B e
ta=-99.156ms
INFO [04-11|10:48:23.991] Indexed transactions
                       blocks=37 txs=0 tail=0 elapsed=
1.510ms
INFO [04-11|10:48:23.991] Snap sync complete, aut
o disabling
INFO [04-11|10:48:24.016] Imported new chain segm
ent                    number=37 hash=c9345a..7b9d66 b
locks=1  txs=0 mgas=0.000 elapsed=5.450ms  mgasps
=0.000 triedirty=0.00B
INFO [04-11|10:48:29.014] Imported new chain segm
ent                    number=38 hash=27182f..2ee9af b
locks=1  txs=0 mgas=0.000 elapsed=3.145ms  mgasps
=0.000 triedirty=0.00B
INFO [04-11|10:48:34.014] Imported new chain segm
ent                    number=39 hash=dc685b..f0fb0b b
locks=1  txs=0 mgas=0.000 elapsed=3.625ms  mgasps
=0.000 triedirty=0.00B
INFO [04-11|10:48:34.143] Looking for peers
                       peercount=1 tried=0 static=0
```

```
                       id=639bf8d973d3567f addr=127.
0.0.1:30304 err=<nil>
TRACE[04-11|10:48:36.571] >> NEIGHBORS/v4
                       id=639bf8d973d3567f addr=127.
0.0.1:30304 err=<nil>
TRACE[04-11|10:48:36.698] << FINDNODE/v4
                       id=90e8fb22684a3640 addr=127.
0.0.1:30306 err=<nil>
TRACE[04-11|10:48:36.699] >> NEIGHBORS/v4
                       id=90e8fb22684a3640 addr=127.
0.0.1:30306 err=<nil>
TRACE[04-11|10:48:37.071] << FINDNODE/v4
                       id=639bf8d973d3567f addr=127.
0.0.1:30304 err=<nil>
TRACE[04-11|10:48:37.073] >> NEIGHBORS/v4
                       id=639bf8d973d3567f addr=127.
0.0.1:30304 err=<nil>
TRACE[04-11|10:48:37.199] << FINDNODE/v4
                       id=90e8fb22684a3640 addr=127.
0.0.1:30306 err=<nil>
TRACE[04-11|10:48:37.201] >> NEIGHBORS/v4
                       id=90e8fb22684a3640 addr=127.
0.0.1:30306 err=<nil>
```

Fig. 6.2 Private Ethereum Blockchain setup

Build smart contract



remix.ethereum.org/#lang=en&optimize=false&ru...

External Http Provider

Note: To use Geth & https://remix.ethereum.org, configure it to allow requests from Remix(see Geth Docs on rpc server)

```
geth --http --http.corsdomain https://remix.ethereum.org
```

To run Remix & a local Geth test node, use this command: (see Geth Docs on Dev mode)

```
geth --http --http.corsdomain="https://remix.ethereum.org" --http.api
web3,eth,debug,personal,net --vmdebug --datadir
<path/to/local/folder/for/test/chain> --dev console
```

WARNING: It is not safe to use the --http.corsdomain flag with a wildcard: **--http.corsdomain ***

For more info: Remix Docs on External HTTP Provider

External HTTP Provider Endpoint: http://127.0.0.1:8545

OK   Cancel

No pinned contracts found for selected workspace & network

Currently you have no unpinned contracts to interact with.

```
commands.
Solidity copilot not activated!
```

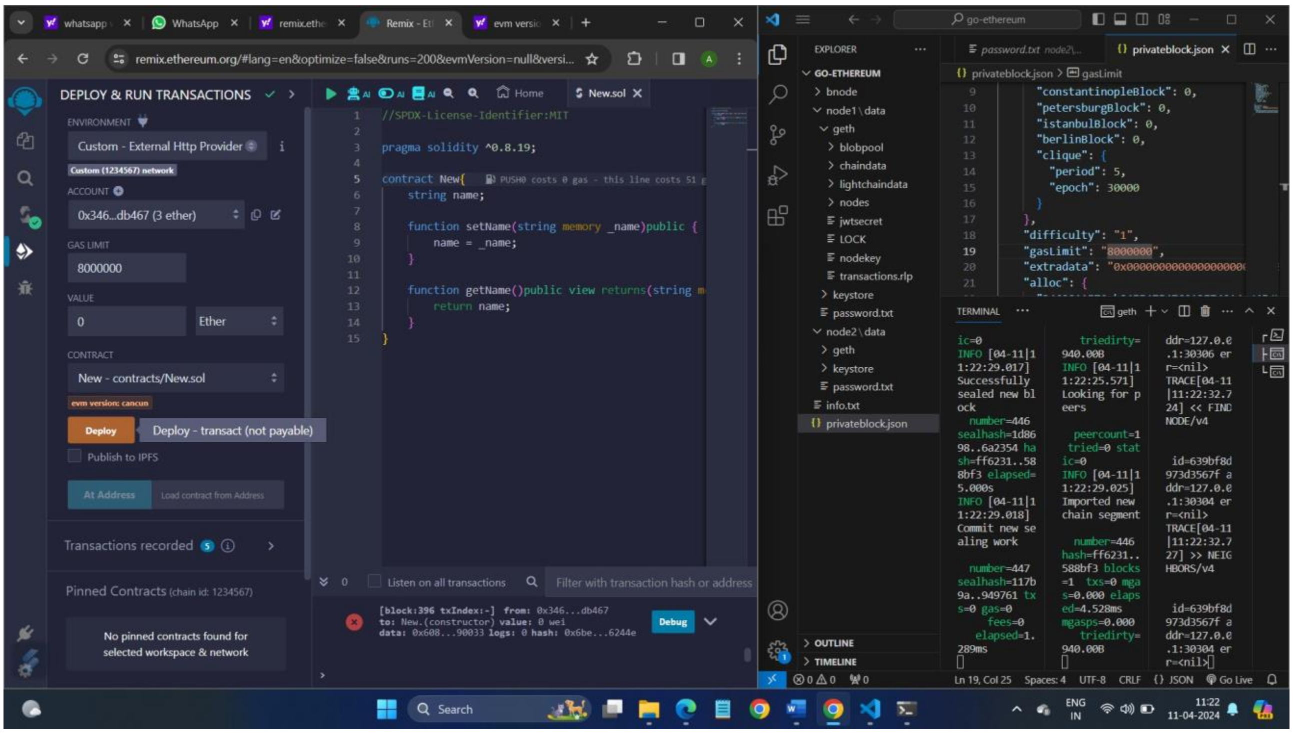
```
ees=0 elapsed="733.
9µs"
INFO [04-11|10:52:4
0.547] Looking for
peers
          peercount
=1 tried=0 static=0
INFO [04-11|10:52:4
4.021] Successfully
 sealed new block
          number=89
 sealhash=e1cadc..9
ef7e1 hash=a25954..
d8dc2d elapsed=4.99
4s
INFO [04-11|10:52:4
4.021] Commit new s
ealing work
          number=90
 sealhash=e760d4..4
31339 txs=0 gas=0 f
ees=0 elapsed="558.
5µs"
```

```
INFO [04-11|10:52:3
7.993] Looking for
peers
          peercount
=1 tried=0 static=0
INFO [04-11|10:52:3
9.033] Imported new
 chain segment
          number=88
 hash=334094..d66fb
d blocks=1  txs=0 m
gas=0.000 elapsed=3
.866ms  mgasps=0.00
0 triedirty=0.00B
INFO [04-11|10:52:4
4.028] Imported new
 chain segment
          number=89
 hash=a25954..d8dc2
d blocks=1  txs=0 m
gas=0.000 elapsed=3
.556ms  mgasps=0.00
0 triedirty=0.00B
```

```
0 addr=127.0.0.1:
30306 err=<nil>
TRACE[04-11|10:52
:47.163] >> NEIGH
BORS/v4
                i
d=90e8fb22684a364
0 addr=127.0.0.1:
30306 err=<nil>
TRACE[04-11|10:52
:47.165] << FINDN
ODE/v4
                i
d=639bf8d973d3567
f addr=127.0.0.1:
30304 err=<nil>
TRACE[04-11|10:52
:47.166] >> NEIGH
BORS/v4
                i
d=639bf8d973d3567
f addr=127.0.0.1:
30304 err=<nil>
```

Fig. 6.3 Build smart contract

Deploy Blockchain


DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Custom - External Http Provider
Custom (1234567) network
ACCOUNT: 0x346...db467 (3 ether)
GAS LIMIT: 8000000
VALUE: 0 Ether
CONTRACT: New - contracts/New.sol
evm version: cancun
Deploy - transact (not payable)
Publish to IPFS
At Address | Load contract from Address
Transactions recorded 5
Pinned Contracts (chain id: 1234567)
No pinned contracts found for selected workspace & network

