Proceedings of the 53rd Hawaii International Conference on System Sciences | 2020

Starling: A Blockchain-based System for Coordinated Obstacle Mapping in Dynamic Vehicular Environments

Daniel Miehle
Technical University of Munich
daniel.miehle@tum.de

Andreas Pfurtscheller
Technical University of Munich
andreas.pfurtscheller@tum.de

Bernd Bruegge
Technical University of Munich
bruegge@in.tum.de

Abstract

Current Vehicle-to-Vehicle solutions cannot ensure the authenticity of safety-critical vehicle and traffic data. Moreover, they do not allow malicious vehicles to be detected and eliminated. However, this is becoming mandatory, as more and more vehicles are on the road and communicating with each other. We propose a system called Starling, which focuses on trusted coordinated obstacle mapping using blockchain technology and a distributed database. Starling enables vehicles to share detected obstacles with other vehicles in a secure and verifiable manner, thus improving road safety. It ensures that data was not manipulated, changed, or deleted and is based on an open protocol so that vehicles can exchange data regardless of their manufacturer. In a case study, we demonstrate how a consensus is reached among vehicles and conduct a comprehensive evaluation of the Starling system using Ethereum and the InterPlanetary File System.

1. Introduction

The world is becoming increasingly interconnected, and so are vehicles. In recent years, science, industry, and governmental institutions have explored approaches to implementing reliable communication between both, vehicular and non-vehicular systems, known as Vehicle-to-Everything (V2X) communication [1]. V2X communication is paving the way for the internet of vehicles, which allows for the gathering, processing, and exchange of information pertaining to vehicles, infrastructure, and their environment. Thereby, Vehicle-to-Vehicle (V2V) communication is one of the most important types of communication with regard to road safety and efficient traffic flow in general.

Research on this topic dates back to the early 1970s, when Rosen et al. [2] proposed a routing system for Vehicle-to-Road infrastructures. In recent years, interest in this subject has increased, driven in particular by the advances in autonomous driving [1]. Autonomous driving systems are designated to make decisions on the basis of the data available to them. This emphasizes the relevance of this topic for the future of connected and autonomous vehicles. Access to more information about the environment enables autonomous vehicles to take early actions against imminent hazards out of sight [3]. Hence, Vehicle-to-Vehicle networks such as the Vehicle Ad-hoc Network (VANET) aim to enhance the visibility of vehicles in situations that cannot be detected by sensors such as cameras. However, current Vehicle-to-Vehicle networks are unable to detect and eliminate malicious vehicles [4] that have the potential to cause accidents, especially in highly dynamic vehicle environments. In addition, the solutions lack openness, security, and data protection and are highly centralized, so that entire systems can fail in the event of a failure. This is unacceptable for safety critical systems of vehicles as the exchange of vehicle and traffic data between them and other services cannot be maintained.

The Starling system presented in this paper attempts to address these challenges by providing trusted vehicle and traffic data for coordinated obstacle mapping using a distributed, peer-to-peer database and blockchain technology for decentralized and verifiable data storage. The goal of Starling is to build an open Vehicle-to-Vehicle network that offers better visibility of obstacles and makes this visibility more secure and tamper-proof for all involved.

This paper is structured as follows. In Section 2, we present the foundations of obstacle detection and mapping, blockchain technology, and distributed databases on which the Starling system is based and related works. We formalize the requirements and architecture of the Starling system in Section 3. In a case study, in Section 4, we describe the the implementation of the Starling prototype. The Starling prototype is evaluated regarding performance and scalability and the results are discussed in Chapter 5 and 6, respectively. Chapter 7 concludes the paper by summarizing the contributions and future work.

URI: https://hdl.handle.net/10125/64235
978-0-9981331-3-3

2. Foundations and Related Works

In the following section, we describe the foundations of our approach and review related works to embed our approach in the scientific background.

2.1. Obstacle Detection and Mapping

Research in the field of obstacle detection for vehicles was carried out as early as the 1980s and 1990s, long before the advent of autonomous driving. Articles from this period (see [5, 6, 7]) mainly focus on obstacle detection for collision and obstacle avoidance without the exchange of obstacle data between vehicles. With the emergence of high-resolution cameras and enhanced sensor technology such as LIDAR, obstacle detection techniques and algorithms have improved, increasing their accuracy and reliability [8, 9, 10]. Wireless communication makes it possible to share and map information on obstacles detected by individual vehicles so that vehicles cannot only predict the trajectory of moving obstacles [11], but also extend the field of vision of any vehicle [12, 13].

2.2. Distributed Ledger Technology

Distributed Ledger Technology (DLT) extends the concept of distributed, peer-to-peer databases by including features such as data immutability, fair access, transparency, and the verifiability of transactions. Following [14], a distributed ledger is a distributed data structure whose entries are digital records of actions written by the participants of a DLT system after reaching a consensus on the validity of the entries.

