Future Generation Computer Systems 149 (2023) 606–621

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

TrustedMaaS: Transforming trust and transparency Mobility-as-a-Service with blockchain

Tri Nguyen *, Huong Nguyen, Juha Partala, Susanna Pirttikangas

Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland

ARTICLE INFO

Article history:
Received 3 March 2023
Received in revised form 7 August 2023
Accepted 9 August 2023
Available online 12 August 2023

Keywords:
Mobility-as-a-Service
Blockchain technology
Smart contract
Hyperledger Fabric
Ethereum
Blockchain-based MaaS

ABSTRACT

Mobility-as-a-Service (MaaS) is an advanced Intelligent Transport System (ITS) that integrates various modes of transportation to meet the demands of travellers. The system relies on frequent communication for data exchange between the MaaS provider and transport service providers. Ideally, such communication would utilize trust technologies between these entities. However, current MaaS systems lack transparency and reliability, and their centralized nature creates a single point of failure for the entire service. To address these issues, this paper proposes a blockchain-based MaaS, which includes an architecture and smart contract functionalities. The solutions are built on permissioned and permissionless Hyperledger Fabric and Ethereum blockchain platforms, respectively, for a realistic deployment of network architecture and proposed smart contracts. Additionally, the paper presents a framework derived from comparing these two blockchain platforms. Finally, the framework is evaluated, and open questions and challenges are analysed.

1. Introduction

According to the Intergovernmental Panel on Climate Change,[1] the transport sector could have the largest impact on emissions without urgent action. To tackle this issue, both research and industrial communities have undertaken numerous studies [1–3] intending to address transportation-related challenges. These include optimizing traffic efficiency and minimizing transport problems, such as traffic congestion and parking difficulties, by introducing innovative services under Intelligent Transport Systems (ITS). One such service, Mobility-as-a-Service (MaaS), is an exciting solution that integrates various modes of mobility to meet the demands of travellers. By connecting different means of transport and enabling coordinated and intelligent services, MaaS represents a significant advancement in ITS.

MaaS is an Information and Communication Technology (ICT) solution that prioritizes user convenience by facilitating smooth journey planning, ticketing, and travelling from source to destination through advanced technologies. MaaS provides a single ICT platform that enables cooperation between various parties to offer services, such as integrating multiple transport modes, diverse tariff options, customization, personalization, and demand orientation, as characterized in [4]. These convenient services encourage travellers to use public transport, leading to a reduction in carbon footprint while maximizing customization and personalization, as highlighted in [5]. However, for MaaS to grow, transport service providers must collaborate to ensure trust in the system's parties, boosting the MaaS scale's increase. A study by Polydoropoulou et al. [6] notes that successful deployment and value creation of MaaS requires cooperation between the public and private sectors, particularly in e-ticketing through trust among MaaS actors. Additionally, the lack of open data, Application Programming Interface (API), and standard ways of operation constitute barriers to the growth of MaaS.

Given the many compelling features of blockchain technology, a blockchain-based MaaS system is an advisable means of supporting the rise of the transportation company network. Blockchain technology, as the next generation of connectivity, is a unique public database maintained by network participants in a decentralized environment. A blockchain-based system emits most data to all participants for validation, resulting in a transparent set of data. Additionally, the formation of a unique database requires immutability to prevent any late modification by leveraging cryptographic schemes. As a promising perspective for trust formation, a decentralized MaaS system can be deployed by utilizing blockchain technology to achieve and guarantee trust among participants. In other words, the traditional MaaS, as seen in Fig. 1(a), requires transport service providers to trust the MaaS service provider, who designs a set of rules and contracts for collaboration. In contrast, the next generation of MaaS, as

* Corresponding author.
E-mail addresses: tri.nguyen@oulu.fi (T. Nguyen), huong.nguyen@oulu.fi (H. Nguyen), juha.partala@oulu.fi (J. Partala), susanna.pirttikangas@oulu.fi (S. Pirttikangas).