```
//SPDX-License-Identifier:MIT

pragma solidity ^0.8.19;

contract New{    PUSH0 costs 0 gas - this line costs 51 
    string name;

    function setName(string memory _name)public {
        name = _name;
    }

    function getName()public view returns(string m
        return name;
    }
}
```

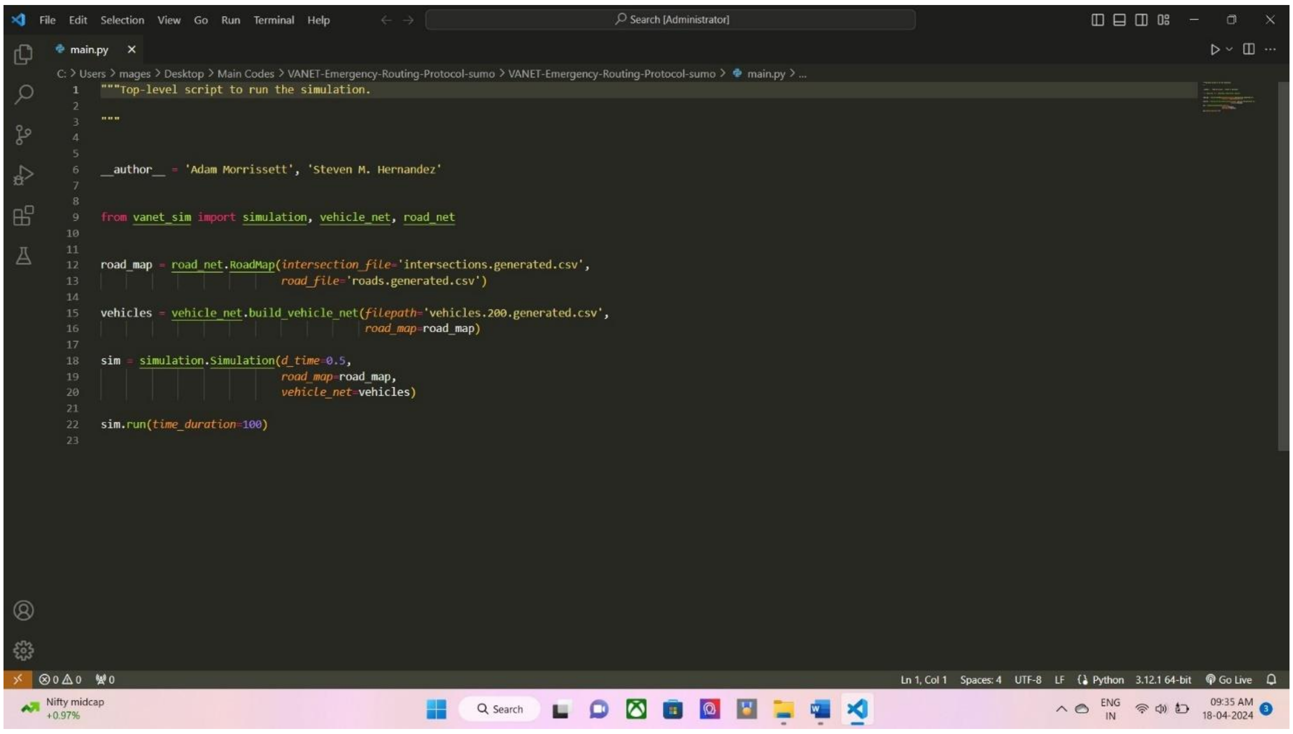
```
[block:396 txIndex:-]  from: 0x346...db467
to: New.(constructor) value: 0 wei
data: 0x608...90033 logs: 0 hash: 0x6be...6244e
```

```
        "constantinopleBlock": 0,
        "petersburgBlock": 0,
        "istanbulBlock": 0,
        "berlinBlock": 0,
        "clique": {
          "period": 5,
          "epoch": 30000
        }
      },
      "difficulty": "1",
      "gasLimit": "8000000",
      "extradata": "0x0000000000000000000
      "alloc": {
```

```
ic=0                 triedirty=          ddr=127.0.0
INFO [04-11|1        940.00B             .1:30306 er
1:22:29.017]         INFO [04-11|1       r=<nil>
Successfully         1:22:25.571]        TRACE[04-11
sealed new bl        Looking for p       |11:22:32.7
ock                  eers                24] << FIND
  number=446                             NODE/v4
sealhash=1d86         peercount=1
98..6a2354 ha        tried=0 stat
sh=ff6231..58        ic=0                 id=639bf8d
8bf3 elapsed=        INFO [04-11|1       973d3567f a
5.000s               1:22:29.025]        ddr=127.0.0
INFO [04-11|1        Imported new        .1:30304 er
1:22:29.018]         chain segment       r=<nil>
Commit new se                            TRACE[04-11
aling work            number=446         |11:22:32.7
                     hash=ff6231..       27] >> NEIG
  number=447         588bf3 blocks       HBORS/v4
sealhash=117b        =1  txs=0 mga
9a..949761 tx        s=0.000 elaps
s=0 gas=0            ed=4.528ms           id=639bf8d
     fees=0          mgasps=0.000        973d3567f a
  elapsed=1.           triedirty=        ddr=127.0.0
289ms                940.00B             .1:30304 er
                                         r=<nil>
```

Fig. 6.4 Deploy Blockchain

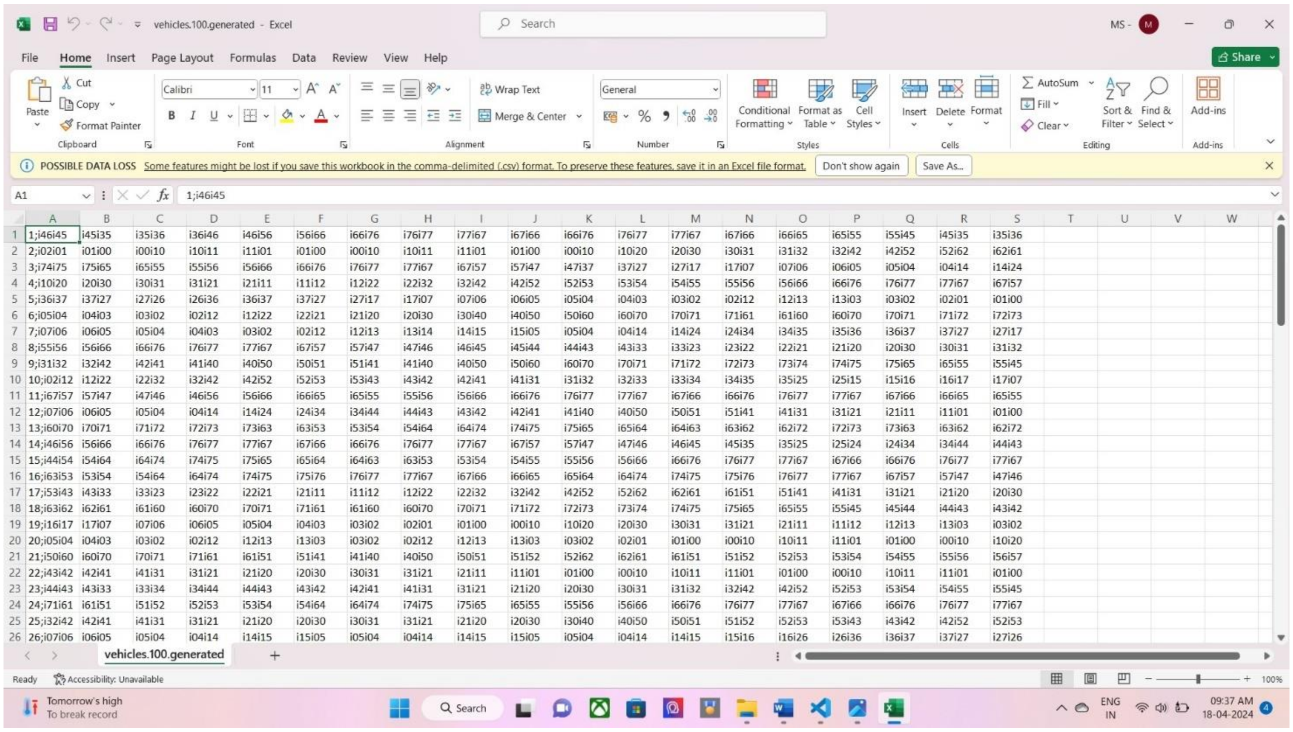
VANET Simulation


C: > Users > mages > Desktop > Main Codes > VANET-Emergency-Routing-Protocol-sumo > VANET-Emergency-Routing-Protocol-sumo > main.py

```python
"""Top-level script to run the simulation.