One type of DLT is blockchain technology, which stores entries in a linear growing chain of blocks that are secured using cryptography [15]. The data structure of a blockchain is an append-only linked list, which includes a total order of its entities, starting with the so-called Genesis block [16]. Each subsequent block contains a cryptographic hash created using a uniform hash function of the block itself and the previous block, thus linking both blocks. This procedure ensures the immutability of transactions stored in a blockchain, since all subsequent blocks would have to be changed and hashed again to change only one transaction within a block. Verifying hashes is a relatively cheap process, which is why the blockchain can be easily verified by tracing the hashes of each block back to the Genesis block [16]. A transaction can contain any type of data, ranging from cryptographically signed financial transactions, to hashes of digital assets, and Turing-complete executable programs [14].

Blockchain networks can be either permissionless or permissioned. In permissionless blockchains, each participant can initiate transactions, perform mining, and create smart contracts. In contrast, not all participants in permissioned blockchains are allowed to execute all operations. The consensus mechanism ensures, that the nodes of the blockchain network are consistent. It allows the participants of the network to decide on the validity of entries, preventing double-spending and sybil attacks. [17]. Double-spending describes the use of a single asset twice (e.g., one Bitcoin). Sybil attacks address attacks that use fake identities to gain the majority in the system in order to inject faulty information into the network [14]. The consensus mechanism ensures that participants in the consensus process behave honestly and reliably, as it would be more effort in economic terms to do the opposite. [18]. The most common consensus mechanism for public blockchains (e.g., Bitcoin [16], Ethereum [19], is the Proof-of-Work consensus, which utilizes the processing power of computers. Proof-of-Work is the solution to a mathematical puzzle (i.e., mining), which is easy to verify, but solving is both difficult and takes effectively random time. Other consensus mechanisms include Proof-of-Stake (PoS), where evidence is given by providing economic power, and Proof-of-Authority (PoA), where mining is performed by trusted and pre-defined nodes.

2.3. Smart Contract

A smart contract is a computerized transaction protocol that facilitates, executes, and enforces the terms of a contract between untrusted parties without the involvement of a trusted third party [20, 21]. By using business logic implemented in smart contracts, it is possible to access the distributed ledger, the processing power of the system, and its storage [22, 18]. This allows to reduce errors, fraud, and verification time and costs, and to automate process executions [21, 23, 24]. Ethereum has established this concept of distributed computing, which clearly differentiates it from Bitcoin [14, 19, 22].

2.4. InterPlanetary File System

The InterPlanetary File System (IPFS) is a distributed, peer-to-peer file-sharing system that combines a distributed hash-table, an incentivized block exchange, and a self-certifying namespace in order to connect all computing devices with the same system of files [25]. The advantages of IPFS are no single point of failure and nodes do not need to trust each other.

2.5. Related Work

In literature, blockchain-based systems focusing on different aspects of Vehicle-to-Vehicle communication have been proposed.

For instance, Onishi [4] provides a report on the advantages and limitations of Vehicle Ad-hoc Networks and blockchain-based systems for Vehicle-to-Vehicle communication. Rowan et al. [26] propose an inter-vehicle communication by using a blockchain-based public key infrastructure that enhances interoperability between untrusted vehicles, for example, for platooning vehicles. Buzachis et al. [27] introduce an intersection management system to manage negotiated agreement between vehicles crossing through an intersection using blockchain technology and smart contracts. Singh and Kim [28] present an intelligent vehicle data sharing system using a custom consensus mechanism called Proof-of-Driving. Rathore et al. [29] propose the TangleCV, a decentralized solution for secure data sharing and recording for connected vehicles using a directed acyclic graph. Ramachandran et al. [30] introduce MOTIVE, a decentralized framework that allows vehicles to make peer-to-peer micropayments for data, compute and other services obtained from other vehicles or road side infrastructure enabling autonomous operation and trusted interactions between vehicles and nearby entities. Hewlett Packard Enterprise and Continental [31] announced a blockchain-based data monetization platform for sharing vehicle and traffic data. The platform is designed to allow for new digital services that improve security while allowing manufacturers to monetize data. In addition, there are various articles addressing reputation and liability management in vehicular environments (see [32, 33, 34]) and Vehicle-to-Grid applications for peer-to-peer payments (see [35, 36, 37]) based on blockchain technology.

However, the proposed solutions do not investigate the feasibility of blockchain technology and distributed databases for coordinated obstacle mapping.

3. Starling

In this section, we present the idea of establishing a distributed, peer-to-peer system for coordinated obstacle mapping in dynamic vehicular environments.

3.1. Problem

Vehicle-to-Vehicle networks such as VANET aim to enhance the visibility of vehicles in situations that cannot be detected by sensors. It is essential that obstacle mapping data come in the correct order and have not been manipulated or deleted, which could lead to accidents. In addition, current Vehicle-to-Vehicle solutions lack openness, security, and privacy and do not allow malicious vehicles to be detected and eliminated [4]. Only if this is ensured can vehicles improve their systems by using verified obstacle mapping data.