[1] https://www.ipcc.ch/

https://doi.org/10.1016/j.future.2023.08.011

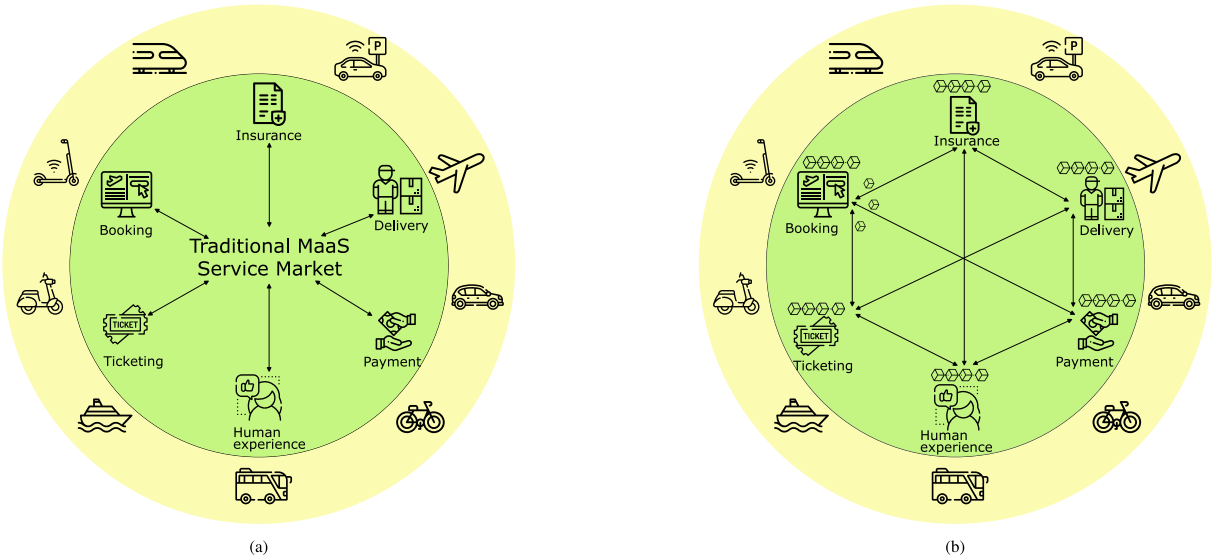

Fig. 1. The depicted models represent two distinct MaaS systems: (a) a traditional MaaS system and (b) a blockchain-based MaaS system. In each model: The outermost layer (highlighted in yellow) represents service providers within the transportation sector, such as rental car agencies or bus services. The intermediate layer (coloured in green) delineates the potential services available for transportation. The central component, denoted by arrows, symbolizes the interactive connections between the transportation services.

shown in Fig. 1(b), aims to construct trust among transport service providers by sharing information across the network to form a unique database that supports services such as payment, ticketing, booking, and enhancing the human experience.

As per Nguyen et al.'s research [7], the blockchain-based MaaS system consists of two sub-systems or protocols. The first protocol facilitates the sharing of transportation data among service providers through a standardized data format and open API. This protocol aims to provide transportation data for the trip planning process. The second protocol involves a blockchain-based ticketing system that stores and secures ticket-related services from potential adversaries for ticket verification and confirmation. This protocol is the primary contribution of the present study as below:

1. **Advanced Deployment in Real-World Scenarios:** Building upon our pioneering work in blockchain-based MaaS [7], this paper presents a refined and practically-deployable architecture, demonstrating its applicability through the development of intricate smart contracts, both on the widely-adopted platforms of Hyperledger Fabric[2] and Ethereum.
2. **Pioneering Comparative Framework:** In the rapidly evolving landscape of blockchain, clarity on platform choices can be a quagmire. Our work introduces a meticulously crafted framework, serving as a touchstone for the direct technical comparison between two of the blockchain behemoths: Hyperledger Fabric and Ethereum. This not only aids developers in making informed decisions but also paves the way for future research on blockchain platform evaluations.
3. **Deep Dive into Future Directions:** Addressing the current gap in literature, our study goes beyond mere architectural presentations and dives deep into the intricate challenges, uncharted questions, and presents novel solutions that can shape the future trajectory of blockchain-based MaaS development.