"""


__author__ = 'Adam Morrissett', 'Steven M. Hernandez'


from vanet_sim import simulation, vehicle_net, road_net


road_map = road_net.RoadMap(intersection_file='intersections.generated.csv',
                            road_file='roads.generated.csv')

vehicles = vehicle_net.build_vehicle_net(filepath='vehicles.200.generated.csv',
                                         road_map=road_map)

sim = simulation.Simulation(d_time=0.5,
                            road_map=road_map,
                            vehicle_net=vehicles)

sim.run(time_duration=100)
```

Fig. 6.5 VANET Simulation

Vehicles Data



vehicles.100.generated - Excel

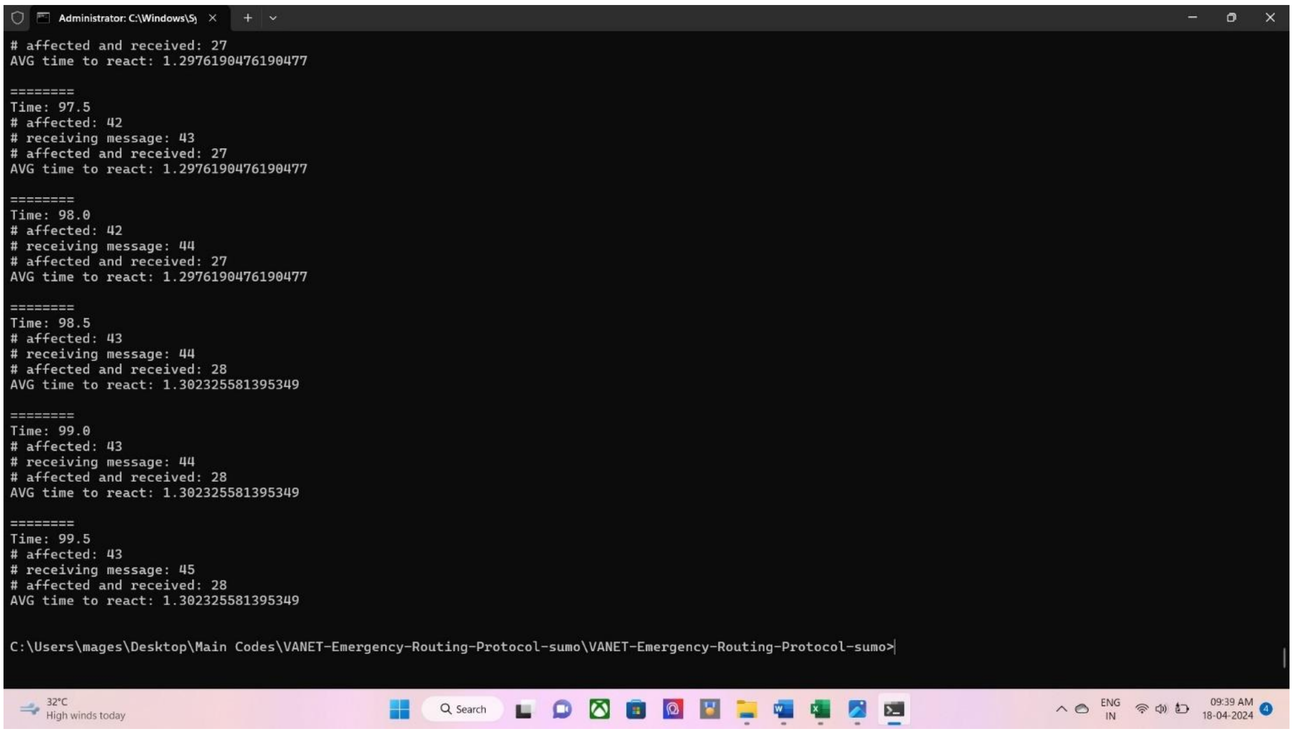
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

A1 1;i46i45

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	1;i46i45	i45i35	i35i36	i36i46	i46i56	i56i66	i66i76	i76i77	i77i67	i67i66	i66i76	i76i77	i77i67	i67i66	i66i65	i65i55	i55i45	i45i35	i35i36
2	2;i02i01	i01i00	i00i10	i10i11	i11i01	i01i00	i00i10	i10i11	i11i01	i01i00	i00i10	i10i20	i20i30	i30i31	i31i32	i32i42	i42i52	i52i62	i62i61
3	3;i74i75	i75i65	i65i55	i55i56	i56i66	i66i76	i76i77	i77i67	i67i57	i57i47	i47i37	i37i27	i27i17	i17i07	i07i06	i06i05	i05i04	i04i14	i14i24
4	4;i10i20	i20i30	i30i31	i31i21	i21i11	i11i12	i12i22	i22i32	i32i42	i42i52	i52i53	i53i54	i54i55	i55i56	i56i66	i66i76	i76i77	i77i67	i67i57
5	5;i36i37	i37i27	i27i26	i26i36	i36i37	i37i27	i27i17	i17i07	i07i06	i06i05	i05i04	i04i03	i03i02	i02i12	i12i13	i13i03	i03i02	i02i01	i01i00
6	6;i05i04	i04i03	i03i02	i02i12	i12i22	i22i21	i21i20	i20i30	i30i40	i40i50	i50i60	i60i70	i70i71	i71i61	i61i60	i60i70	i70i71	i71i72	i72i73
7	7;i07i06	i06i05	i05i04	i04i03	i03i02	i02i12	i12i13	i13i14	i14i15	i15i05	i05i04	i04i14	i14i24	i24i34	i34i35	i35i36	i36i37	i37i27	i27i17
8	8;i55i56	i56i66	i66i76	i76i77	i77i67	i67i57	i57i47	i47i46	i46i45	i45i44	i44i43	i43i33	i33i23	i23i22	i22i21	i21i20	i20i30	i30i31	i31i32
9	9;i31i32	i32i42	i42i41	i41i40	i40i50	i50i51	i51i41	i41i40	i40i50	i50i60	i60i70	i70i71	i71i72	i72i73	i73i74	i74i75	i75i65	i65i55	i55i45
10	10;i02i12	i12i22	i22i32	i32i42	i42i52	i52i53	i53i43	i43i42	i42i41	i41i31	i31i32	i32i33	i33i34	i34i35	i35i25	i25i15	i15i16	i16i17	i17i07
11	11;i67i57	i57i47	i47i46	i46i56	i56i66	i66i65	i65i55	i55i56	i56i66	i66i76	i76i77	i77i67	i67i66	i66i76	i76i77	i77i67	i67i66	i66i65	i65i55
12	12;i07i06	i06i05	i05i04	i04i14	i14i24	i24i34	i34i44	i44i43	i43i42	i42i41	i41i40	i40i50	i50i51	i51i41	i41i31	i31i21	i21i11	i11i01	i01i00
13	13;i60i70	i70i71	i71i72	i72i73	i73i63	i63i53	i53i54	i54i64	i64i74	i74i75	i75i65	i65i64	i64i63	i63i62	i62i72	i72i73	i73i63	i63i62	i62i72
14	14;i46i56	i56i66	i66i76	i76i77	i77i67	i67i66	i66i76	i76i77	i77i67	i67i57	i57i47	i47i46	i46i45	i45i35	i35i25	i25i24	i24i34	i34i44	i44i43
15	15;i44i54	i54i64	i64i74	i74i75	i75i65	i65i64	i64i63	i63i53	i53i54	i54i55	i55i56	i56i66	i66i76	i76i77	i77i67	i67i66	i66i76	i76i77	i77i67
16	16;i63i53	i53i54	i54i64	i64i74	i74i75	i75i76	i76i77	i77i67	i67i66	i66i65	i65i64	i64i74	i74i75	i75i76	i76i77	i77i67	i67i57	i57i47	i47i46
17	17;i53i43	i43i33	i33i23	i23i22	i22i21	i21i11	i11i12	i12i22	i22i32	i32i42	i42i52	i52i62	i62i61	i61i51	i51i41	i41i31	i31i21	i21i20	i20i30
18	18;i63i62	i62i61	i61i60	i60i70	i70i71	i71i61	i61i60	i60i70	i70i71	i71i72	i72i73	i73i74	i74i75	i75i65	i65i55	i55i45	i45i44	i44i43	i43i42
19	19;i16i17	i17i07	i07i06	i06i05	i05i04	i04i03	i03i02	i02i01	i01i00	i00i10	i10i20	i20i30	i30i31	i31i21	i21i11	i11i12	i12i13	i13i03	i03i02
20	20;i05i04	i04i03	i03i02	i02i12	i12i13	i13i03	i03i02	i02i12	i12i13	i13i03	i03i02	i02i01	i01i00	i00i10	i10i11	i11i01	i01i00	i00i10	i10i20
21	21;i50i60	i60i70	i70i71	i71i61	i61i51	i51i41	i41i40	i40i50	i50i51	i51i52	i52i62	i62i61	i61i51	i51i52	i52i53	i53i54	i54i55	i55i56	i56i57
22	22;i43i42	i42i41	i41i31	i31i21	i21i20	i20i30	i30i31	i31i21	i21i11	i11i01	i01i00	i00i10	i10i11	i11i01	i01i00	i00i10	i10i11	i11i01	i01i00
23	23;i44i43	i43i33	i33i34	i34i44	i44i43	i43i42	i42i41	i41i31	i31i21	i21i20	i20i30	i30i31	i31i32	i32i42	i42i52	i52i53	i53i54	i54i55	i55i45
24	24;i71i61	i61i51	i51i52	i52i53	i53i54	i54i64	i64i74	i74i75	i75i65	i65i55	i55i56	i56i66	i66i76	i76i77	i77i67	i67i66	i66i76	i76i77	i77i67
25	25;i32i42	i42i41	i41i31	i31i21	i21i20	i20i30	i30i31	i31i21	i21i20	i20i30	i30i40	i40i50	i50i51	i51i52	i52i53	i53i43	i43i42	i42i52	i52i53
26	26;i07i06	i06i05	i05i04	i04i14	i14i15	i15i05	i05i04	i04i14	i14i15	i15i05	i05i04	i04i14	i14i15	i15i16	i16i26	i26i36	i36i37	i37i27	i27i26

Fig. 6.6 Vehicles Data

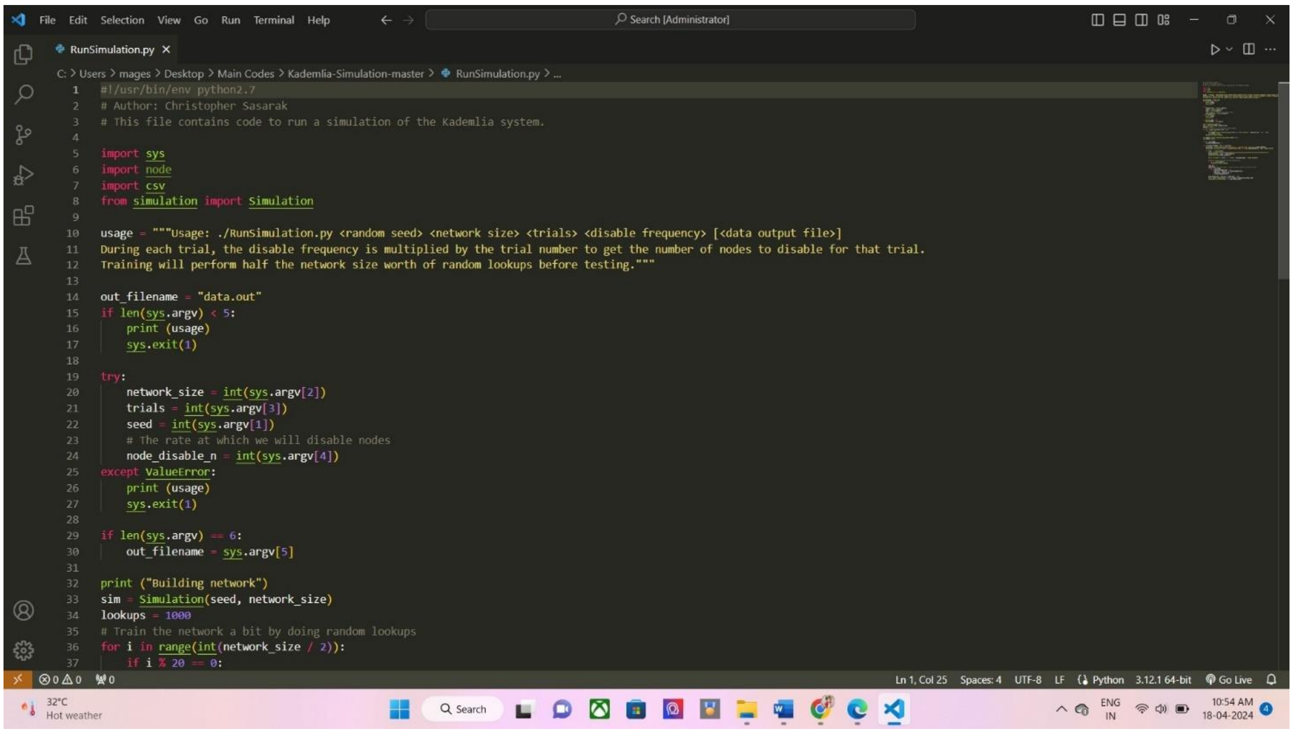
Vehicles Connection