3.2. Vision

The vision for Starling is to provide verified and trusted vehicle and traffic data for coordinated obstacle detection that can be used by vehicles. By using blockchain technology and distributed databases, Starling aims to provide greater visibility of obstacles and make this visibility more secure and tamper-proof for all stakeholders by verifying and matching obstacles, which prevents data redundancy and double entries. Vehicles would know of verified obstacles in their environment without being able to detect them with their own sensors. For example, autonomous driving systems can consider obstacles much earlier in order to avoid accidents and make road traffic more efficient. In addition, Starling introduces the property reputation, which is an indicator of how trustworthy a vehicle is when reporting obstacles and is increased as other vehicles recognize the same obstacles.

3.3. Requirements

In this section, we describe the functional and the non-functional requirements of the Starling system. We start with defining the functional requirements of the Starling system, which aim to describe the interactions between the system and its environment independent of its implementation. The environment includes users and other external systems Starling interacts with.

1. **Provide Obstacle Repository**: The system must enable vehicles to persist mapping data of detected obstacles for coordinated obstacle mapping. This data represents obstacles in the immediate neighborhood of the detecting vehicle.

2. **Avoid Duplication**: The system must ensure that no duplicates of obstacles are persisted in the system, i.e. in case two vehicles detect the same obstacle, there should be a unique entry in the repository.

3. **Detect Faulty Obstacle Data**: The system must be able to detect manipulated or deleted obstacle mapping data. Incorrect data stored in the repository should be detected and not accepted as valid. Moreover, vehicles should

have the property reputation, which indicates the trustworthiness of a vehicle.

4. **Provide Data Access**: The system must provide an interface for clients to retrieve obstacle mapping data from the repository, to retrieve traffic data, or to detect obstacles out of sight.

5. **Provide Authority Access**: The system must ensure that authorities such as police, judiciary, and insurance companies are able to comprehend decisions about vehicles based on the coordinated mapping data.

6. **Provide Traceability**: The system must ensure the complete traceability of all actions in order to reconstruct traffic accidents.

Next, we define the non-functional requirements, which are so-called quality requirements.

7. (*Supportability*) **Open Platform**: The system must be an open platform, which means that the barriers to entry for both, car manufacturers and users must be low. The open platform must create transparency for all stakeholders involved.

8. (*Reliability*) **High Availability**: The system must be available without downtime as it runs in transport environments. A failure is unacceptable.

9. (*Reliability*) **Data Integrity**: The data in the system must be stored so that it cannot be manipulated. For example, traffic incidents must be comprehended when they occur. Therefore, the original data associated with such an incident must be accessible.

10. (*Usability*) **Protect Privacy**: The system must ensure that the tracking of vehicles and their owners should not be possible for anyone other than the vehicle owner and entitled authority actors.

11. (*Performance*) **Low Latency**: The system should be able to store and retrieve obstacles with low latency, as the environment including obstacles changes rapidly due to the high dynamics of traffic. The storing of obstacles should not exceed 500 milliseconds. The verification and retrieving of obstacle data should not exceed 1 second.

12. (*Performance*) **High Throughput**: The system should be able to handle a high data throughput. When scaling the system with many vehicles, the data throughput (i.e., the number of transactions) increases linearly with it.

3.4. Analysis

The analysis object model of the Starling system is derived from the identified requirements as depicted in Figure 1. The main actors of the Starling system are `Vehicles`, the vehicle owners, and authorities such as police officers, which can interact with the system by means of the `VehicleClient` and `AuthorityClient`, respectively. Their abstraction is combined into the `Client` superclass, which provides shared functionality such as retrieving obstacles. The `VehicleClient` allows `Vehicles` to access the `Repository` in order to store and retrieve obstacles. It facilitates the use of the `VehicleIdentifier`, which can be required by the authorities during investigations. The `AuthorityClient` enables authorities to view mapped obstacles and general traffic information.

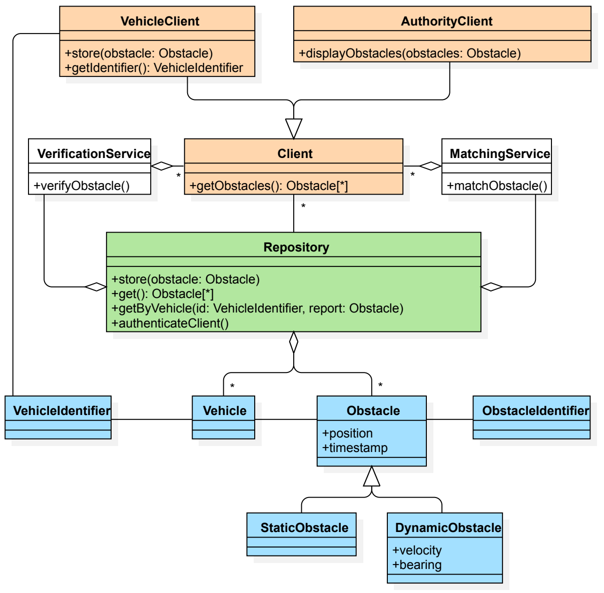

Figure 1. The Analysis object model depicts the classes derived from the requirements. The repository is the key component of the Starling system, containing entity objects like vehicles and obstacle reports. The vehicle and authority clients enable the access to obstacles stored in the repositories.