The paper is structured as follows: Section 2 reviews the relevant literature. In Section 3, we briefly introduce the technologies that will be used in our proposal. Section 4 presents our proposal, which includes the design of smart contracts and a blockchain-based MaaS system. The experiments and results are described in Section 5. In Section 6, we compare two blockchain platforms using a framework for technological comparison and provide further discussion. Finally, Section 8 concludes the paper.

2. Related work

2.1. Leveraging blockchain to transportation systems

One of the earliest studies to mention blockchain technology in ITS is by Yuan et al. [8]. The authors propose a blockchain-based ITS called B^{2}ITS that offers security, trust, and decentralization features. In particular, the study highlights how blockchain can enable a decentralized autonomous system in transportation [9,10] by enabling diverse uses, such as decentralized applications and autonomous organizations [11]. The authors suggest that using blockchain can help foster trust in a decentralized environment through smart contracts, making it a promising approach to boosting the growth of ITS. They also note that using blockchain in ITS is the first step in creating parallel transportation management systems.

In a study by Li et al. [12], the authors discuss two critical issues for reliable announcements in a vehicular announcement network: privacy preservation and motivation for announcement forwarding. They propose a blockchain-based solution called CreditCoin, a privacy-preserving incentive announcement network that encourages participants to share information. CreditCoin uses a Ripple consensus [13] maintained by roadside units and official public vehicles to create a reputation system where each user has reputation points. These points are used to evaluate received announcements and gain more reputation points by sharing announcements.

Another promising use of blockchain in MaaS is highlighted in a study by Karinsalo et al. [14], which emphasizes the importance of smart contracts. The authors propose a business design that uses artificial intelligence and blockchain to create a MaaS ecosystem with smart contracts that can establish trust in identity, automate MaaS functions, use digital currency for convenient trade, and provide reliable audits among different

[2] Hyperledger Fabric and Fabric, in short, can be used interchangeably with the same meaning

parties. They describe the deployment of smart contracts with QR tokens that travellers can use for travelling. Although the study mentions the benefits that blockchain can bring to MaaS, it does not provide a detailed description of the duties and components of a blockchain.

A discussion of the role of blockchain in a MaaS ecosystem is provided in a study by Bothos et al. [15], which suggests using blockchain-based smart contract platforms to handle MaaS operators and reduce transaction costs. The authors emphasize the usefulness of digital tokens for micropayments and the reduction of transaction fees, particularly with business logic automation. Additionally, blockchain can provide a solution for storing information related to payment, ticketing, and booking through unique identifiers, which can help increase the openness and transparency of a MaaS ecosystem.

While much of the existing research emphasizes the benefits of employing blockchain to develop decentralized MaaS systems, there remains a significant oversight of a holistic system consideration. For instance, studies such as [12,14,15] utilize blockchain as a subsystem, focusing on aspects like reputation systems and micropayments facilitated through financial benefits. This approach is predictable, considering the established use of blockchain in the financial sector. In contrast, our previous work [7] envisions a comprehensive design for MaaS leveraging blockchain, placing particular emphasis on the growth and development of blockchain-based MaaS. Here, we propose a blockchain-based MaaS design within a decentralized framework, demonstrating that blockchain serves not merely as a subsystem but is, indeed, the core technology facilitating the decentralization of traditional MaaS.