```
# affected and received: 27
AVG time to react: 1.2976190476190477

========
Time: 97.5
# affected: 42
# receiving message: 43
# affected and received: 27
AVG time to react: 1.2976190476190477

========
Time: 98.0
# affected: 42
# receiving message: 44
# affected and received: 27
AVG time to react: 1.2976190476190477

========
Time: 98.5
# affected: 43
# receiving message: 44
# affected and received: 28
AVG time to react: 1.302325581395349

========
Time: 99.0
# affected: 43
# receiving message: 44
# affected and received: 28
AVG time to react: 1.302325581395349

========
Time: 99.5
# affected: 43
# receiving message: 45
# affected and received: 28
AVG time to react: 1.302325581395349


C:\Users\mages\Desktop\Main Codes\VANET-Emergency-Routing-Protocol-sumo\VANET-Emergency-Routing-Protocol-sumo>
```

Fig. 6.7 Vehicles Connection

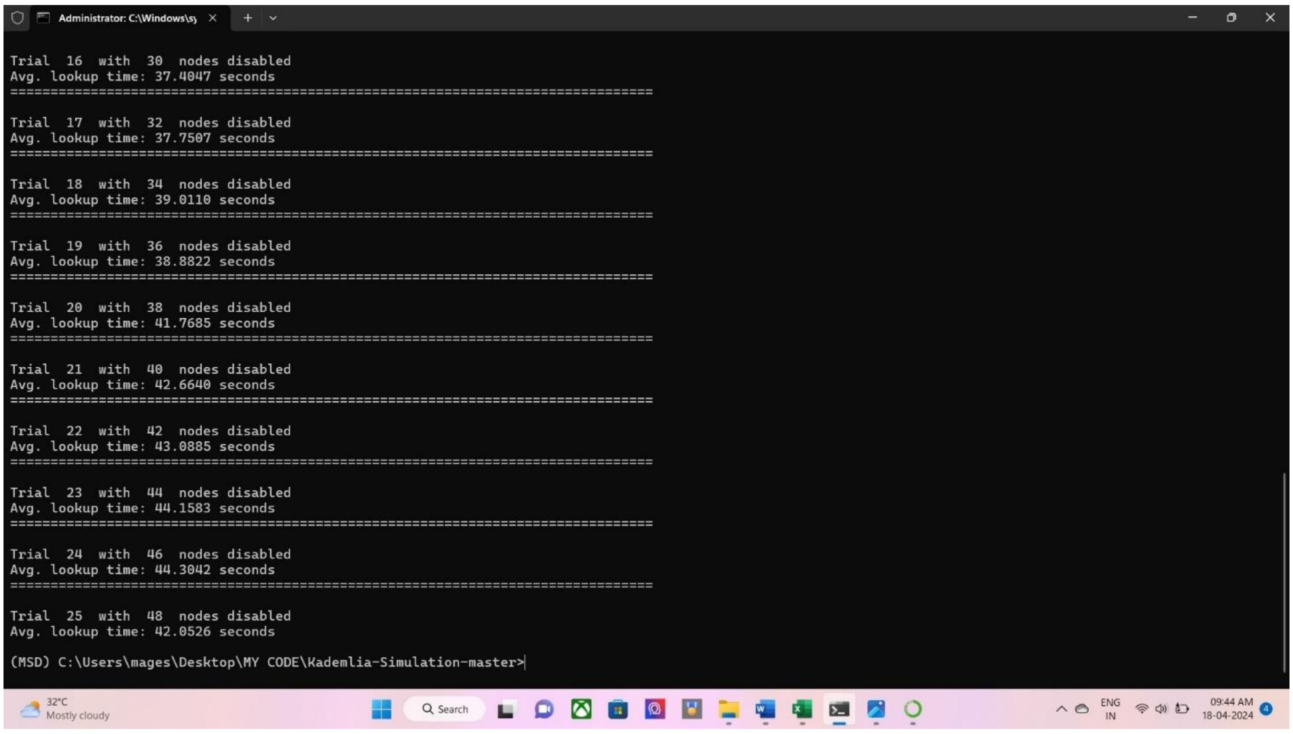
Kademlia Dataset