The core function of the system is to store obstacles in the `Repository`, where the obstacle mapping data is persisted. It allows to read and write obstacles, to register vehicles, and it provides operations such as user authentication.

An `Obstacle` is initially identified and reported by the sensors of a `Vehicle`. An obstacle can be either static or dynamic and consists of a timestamp and a position, which describes the location where it has been detected by a vehicle. In addition, a dynamic obstacle contains information on the velocity and bearing of the physical obstacle represented. Starling assigns the sensor readings to obstacles already reported by other vehicles so that no duplication of physical obstacles occurs. This association is initiated by the `VehicleClient` and performed by the `MatchingService`. The `MatchingService` is called with information about the detected obstacle and reacts with either a matched or a new obstacle. After obstacles are retrieved from the `Repository`, the `VerificationService` verifies them before further processing.

3.5. Architecture

Starling is decomposed into six independent subsystems. These subsystems are grouped into three hierarchical layers, resulting in an open layered architecture. An additional subsystem indicates services provided by the vehicle's autonomous driving system. We explain the individual layers from bottom to top.

The **Verification Layer** provides `ObstacleVerification` services to the upper layers and does not depend on any other layer. The subsystem `VerificationStore` is derived from the `VehicleRepository`. Since the `Verification Store` does not contain the `ObstacleRepository`, we preserve the relationship between the classes `Vehicle` and `Obstacle` from the analysis object model. Therefore, we introduce the `HashRepository` to our subsystem, which contains permanent links to the `Obstacles` stored in the `ObstacleRepository` subsystem in the layer above.

The **Obstacle Layer** provides services for storing and retrieving obstacle data as well as matching obstacles. It encapsulates the `Obstacle Store` that contains the `ObstacleRepository` with its obstacle mapping data and the `ObstacleMatching` subsystem accessible using the `Obstacle API`. They use the services of the Verification layer below to verify the obstacle data they are working with.

The **Client Layer** contains the client applications `Vehicle Node` and `Authority Node`, which provide system access for the actors `Vehicle` and `Authority`, respectively. They use services provided by subsystems from both, the Obstacle and the Verification Layer.

4. Case Study

Here, we describe the implementation of the Starling prototype using Ethereum and IPFS and present a detailed analysis and discussion of the prototype.

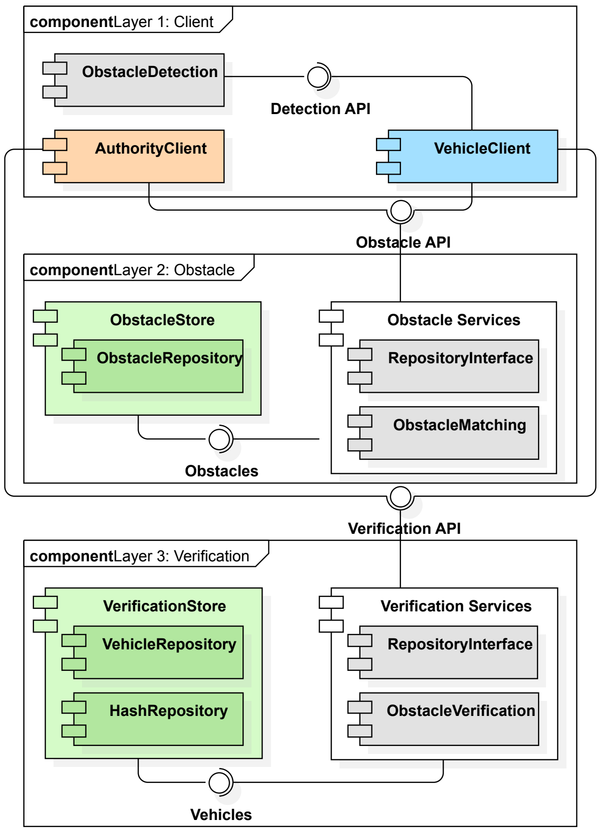

Figure 2. The subsystem decomposition model shows the six subsystems, which are organized into three layers within an open layered architecture.

4.1. Objective

The aim is to prove the feasibility of the decentralized Starling system for coordinated obstacle mapping. We implemented a prototype to gain further insight into the advantages and limitations of combining blockchain technology and distributed databases.

4.2. Hardware / Software Mapping

In the following, the decomposed subsystems are mapped to commercially available software and hardware components with which we implemented the prototype. Considering the requirements and constraints, we obtain the hardware-software mapping as shown in Figure 3.

First, we explain the components that enable our decentralized data storage, which is based on two separate components. For the **distributed databases**, we use the high-throughput, serverless, distributed, and queryable database OrbitDB[1]. OrbitDB is based on the peer-to-peer protocol IPFS [2] designed to create a distributed database. To achieve consistency across all nodes, OrbitDB uses an immutable, operation-based conflict-free replicated data structure (CRDT) [38], which was proposed by Shapiro et al. [39] in the context of the increasing emergence of distributed systems. CRDTs include, among others, an append-only log that can be used to model a mutable, shared state between peers in peer-to-peer applications meeting the requirements for documentation quality, flexibility, and throughput at the time of system design. We use the official IPFS client implementation, which is written in Go[3] providing the IPFS HTTP API[4] as an interface that allows clients to interact with it and the stored data.