Building on our previous work on a design for blockchain-based MaaS [7], we propose a proof-of-concept extension of the blockchain-based MaaS in this paper. Our approach is to create a decentralized network of transportation service providers who can exchange their service information across the network and generate tickets based on travellers' requests, thus providing trust in operations such as routing and ticketing without intermediaries as MaaS providers. By deploying blockchain-based MaaS, we explore the existing challenges and open questions. In particular, we examine the use of two types of blockchains: permissioned and permissionless.

2.2. Blockchain performance

Blockchain-based use cases present unique performance considerations when compared to traditional systems, which primarily rely on potent computation and communication capabilities. In contrast, the performance of blockchain systems is influenced by additional parameters such as participant interaction and augmented communication requirements. Blockchain technology, in essence, opts for consistency at the cost of performance by facilitating additional communication among participants. Moreover, the aggregate computational power of network participants does not emulate the cohesive computational power within a single organization, leading to a different resource orchestration approach.

Considering this perspective, Thakkar et al. [16] present a comprehensive comparison of various Hyperledger Fabric configurations, including block size, endorsement policy, channels, and resource allocation. Their research underlines the interdependencies among block sizes, throughput, and latency, as well as the effect of different policies on these parameters. Another significant finding they report is the direct relationship between CPU computation and throughput as the number of channels increases. Besides, [17] implements similar experiments with Hyperledger Fabric regarding performance in storage, communication, and process, but in various transaction types: query and update. As another analysis for Hyperledger Fabric performance, [18] points to the gap between analytical results and actual results with various block sizes, submitted transactions, organizations, channels, and block intervals.

The diversity of blockchain platforms opens a variety of research questions related to performance correlations. An early comparative study by Pongnumkul et al. [19] demonstrates superior throughput and lower latency of Fabric when compared to Ethereum. Dinh et al. [20] introduced BLOCKBENCH, a benchmark framework for evaluating blockchain platforms, further affirming Fabric's superior throughput and lower resource consumption compared to Ethereum variants. However, both studies lack comprehensive details on block and transaction capacity settings for each blockchain platform, especially difficulty configuration in Proof-of-Work-based Ethereum. Additionally, the interplay among smart contracts, which is a vital component in autonomous applications, is not extensively addressed.

3. Technologies

3.1. Mobility-as-a-Service

With the widespread adoption of MaaS, numerous studies have explored its various aspects [21]. MaaS is a critical concept in mobility management, providing travellers with flexible transport modes and technology-enabled services [22,23]. As a theoretical analysis for MaaS design and implementation, Hensher et al. [24] identify three fundamentals: budgets, brokers, and bundles. Budgets reflect the payment preferences of travellers, brokers aggregate independent transport suppliers, and bundles consider possible mobility packages that travellers desire to purchase. According to Mulley [22], MaaS transforms from owned physical properties, such as automobiles, to services in a MaaS ecosystem.

In short, MaaS is a transport platform consolidating a set of transport services to provide convenient transport solutions and a suitable package of mobility services through integrating independent transport services [25]. MaaS leverages advanced technologies to connect numerous transport providers and offers an array of transport services, including booking, paying, and planning for travel, as illustrated in Fig. 1(a). By leveraging advanced technology, MaaS aims to provide benefits for travellers and the environment, including reducing the carbon footprint by reducing the number of individual automobiles. MaaS satisfies mobility demands by improving the travel experience and provides travellers with personalized transport solutions and tariff options.

3.2. Blockchain technology

Blockchain is a well-known technology that enables the creation of decentralized systems. It was first introduced through Bitcoin [26], a cryptocurrency that operates in a decentralized environment. At its core, blockchain is a public database distributed among system participants. Immutability is a crucial feature of this technology to prevent faulty participants from violating the protocol in a decentralized environment. Bitcoin addresses this by proposing a unique chain of data blocks, each containing a chronological set of transactions and a hash value of the previous block as a hash pointer scheme. Additionally, every block stores the root of a Merkle tree structure [27], formed by the set of transactions, as shown in Fig. 2.