```python
#!/usr/bin/env python2.7
# Author: Christopher Sasarak
# This file contains code to run a simulation of the Kademlia system.

import sys
import node
import csv
from simulation import Simulation

usage = """Usage: ./RunSimulation.py <random seed> <network size> <trials> <disable frequency> [<data output file>]
During each trial, the disable frequency is multiplied by the trial number to get the number of nodes to disable for that trial.
Training will perform half the network size worth of random lookups before testing."""

out_filename = "data.out"
if len(sys.argv) < 5:
    print (usage)
    sys.exit(1)

try:
    network_size = int(sys.argv[2])
    trials = int(sys.argv[3])
    seed = int(sys.argv[1])
    # The rate at which we will disable nodes
    node_disable_n = int(sys.argv[4])
except ValueError:
    print (usage)
    sys.exit(1)

if len(sys.argv) == 6:
    out_filename = sys.argv[5]

print ("Building network")
sim = Simulation(seed, network_size)
lookups = 1000
# Train the network a bit by doing random lookups
for i in range(int(network_size / 2)):
    if i % 20 == 0:
```

Fig. 6.8 Kademlia Dataset

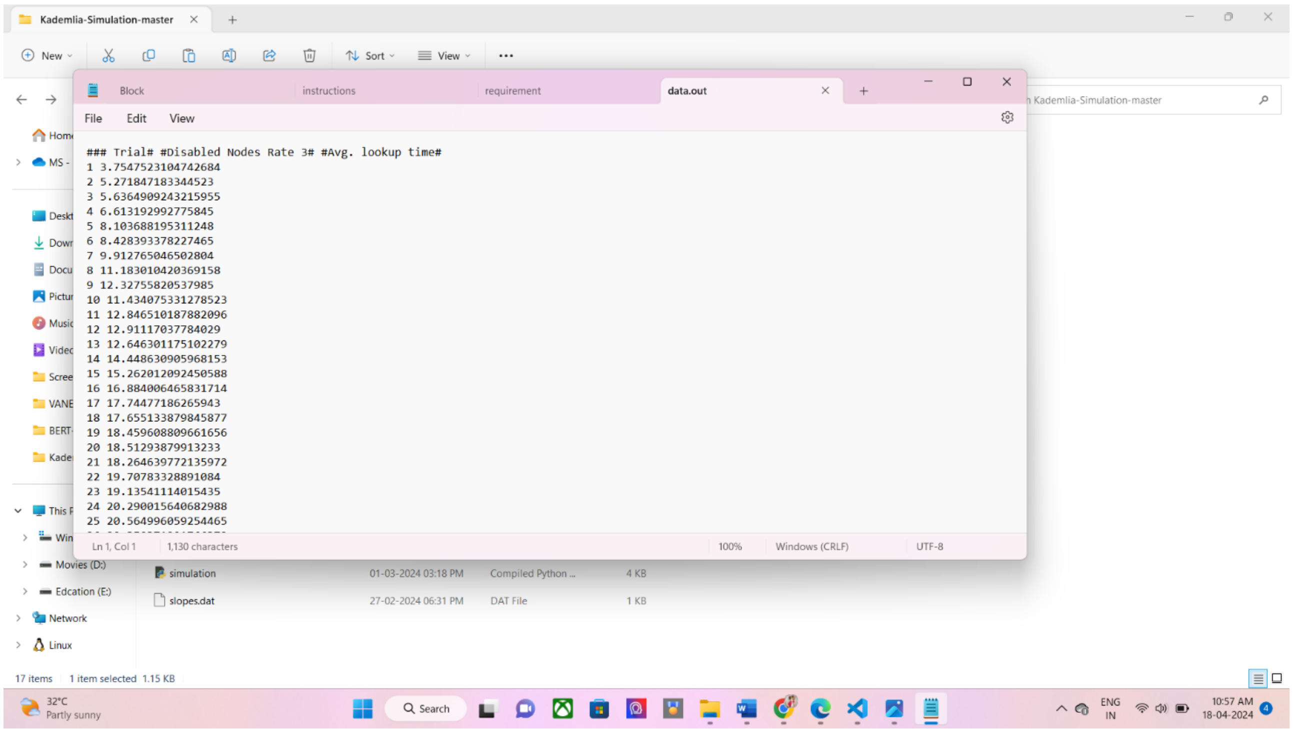
Data Forwarding



```
Trial  16  with  30  nodes disabled
Avg. lookup time: 37.4047 seconds
================================================================================

Trial  17  with  32  nodes disabled
Avg. lookup time: 37.7507 seconds
================================================================================

Trial  18  with  34  nodes disabled
Avg. lookup time: 39.0110 seconds
================================================================================

Trial  19  with  36  nodes disabled
Avg. lookup time: 38.8822 seconds
================================================================================

Trial  20  with  38  nodes disabled
Avg. lookup time: 41.7685 seconds
================================================================================

Trial  21  with  40  nodes disabled
Avg. lookup time: 42.6640 seconds
================================================================================

Trial  22  with  42  nodes disabled
Avg. lookup time: 43.0885 seconds
================================================================================

Trial  23  with  44  nodes disabled
Avg. lookup time: 44.1583 seconds
================================================================================

Trial  24  with  46  nodes disabled
Avg. lookup time: 44.3042 seconds
================================================================================

Trial  25  with  48  nodes disabled
Avg. lookup time: 42.0526 seconds

(MSD) C:\Users\mages\Desktop\MY CODE\Kademlia-Simulation-master>
```

Fig. 6.9 Data Forwarding

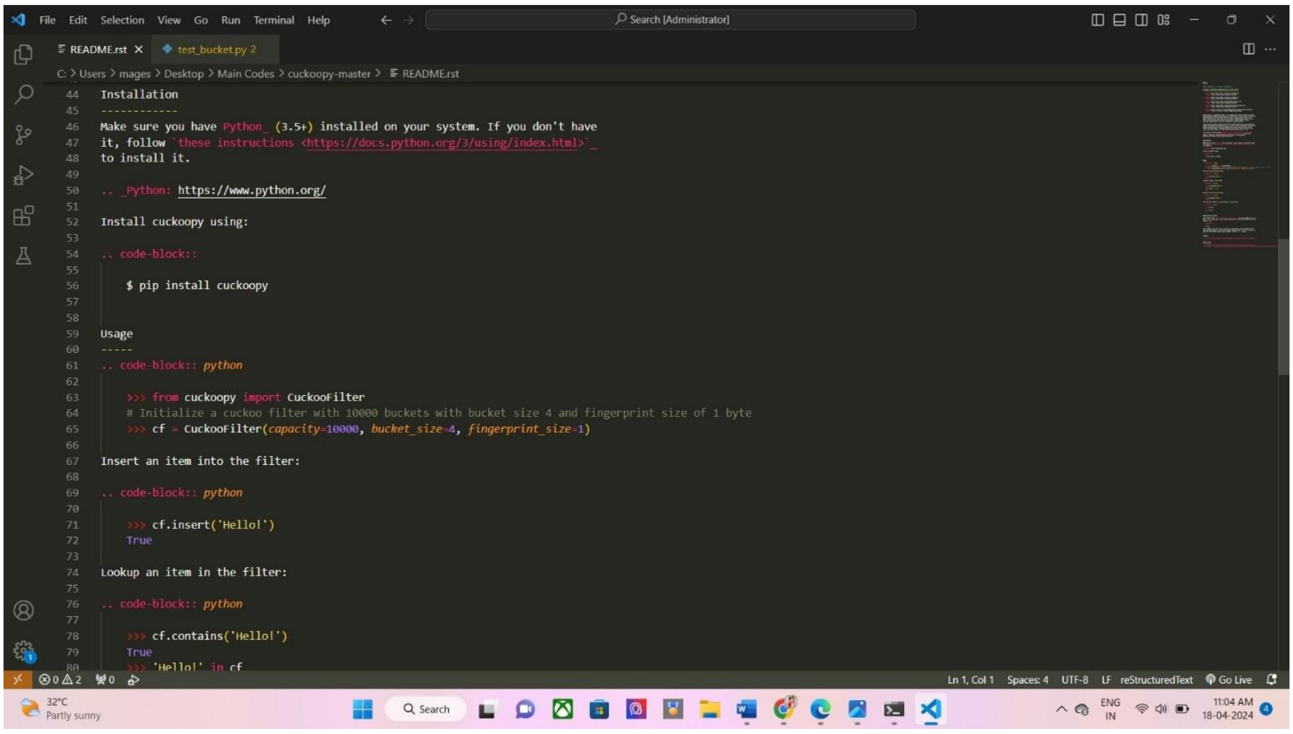
Output



```
### Trial# #Disabled Nodes Rate 3# #Avg. lookup time#
1 3.7547523104742684
2 5.271847183344523
3 5.6364909243215955
4 6.613192992775845
5 8.103688195311248
6 8.428393378227465
7 9.912765046502804
8 11.183010420369158
9 12.32755820537985
10 11.434075331278523
11 12.846510187882096
12 12.91117037784029
13 12.646301175102279
14 14.448630905968153
15 15.262012092450588
16 16.884006465831714
17 17.74477186265943
18 17.655133879845877
19 18.459608809661656
20 18.51293879913233
21 18.264639772135972
22 19.70783328891084
23 19.13541114015435
24 20.290015640682988
25 20.564996059254465
```

Fig. 6.10 Output

Cuckoo Filter



```
Installation
------------
Make sure you have Python_ (3.5+) installed on your system. If you don't have
it, follow `these instructions <https://docs.python.org/3/using/index.html>`_
to install it.