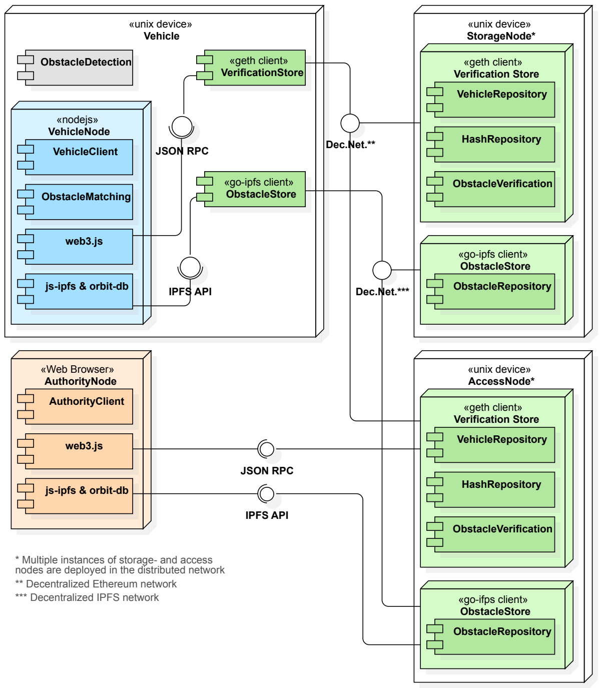

Figure 3. The deployment diagram maps the decomposed subsystems to commercially available software and hardware components.

In order to build the **blockchain** test network, we decided to use Ethereum with the Proof-of-Work consensus mechanism. Ethereum is a general purpose permissionless blockchain, which provides detailed documentation, a high degree of decentralization, and enables the use of smart contacts. To connect to the Ethereum test network, one has the choice between the official Ethereum client named Geth [5] and the independently developed Parity client[6]. We use Geth in Starling, as it is the reference implementation of Ethereum that provides client access via the standardized JSON RPC API[7].

4.2.1. StorageNode and AccessNode The core component of Starling is the distributed data storage in which detected obstacles are mapped and stored. The distributed data storage is separated into two independent components, the distributed file system IPFS, and the Ethereum blockchain.

4.2.2. VehicleNode The VehicleNode provides the system with the actual obstacle mapping data. For the case study, a Node.js[8] environment was chosen, in which the VehicleNode is executed. It connects the ObstacleMapping service with the decentralized ObstacleStore. This component expects an interface provided by the ObstacleDetection on-board system that allows it to receive notifications of detected obstacles. The ObstacleDetection on-board system is not part of the Starling system and is therefore simulated. The subsystem ObstacleMatching compares the detected obstacles with those already stored in Starling. These obstacles are then stored in IPFS using the OrbitDB Javascript library, which connects to the local IPFS Store via the IPFS Javascript library[9]. Once an obstacle data set is stored in IPFS, a unique hash value of this data set is generated and recorded on the Ethereum blockchain. The hash value serves as the immutable digital identity of the obstacle data and can be verified at any time. The hash value is stored via the local GethClient with web3.js[10], which is a collection of Javascript libraries that allow you to interact with a local or remote Ethereum node over an HTTP or IPC socket connection. VehicleNodes can retrieve mapped obstacles via the OrbitDB Javascript library. Based on this information, the autonomous driving systems can, for example, make decisions such as initiating a safety brake. The ObstacleVerification subsystem is implemented by a smart contract running on Ethereum, which contains the business logic for obstacle verification. Once the verification process is

[1]https://github.com/orbitdb/orbit-db
[2]https://ipfs.io/
[3]https://github.com/ipfs/go-ipfs
[4]https://docs.ipfs.io/reference/api/http/
[5]https://github.com/ethereum/go-ethereum
[6]https://github.com/paritytech/parity-ethereum
[7]https://github.com/ethereum/wiki/wiki/JSON-RPC
[8]https://nodejs.org/
[9]https://github.com/ipfs/js-ipfs
[10]https://github.com/ethereum/web3.js/

completed, the VerificationStore service in the ObstacleVerification subsystem can be used to perform verification by call.

4.2.3. AuthorityNode The AuthorityNode provides authority actors access to the obstacle mapping data by means of a graphical user interface. It is a Javascript application that runs in web browsers with the same libraries that are used for the VehicleNode. To access the mapped obstacle data, it connects to the decentralized database over a public interface exposed by a remote ObstacleRepository, which is represented by the AccessNode.

4.2.4. Network All subsystems and devices of the Starling system can be connected using an arbitrary network. This can be a private network deployed in a local environment or a public one like the internet.

5. Evaluation

We implemented viable subsystems in order to evaluate the proposed Starling system.

5.1. Simulation

As the operation of Starling in a real scenario was not possible, we simulated vehicles that reliably detected obstacles on a selectable route. For the simulation, we developed further subsystems (e.g., simulation manager) that communicate with each other and were integrated into the Starling system. The simulation manager has access to all simulated vehicles as well as their position, speed, and storage data, but is not aware of the extended environment of the vehicles. The data provided by the simulation manager represent the obstacle data that a real vehicle would receive from its sensors. Hence, the simulation manager can be considered as the ObstacleDetection subsystem.