.. _Python: https://www.python.org/

Install cuckoopy using:

.. code-block::

    $ pip install cuckoopy


Usage
-----
.. code-block:: python

    >>> from cuckoopy import CuckooFilter
    # Initialize a cuckoo filter with 10000 buckets with bucket size 4 and fingerprint size of 1 byte
    >>> cf = CuckooFilter(capacity=10000, bucket_size=4, fingerprint_size=1)

Insert an item into the filter:

.. code-block:: python

    >>> cf.insert('Hello!')
    True

Lookup an item in the filter:

.. code-block:: python

    >>> cf.contains('Hello!')
    True
    >>> 'Hello!' in cf
```

Fig. 6.11 Cuckoo Fiter

CHAPTER 7

TESTING

7.1 TESTING OBJECTIVES

Testing is performed to identify errors. A good test case is one that has a high probability of finding an undiscovered error. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases: assemblies and/or a finished product. It is the process of exercising software.

- Static analysis is used to investigate the structural properties of the source.
- Dynamic testing is used to investigate the behavior of the source code.

7.2 TYPES OF TESTING

7.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the Internal program logic is functioning properly, and that program input produces valid output. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the 38 completions of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined input and expected results.

7.2.2 Integration Testing

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the setoff modules which make up the project. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs.

The new module and its interring communications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.2.3 Functional Testing

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values, such as logically related inputs, files of identical elements, and empty files. Three type of test in functional test

- Performance Test
- Stress Test
- Structure Test
- System Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is cantered on the following items.

- Valid Input: identified classes of valid input must be accepted.
- Invalid Input: identified classes of invalid input must be rejected.
- Functions: identified functions must be exercised.
- Output: identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

7.2.3.1 Performance Testing

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit. The Performance test ensures that the output is produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results, the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

7.2.3.2 Stress Testing

Stress test is the test designed to intentionally break the unit. A great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.2.3.3 Structure Testing

Structure tests are concerned with exercising the internal logic of a program and traversing particular execution path. The way in which White-Box test strategy was employed to ensure that the test cases could guarantee that all independent paths within a module have been exercise at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.

Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs.

7.2.3.4 System Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability if finding an as-yet- undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation which is and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed foe user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete. Software testing is the critical element of software quality coding. Testing is the process of executing the program with the intent of finding the error. A successful test is one that undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one of the two ways.

7.3 TESTING TECHNIQUES

7.3.1 Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as- yet undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Testing is the process of executing the program with the intent of finding the errors. A good test case design is one that has a high probability of finding the errors. A successful test is one that uncovers an undiscovered error. Any engineering product can be tested in one of the following two ways:

7.3.2 White Box Testing

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

- Flow graph notation
- Cyclometric complexity

7.3.3 Black Box Testing

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. It focuses on ensuring that the system behaves correctly from a user's perspective and meets the specified requirements.

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

7.3.3.1 Boundary Value Analysis

Boundary value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range. The idea comes from the boundary. Given that we have a set of test vectors to test the system, a topology can be defined on that set.

7.3.3.2 Equivalence partitioning

Equivalence partitioning or equivalence class partitioning (ECP) is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. In principle, test cases are designed to cover each partition at least once.

7.3.3.3 Comparison Testing

Comparison testing involves comparing the contents of databases files, folders, etc. A comparison testing can be carried out in two ways. Direct comparison testing: It is a testing of particular parts of each site against each other. Usually, multiple sites are compared side by side. Objective comparison testing: Execute the same test for each site separately. Usually, one site at a time is tested.

7.4 Test Strategy and Approach

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For the reason a template for software testing a set step into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer base system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.

7.4.1 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together” interfacing. There may be the chances of data lost across on another’s sub functions, when combined may produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; goals data structure can present problems.

7.4.2 User Acceptance Testing

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developed.

CHAPTER 8

RESULTS AND DISCUSSION

8.1 PERFORMANCE ANALYSIS

8.1.1 Communication overhead comparison

One of the major problems consider for our work is the communication overhead problem. Kademlia algorithm is used in our prototype for the congestion control. The comparison is done between the following existing literatures.

- Yang et al
- CADC

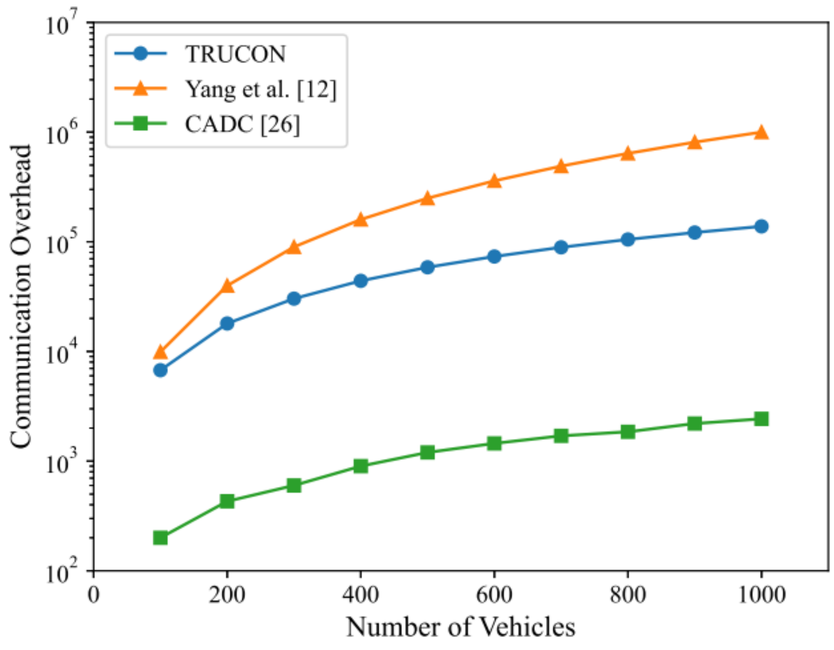


Fig. 8.1 Communication Overhead comparison

Results clearly shows that our prototype is better than Yang et al but not as CADC. Instead, we have trust management where CADC does not. So, the Blockchain based trust management system in our work is more feasible. Kademlia algorithm is implemented and run successfully.

8.1.2 Gas Consumption

In Ethereum, transaction initiator needs to pay the miners for computation costs. And the total fee is determined by specified gas price and limit. The following shows each cost of the operations.

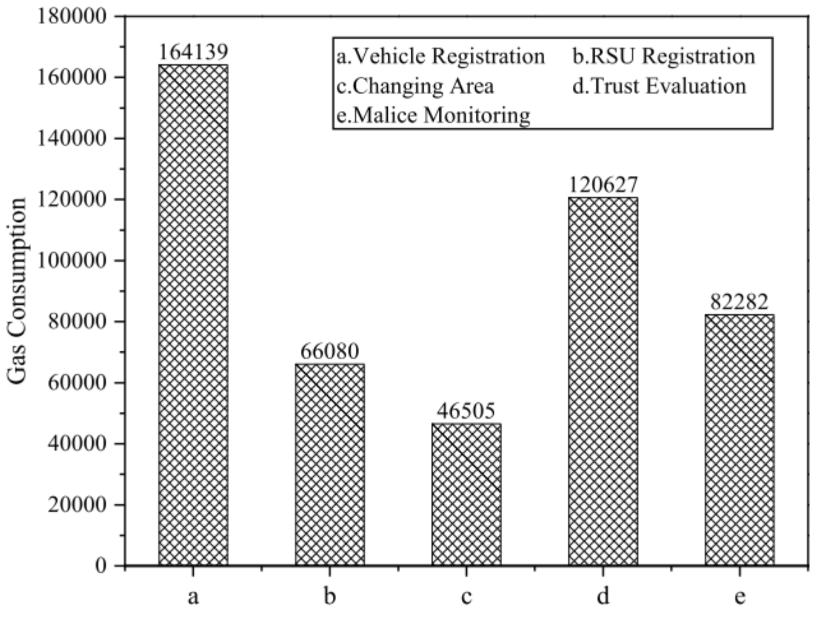


Fig. 8.2 Gas Consumption

8.1.3 Trust Value Calculation

Types of Vehicles

- Lazy vehicles
- Honest vehicles
- Malicious vehicles

Lazy vehicles