The graphical web interface depicted in Figure 4, shows the simulation using a map. Using this interface, any information such as traffic conditions can be displayed and the status of a specific vehicle can be reproduced at a certain point in time.

5.2. Design

For the evaluation, ten equivalent virtual machines were provisioned, each with 8GB RAM, 4 CPU cores and 100GB hard disk space running Ubuntu 16.04 LTS as the operating system. The machines operated in a data center of a cloud hosting provider and were connected

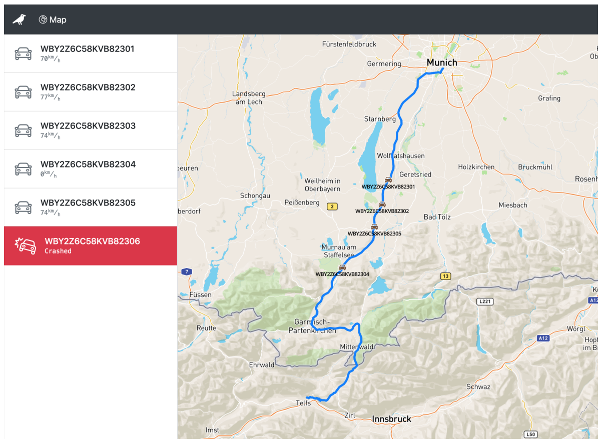

Figure 4. Starling user interface showing the simulation manager and the simulated vehicles.

with each other through a local network. The Docker Engine was installed on all machines, enabling them to be combined into a cluster using Docker's swarm mode. By running the simulation, the performance and scalability of combining the Ethereum blockchain and the decentralized, peer-to-peer data storage IPFS will be quantitatively evaluated for the use of coordinated obstacle mapping. Based on this data, it is verified whether the requirements are fulfilled and if these technologies are ready to be used for the purpose of coordinated obstacle mapping.

5.3. Results

We have performed three simulation runs with different objectives. In the **first simulation run** with six vehicles, the verification process – using the hash values stored in the blockchain – was disabled when matching the detected obstacle with the already mapped ones. The detected obstacles could be matched to the obstacles returned by a query on OrbitDB. In addition, towards the end of the simulation, when all six vehicles were stuck in traffic jams, the exact number of six obstacles were mapped, with the obstacles detected by the vehicles always matching the stored ones. Observing the state of the blockchain using the Ethereum Network Status interface, an average of 13 transactions per block are shown, which is approximately 1.6 transactions per second with an average block time of 8 seconds.

In the **second simulation run**, we continued with six vehicles over a 24-hour period to measure the convergence behavior of the average block time, which settled at 15.6 seconds, while 98% of the blocks propagated in under one second. Another relevant time interval is the average delay that occurs when OrbitDB replicates data between its distributed nodes. Before measuring this value, the internal clocks of the nodes connected to OrbitDB were synchronized with the same NTP time server, so that the replication delay could be calculated without deviations due to inaccurate clocks. Beyond that, the number of nodes involved was limited to two, so that the actual replication time between these nodes could be calculated. The simulation was started and the timestamps at the time of saving on one node and replication on the other node were extracted. This data set was used to calculate the difference between the two timestamps and their mean value, which is 197 milliseconds.

Finally, we tested the system regarding its scalability in the **third simulation run**. For this purpose, the number of vehicle nodes was increased to 20 while the reaction of the system to this change was observed. It could be seen that with a growing number of vehicles the number of Ethereum and OrbitDB transactions increases. While the vehicles were scattered across the route to the accident site, the blockchain processed an average of 26 transactions per block, which is approximately 1.67 transactions per second with an average block time of 15.6 seconds. This value increases to up to 7 transactions per second as more vehicles get into traffic jams. During this period, however, the replication time of OrbitDB remained constant at about 200 ms.

6. Discussion

The Starling system is a successful proof of concept and is capable of performing coordinated obstacle mapping using Ethereum and IPFS in a reliable way. However, due to the limiting properties of the **Ethereum** virtual machine (e.g., support of floating point numbers), the matching algorithm had to be implemented in the vehicle node running in the distributed database instead in the smart contract. The results of our evaluation show that the average block times using the **Proof-of-Work** consensus mechanism are higher (approx. 15 seconds) than the needed near-real time requirement for obstacle mapping (1 second). This means that the obstacles queried from the distributed database are not validated, which is why incorrect obstacle data could not be identified. However, as soon as transactions are added to the blockchain, obstacle reports can be traced and verified in retrospect.

Concerning **data privacy**, Ethereum provides pseudo-anonymity in the form of addresses. However, once the relation between an address and a vehicle is uncovered, vehicle data can be associated to the vehicle owner. Therefore, vehicles need to generate a one-time address per-trip or per-event while persisting their used addresses locally. In case a one-time address is needed, an authorized actor such as the vehicle owner can access these addresses stored by the vehicle.