Lazy vehicles are selfish, which means they don't share data or give scores while receiving data from others. It is not good for the stability and sustainability of the whole system.

Honest Vehicles

Honest vehicles share the proper data with the RSU and the other vehicles. They give correct trust values to the source vehicles without any mis-consideration. The trust value of these vehicles keeps increasing due to their honesty.

Malicious vehicles

If any vehicle shares unwanted or fake data then it is said to be a malicious vehicle due to its behaviour. These vehicles get reported by the RSU and the other vehicles.

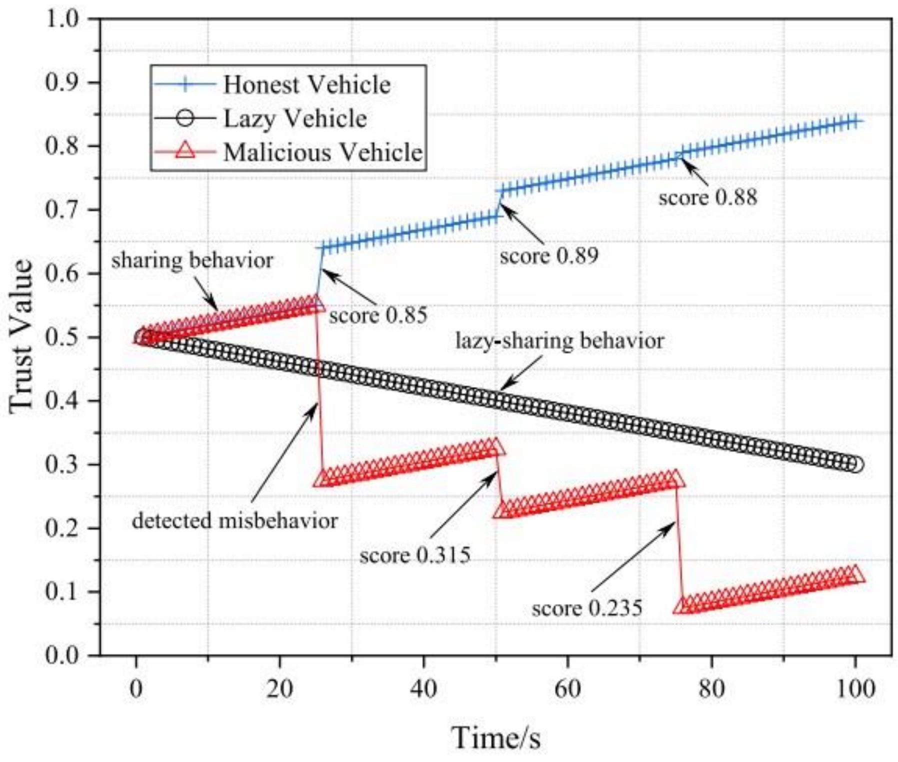


Fig. 8.3 Trust value calculation

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

To ensure the safety of the traffic and the accidents avoidance, a blockchain based trusted data sharing mechanism with congestion control is built to solve the trust problems existing in IoV. The system initialization phase consists of the deploying blockchain. Next for the congestion control mechanism we use the Kademlia data forwarding algorithm to share data and Cuckoo filter to share deduplicated data to RSU. Atlast the trust value calculation is done by the vehicles and RSU. Our system works efficiently while comparing the existing literatures because we consist both the trust management and the congestion control mechanism.

9.2 Future Work

Future work in the realm of blockchain-based trusted data sharing with congestion control in the Internet of Vehicles (IoV) could explore several avenues for further enhancement and development:

1.) Scalability

2.) Security Enhancements

3.) Privacy preservation

The shortcoming of our current work is that applicable scenarios are limited and we did not take privacy issues into consideration. The effect of congestion control is limited when there are few vehicles. In the future, we will keep trying to reduce the communication overhead. Besides, we will also work to protect the security of private information of vehicles such as their real identity and location.

REFERENCES

1. A. K. Sangaiah, J. S. Ramamoorthi, J. J. P. C. Rodrigues, and M. Alrashoud. (2021) "LACCVoV: Linear adaptive congestion control with optimization of data dissemination model in vehicle-to-vehicle communication," IEEE Transaction. Intelligent. Transportation System, vol. 22, no. 8, pp. 5319–5328.

2. A. Hbaieb, S. Ayed, and L. Chaari. (2022) "A survey of trust management in the Internet of Vehicles," Computer Networks, vol. 203, Art. no. 108558.

3. B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher. (2014) "Cuckoo filter: Practically better than Bloom," in Proc. 10th ACM International Conference. Technology., pp. 75–88.

4. C. Lai, K. Zhang, N. Cheng, H. Li, and X. Shen. (2017) "SIRC: A secure incentive scheme for reliable cooperative downloading in highway VANETs," IEEE Transaction. Intelligent Transportation System, vol. 18, no. 6, pp. 1559–1574.

5. C. Boudagdigue, A. Benslimane and M. Elmachkour. (2018) "A distributed advanced analytical trust model for IoT," in Proc. IEEE International Conference Communication. pp. 1–6.

6. C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang. (2021) "A blockchain based multi-cloud storage data auditing scheme to locate faults," IEEE Transaction. Cloud Computing, early access.

7. C. Feng, B. Liu, K. Yu, S. K. Goudos, and S. Wan. (2022) "Blockchain empowered decentralized horizontal federated learning for 5G-enabled UAVs," IEEE Transaction vol. 18, no. 5, pp. 3582–3592.

8. G. E. M. Zhioua, N. Tabbane, H. Labiod, and S. Tabbane. (2015) “A fuzzy multi-metric QoS-balancing gateway selection algorithm in a clustered VANET to LTE advanced hybrid cellular network,” IEEE Transaction. Vehicle Technology, vol. 64, no. 2, pp. 804–817.

9. G. Luo et al. (2021) “Software-defined cooperative data sharing in edge computing assisted 5G-VANET,” IEEE Transaction. Mobile Computing, vol. 20, no. 3, pp. 1212–1229.

10. I. García-Magariño, S. Sendra, R. Lacuesta, and J. Lloret. (2019) “Security in vehicles with IoT by prioritization rules, vehicle certificates, and trust management,” IEEE Internet Things J., vol. 6, no. 4, pp. 5927–5934.

11. J. Wang et al. (2018) “Dynamic clustering and cooperative scheduling for vehicle-to-vehicle communication in bidirectional road scenarios,” IEEE Transaction Intelligent Transportation System, vol. 19, no. 6, pp. 1913–1924.

12. J. Kang et al. (2019) “Blockchain for secure and efficient data sharing in vehicular edge computing and networks,” IEEE Internet Things J., vol. 6, no. 3, pp. 4660–4670.

13. J. Sun, H. Xiong, S. Zhang, X. Liu, J. Yuan, and R. H. Deng. (2020) “A secure flexible and tampering-resistant data sharing system for vehicular social networks,” IEEE Transaction. Veh. Technol., vol. 69, no. 11, pp. 12938–12950.

14. M. E. Mahmoud and X. Shen. (2011) “An integrated stimulation and punishment mechanism for thwarting packet dropping attack in multihop wireless networks,” IEEE Transaction Vehicle Technology, vol. 60, no. 8, pp. 3947–3962.

15. P. Maymounkov and D. Mazieres. (2002) “Kademlia: A peer-to-peer information system based on the XOR metric,” in Proc. Int. Workshop Peer to Peer System Cambridge, MA, USA: Springer, pp. 53–65.

16. P. K. Singh, R. Singh, S. K. Nandi, K. Z. Ghafoor, D. B. Rawat, and S. Nandi. (2020) “Blockchain-based adaptive trust management in Internet of Vehicles using smart contract,” IEEE Transaction Intelligent Transportation System, vol. 22, no. 6, pp. 3616–3630.

17. Q. Li, A. Malip, K. M. Martin, S.-L. Ng, and J. Zhang. (2012) “A reputation-based announcement scheme for VANETs,” IEEE Transaction Vehicular Technology, vol. 61, no. 9, pp. 4095–4108.

18. S. Abbes and S. Rekhis. (2021) “A blockchain-based solution for reputation management in IoV,” in Proc. Int. Wireless Communication Mobile Computing. (IWCMC), pp. 1129–1134.

19. S. Duan et al. (2022) “Multitype highway mobility analytics for efficient learning model design: A case of station traffic prediction,” IEEE Transaction Intelligent Transportation System, vol. 23, no. 10, pp. 19484–19496.

20. S. Liu, J. Yu, X. Deng, and S. Wan. (2022) “FedCPF: An efficient communication federated learning approach for vehicular edge computing in 6G communication networks,” IEEE Transaction Intelligent Transportation System, vol. 23, no. 2, pp. 1616–1629.

21. X. Liu, H. Huang, F. Xiao, and Z. Ma. (2020) “A blockchain-based trust management with conditional privacy-preserving announcement scheme for VANETs,” IEEE Internet Things J., vol. 7, no. 5, pp. 4101–4112.

22. Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang. (2020) “Blockchain empowered arbitrable data auditing scheme for network storage as a service,” IEEE Transaction Services Computing, vol. 13, no. 2, pp. 289–300.

23. Y. Xu, Z. Liu, C. Zhang, J. Ren, Y. Zhang, and X. Shen. (2022) “Blockchain based trustworthy energy dispatching approach for high renewable energy penetrated power systems,” IEEE Internet Things J., vol. 9, no. 12, pp. 10036–10047.

24. Z. Xu, W. Liang, K.-C. Li, J. Xu, and H. Jin. (2021) “A blockchain-based roadside unit-assisted authentication and key agreement protocol for Internet of Vehicles,” J. Parallel Distributed Computing, vol. 149, pp. 29–39.

APPENDIX

(a) Blockchain setup

Genesis.json

```
{
  "config": {
    "chainId": 12345,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "muirGlacierBlock": 0,
    "berlinBlock": 0,
    "londonBlock": 0,
    "arrowGlacierBlock": 0,
    "grayGlacierBlock": 0,
    "clique": {
      "period": 5,
      "epoch": 30000
      }
    },
    "difficulty": "1",
    "gasLimit": "800000000",
    "extradata":
"0x00000000000000000000000000000000000000000000000000000000000000007d
f9a875a174b3bc565e6424a0050ebc1b2d1d82000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000",
    "alloc": {
      "Aab411fD321f113d9966ecf12D184Ab34d6171c1": { "balance": "500000" },
      "CbbeF9f43d2FC93d7d022580fA690CF680772232": { "balance": "500000" }
    }
  }
```

(b) VANET simulation

```
"""Top-level script to run the simulation.

"""

__author__ = 'Adam Morrissett', 'Steven M. Hernandez'
from vanet_sim import simulation, vehicle_net, road_net
```

```
road_map = road_net.RoadMap(intersection_file='intersections.generated.csv',
                            road_file='roads.generated.csv')

vehicles = vehicle_net.build_vehicle_net(filepath='vehicles.200.generated.csv',
                                          road_map=road_map)

sim = simulation.Simulation(d_time=0.5,
                            road_map=road_map,
                            vehicle_net=vehicles)

sim.run(time_duration=100)
```

(c) Kademlia Simulation

```
#!/usr/bin/env python2.7
# Author: Christopher Sasarak
# This file contains code to run a simulation of the Kademlia system.

import sys
import node
import csv
```

```
from simulation import Simulation


usage = """Usage: ./RunSimulation.py <random seed> <network size> <trials>
<disable frequency> [<data output file>]
During each trial, the disable frequency is multiplied by the trial number to get the
number of nodes to disable for that trial.
Training will perform half the network size worth of random lookups before
testing."""


out_filename = "data.out"
if len(sys.argv) < 5:
    print (usage)
    sys.exit(1)


try:
    network_size = int(sys.argv[2])
    trials = int(sys.argv[3])
    seed = int(sys.argv[1])


 # The rate at which we will disable nodes
    node_disable_n = int(sys.argv[4])
except ValueError:
    print (usage)
    sys.exit(1)


if len(sys.argv) == 6:
    out_filename = sys.argv[5]


print ("Building network")
sim = Simulation(seed, network_size)
lookups = 1000
# Train the network a bit by doing random lookups
for i in range(int(network_size / 2)):
    if i % 20 == 0:
        sys.stdout.write("\rInitializing network {0:.1f}%".format(i / (network_size / 2.0)
* 100))
        sys.stdout.flush()
    sim.perform_node_lookup()


sys.stdout.write("\rInitializing network 100% \n\n")
sys.stdout.flush()


for n in sim.nodes:
    n.totalLookupDuration = 0


with open(out_filename, 'wb') as csv_file:
    csv_writer = csv.writer(csv_file, delimiter=' ', quotechar='#',
quoting=csv.QUOTE_MINIMAL)

    csv_writer.writerow(["# Trial", "Disabled Nodes Rate {}".format(node_disable_n),
"Avg. lookup time"])

    for trial in range(trials):
        print
("==========================================================
=====================")
        disabled_nodes = trial * node_disable_n
        sim.disable_nodes(node_disable_n)

        print ("\nTrial "), trial + 1, " with ", disabled_nodes, " nodes disabled"

        # Perform lookups and then get the average time
        for i in range(lookups):
            sim.perform_node_lookup()

        time_sum = 0
        node_count = 0 # The number of nodes that had lookups performed on them
        for n in sim.nodes:
            if n.totalLookupDuration != 0:
```

```
 time_sum = time_sum + n.totalLookupDuration
            node_count = node_count + 1
            n.totalLookupDuration = 0

      avg_lookup_time = time_sum / node_count / 1000
      print ("Avg. lookup time: {0:.4f} seconds").format(avg_lookup_time)
      csv_writer.writerow([trial + 1, avg_lookup_time])

# Author: Christopher Sasarak
# This file contains a simulation class. Simulation objects
# can be used to run different simulations

import random
import hashlib
import math
import kademliaConstants
from node import Node, make_hash

class Simulation:

   def __init__(self, seed, init_size=None):
      """
      This constructor will set up a Simulation object by populating it with
      an initial network.

      init_size -- Option parameter which defines an initial network size. If it
                   is not provided, then kademliaConstants.initial_network_size will be
used.
      seed -- Seed the random number generator used by this Simulation
      """
      self.rand = random.Random()
      self.rand.seed(seed)

      self.init_size = init_size if init_size else kademliaConstants.initial_network_size

      self.nodes = None
      self.build_network()

   def perform_node_lookup(self):
      """
      Perform a node lookup between two random nodes, return the
      time that it took to complete the lookup.
      """
      node = self.rand.choice(self.nodes)
      # Generate a random key to look for, use random_IP as a random
      # string generator
      target_key = hashlib.sha1(self.random_IP()).hexdigest()
      target_key = int(target_key, 16)
      target_key = target_key % (int(math.pow(2,
kademliaConstants.bit_string_size)))

      node.lookup_node(target_key)




   def build_network(self, n = None):
      """
      Build an initial network with, IDs are determined by a random
      function, and k-buckets are empty. Running this multiple times will replace the
      current network each time.

      n -- Optional parameter to set the initial size. If not provided, then it will use
          the size parameter set by the containing Simulation object.
      """
      size = n if n else self.init_size
      self.nodes = list()
      for i in range(0, size):
```

```
new_node = Node(self.random_IP(), self.rand)


          # Select a random existing node if this isn't the first node,
          # then use it to perform a network join
          if not i == 0:
              contact_node = self.rand.choice(self.nodes)
              # Update new_node with the random contact node
              new_node.join_network(contact_node)


          self.nodes.append(new_node)


   def random_IP(self):
      """
      Generate a random IP address.


      returns a string representation of an IP address.
      """
      octets = list()


      for i in range(0, 4):
          octets.append(str(self.rand.randint(0, 255)))
```

```
    return ".".join(octets)


def __str__(self):
    """
    Create a string representation of the simulation.


    """
    string = "Simulation: \n"


    for n in self.nodes:
        string = "{} {}\n".format(string, str(n))


    return string


def disable_nodes(self, n):
    """
    Disable n nodes at random in the system.


    n -- The number of nodes to disable in this simulation.
    """
    nodes = self.rand.sample(self.nodes, n)
```

```
        for n in nodes:
            n.disabled = True
```

(d) Cuckoo Filter

```
#!/usr/bin/env python3

from cuckoopy import __version__, __author__
try:
    from setuptools import setup
except ImportError:
    from distutils.core import setup

with open('README.rst') as f:
        readme = f.read()

setup(name='cuckoopy',
      version=__version__,
      description='Cuckoo Filter implementation in Python',
      long_description=readme,
      author=__author__,
      author_email='rajathagasthya@gmail.com',
    url='https://github.com/rajathagasthya/cuckoopy',
    license='MIT',
    packages=['cuckoopy'],
    install_requires=[],
    classifiers=[
        'Development Status :: 3 - Alpha',
        'Intended Audience :: Developers',
        'Natural Language :: English',
        'License :: OSI Approved :: MIT License',
        'Programming Language :: Python',
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 3.5',
        'Programming Language :: Python :: 3.6'
    ],
)
```