Since deleting data from the blockchain is not possible by design, a further challenge is the **increasing demand and costs** for data storage, especially for resource-limited vehicle nodes. As the majority of data stored in vehicles will be obsolete in terms of time or irrelevant based on their position, we propose introducing so-called light nodes. Light nodes do not download or verify the entire chain of blocks, instead they rely on full nodes for sending transactions and querying obstacle data. However, this could not be realized in the course of our simulation, because the still experimental light nodes of Ethereum were unable to discover any full nodes, and OrbitDB was not yet capable of this functionality. To tackle this issue we trade decentralization for storage, which shifts the control of the network towards the operators of full nodes. This compromise could be accepted in the Starling network, as full node operators will be a diverse group of stakeholders including vehicle manufacturers, governments, and insurance companies.

While blockchain technology can guarantee that the obstacle mapping data is not tampered with, it does not guarantee that the data recorded by the sensors is accurate. Additional technical controls such as secure sensor elements with unique private keys to ensure increased data integrity may be required.

The second component that was evaluated was the distributed data storage **IPFS**, which met our requirements for storing obstacle data, even replicating in half the time stated in the quality requirements. In the future, the obstacle matching should be executed by the blockchain and not by the vehicle, as it is implemented in the current version of the Starling prototype.

7. Conclusion

In this paper, we successfully designed and developed the Starling system, which allows vehicles to map detected obstacles in a coordinated manner by means of distributed ledger technology. Furthermore, we implemented a reliable algorithm for matching detected obstacles with already mapped ones, so that the validity of the latter can be verified, duplicated reports prevented, and their traceability ensured. The use of a distributed ledger for the coordinated mapping of obstacles in the context of road traffic embodies a promising technology, especially when considering the immutability and traceability of the data stored. Decisions made by autonomous vehicles based on the sensed surroundings can be comprehended in retrospect.

We addressed the functional and non-functional requirements described in Section 3.2.: The requirements of (1) Provide Obstacle Repository, (2) Avoid Duplication, (3) Detect Faulty Obstacle Data, (4) Provide Data Access, (5) Provide Authority Access and (6) Provide Traceability were addressed by introducing IPFS and our User Interface. (7) Open Platform, (8) High Availability, and (9) Data Integrity were archived using a public blockchain protocol based on proof-of-work (i.e., Ethereum) allowing everyone to connect to Starling. Starling is distributed among several nodes fulfilling high availability in a way that hardly any other ordinary database can. Finally, (10) Protect Privacy, (11) Low Latency, and (12) High Throughput could be partially fulfilled.

In order to ensure data privacy in the future, vehicles need to generate a one-time address while persisting its used addresses locally. In addition, as Starling has to deal with several thousand transactions per second, future work should investigate distributed ledger solutions focusing on scalability (e.g., Tendermint, IOTA, and Hedera Hashgraph), second layer solutions (e.g., Raiden and Lightning Network), and the use of light nodes for resource-limited vehicle nodes. Furthermore, it should be investigated how manufacturers and vehicle owners can monetize their data to create new revenue streams.

References

[1] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected Vehicles: Solutions and Challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, 2014.

[2] D. A. Rosen, F. J. Mammano, and R. Favout, "An Electronic Route-Guidance System for Highway Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 19, no. 1, pp. 143–152, 1970.

[3] H. Hartenstein and K. Laberteaux, *VANET: Vehicular Applications and Inter-Networking Technologies*, vol. 1. John Wiley & Sons, 2009.

[4] H. Onishi, "A Survey: Engineering Challenges to Implement VANET Security," *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–6, 2018.

[5] H. Endo, "Obstacle Detection System for Use in Vehicles," 1984. US Patent 4,477,184.

[6] K. Storjohann, T. Zielke, H. A. Mallot, and W. von Seelen, "Visual Obstacle Detection for Automatically Guided Vehicles," in *IEEE International Conference on Robotics and Automation*, pp. 761–766 vol.2, 1990.

[7] Y. Asayama, "Obstacle Detecting Device for a Vehicle," 1995. US Patent 5,386,285.

[8] R. Labayrade, D. Aubert, and J.-P. Tarel, "Real Time Obstacle Detection in Stereovision on non Flat Road Geometry Through "v-disparity" Representation," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, pp. 646–651, 2002.

[9] W. Fuzhong, L. Haibo, and Y. Fashan, "Obstacle Avoiding Strategy of Mobile Robot via Binocular Stereovision," in *27th Chinese Control Conference*, pp. 457–461, IEEE, 2008.

[10] J. Han, D. Kim, M. Lee, and M. Sunwoo, "Enhanced Road Boundary and Obstacle Detection Using a Downward-Looking LIDAR Sensor," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 3, pp. 971–985, 2012.

[11] L. C. McNinch, R. A. Soltan, K. R. Muske, H. Ashrafiuon, and J. C. Peyton Jones, "Application of a Coordinated Trajectory Planning and Real-Time Obstacle Avoidance Algorithm," in *Proceedings of the 2010 American Control Conference*, pp. 3824–3829, 2010.

[12] S. Fujii, A. Fujita, T. Umedu, S. Kaneda, H. Yamaguchi, T. Higashino, and M. Takai, "Cooperative Vehicle Positioning via V2V Communications and Onboard Sensors," in *IEEE Vehicular Technology Conference (VTC Fall)*, pp. 1–5, 2011.

[13] J. Ward, S. Worrall, G. Agamennoni, and E. Nebot, "The Warrigal Dataset: Multi-Vehicle Trajectories and V2V Communications," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 3, pp. 109–117, 2014.

[14] M. C. Ballandies, M. M. Dapp, and E. Pournaras, "Decrypting Distributed Ledger Design - Taxonomy, Classification and Blockchain Community Evaluation," 2018.

[15] M. Swan, *Blockchain: Blueprint for a new Economy*. O'Reilly Media, Inc., 2015.

[16] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[17] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, and R. Brooks, "A Brief Survey of Cryptocurrency Systems," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pp. 745–752, IEEE, 2016.

[18] F. M. Benčić and I. Podnar Žarko, "Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph," in *IEEE 38th*

International Conference on Distributed Computing Systems (ICDCS), pp. 1569–1570, 2018.

[19] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[20] N. Szabo, "Formalizing and Securing Relationships on Public Networks," *First Monday*, 1997.

[21] M. Alharby and A. van Moorsel, "Blockchain Based Smart Contracts: A Systematic Mapping Study," *Computer Science & Information Technology (CS & IT)*, 2017.

[22] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A Taxonomy of Blockchain-Based Systems for Architecture Design," in *IEEE International Conference on Software Architecture (ICSA)*, IEEE, 2017.

[23] L. W. Cong, Z. He, and J. Zheng, "Blockchain Disruption and Smart Contracts," *SSRN Electronic Journal*, 2017.

[24] M. Kõlvart, M. Poola, and A. Rull, "Smart Contracts," *The Future of Law and eTechnologies*, 2016.

[25] J. Benet, "IPFS-Content Addressed, Versioned, P2P File System," *arXiv*, 2014.

[26] S. Rowan, M. Clear, M. Gerla, M. Huggard, and C. M. Goldrick, "Securing Vehicle to Vehicle Communications Using Blockchain Through Visible Light and Acoustic Side-Channels," *arXiv*, 2017.

[27] A. Buzachis, A. Celesti, A. Galletta, M. Fazio, and M. Villari, "A Secure and Dependable Multi-Agent Autonomous Intersection Management (MA-AIM) System Leveraging Blockchain Facilities," *IEEE/ACM International Conference on Utility and Cloud Computing Companion*, pp. 226–231, 2018.

[28] M. Singh and S. Kim, "Blockchain Based Intelligent Vehicle Data sharing Framework," *arXiv*.

[29] Rathore, Heena and Samant, Abhay and Jadliwala, Murtuza and Mohamed, Amr, "TangleCV: Decentralized Technique for Secure Message Sharing in Connected Vehicles," *Proceedings of the ACM Workshop on Automotive Cybersecurity*, pp. 45–48, 2019.

[30] G. S. Ramachandran, X. Ji, P. Navaney, L. Zheng, M. Martinez, and B. Krishnamachari, "Micropayments for Trusted Vehicular Services Using MOTIVE," *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 701–702, 2019.

[31] C. Hewlett Packard Enterprise Company, "Hewlett Packard Enterprise and Continental launch blockchain-based data monetization platform," 2019.

[32] Z. Yang, K. Zheng, K. Yang, and V. C. M. Leung, "A Blockchain-Based Reputation System for Data Credibility Assessment in Vehicular Networks," in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–5, 2017.

[33] Z. Lu, W. Liu, Q. Wang, G. Qu, and Z. Liu, "A Privacy-Preserving Trust Model Based on Blockchain for VANETs," *IEEE Access*, vol. 6, pp. 45655–45664, 2018.

[34] C. Oham, S. S. Kanhere, R. Jurdak, and S. Jha, "A Blockchain Based Liability Attribution Framework for Autonomous Vehicles," *arXiv*, 2018.

[35] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling Localized Peer-to-Peer Electricity Trading Among Plug-in Hybrid Electric Vehicles Using Consortium Blockchains," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3154–3164, 2017.

[36] F. Gao, L. Zhu, M. Shen, K. Sharif, Z. Wan, and K. Ren, "A Blockchain-Based Privacy-Preserving Payment Mechanism for Vehicle-to-Grid Networks," *IEEE Network*, vol. 32, pp. 184–192, 2018.

[37] Z. Zhou, L. Tan, and G. Xu, "Blockchain and Edge Computing Based Vehicle-to-Grid Energy Trading in Energy Internet," *IEEE 2nd Conference on Energy Internet and Energy System Integration (EI2)*, pp. 1–5, 2018.

[38] OrbitDB Development Team, "OrbitDB: Peer-to-Peer Databases for the Decentralized Web," 2019.

[39] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "Conflict-Free Replicated Data Types," in *Symposium on Self-Stabilizing Systems*, pp. 386–400, Springer, 2011.