The success of blockchain technology owes much to the assistance of consensus mechanisms. These mechanisms determine the next state of the system due to the absence of intermediaries. As previously mentioned, a blockchain system consists

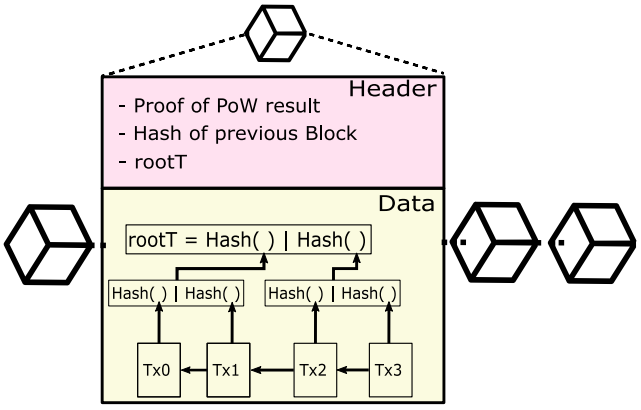

Fig. 2. The structure of Bitcoin's blockchain consists of a header and data. The header contains main information, such as the value of the PoW result, the Merkle tree value, and the previous block's hash value; meanwhile, the data part stores a chronological order of transactions.

of a chain of data blocks; thus, each confirmed block signifies the participants' approval of the subsequent system state. The primary task of consensus is to secure the majority agreement among network participants to ensure that everyone shares the same system view.

Blockchain-based systems can be divided into two main categories based on participant permissions: permissionless and permissioned. Permissionless blockchains allow participants to join and leave the system freely without any identity requirements, while permissioned blockchains only permit identified participants to join and manage the blockchain. Since permissionless blockchains do not require participant identity, they often use a consensus algorithm like Proof-of-X (PoX) proposed by Tschorsch et al. [28] to establish trust between participants based on proof-of-resources. On the other hand, permissioned blockchains rely on a set of identified participants with reputations, providing a security feature in communication without the need for proof of correctness in proposals.

Blockchain technology's success has led to creation of a blockchain-based smart contract platform. A smart contract is a programmable code declared and defined by multiple parties for an agreement, similar to a traditional contract on paper. The advent of blockchain technology has made smart contracts even more attractive. Ethereum [11,29] introduced a blockchain-based smart contract in 2014, which has seen extensive use in non-fungible tokens and decentralized finance. Ethereum also introduced the concept of decentralized applications and decentralized autonomous organizations through smart contracts. Decentralized applications offer an array of services as back-end programs based on arbitrary programming languages, which can operate independently or jointly to create an information ecosystem. Similarly, decentralized autonomous organizations are self-organizing and can become self-sustaining networks in a democratized environment, with participants having permission to be part of the committee based on pre-defined and immutable smart contracts.

3.2.1. Ethereum: Permissionless platform

Ethereum [11,29] was the first to introduce a blockchain-based smart contract platform with an order-execute workflow. This design involves two phases: ordering and executing. Instead of executing each operation independently, the system's participants agree on a sequence of operations or transactions before executing them to ensure a total order. Ethereum is a permissionless platform, meaning participants can join and leave the system, leading to a high churn rate. As with other permissionless blockchains, trust among participants is crucial, and a consensus mechanism such as Proof-of-Work (PoW), as used in Bitcoin [26], is required to maintain the integrity of the system. Unlike blockchain-based cryptocurrency, a blockchain-based smart contract infrastructure requires a machine for operational execution and a specific programming language for smart contract definition. The traditional blockchain focuses on avoiding conflict transactions in the system, meaning new transactions cannot conflict with confirmed transactions. In contrast, a blockchain-based smart contract requires an environment for execution. In Ethereum, this environment is provided by the Ethereum Virtual Machine (EVM), a stack machine that can handle 256-bit cryptography schemes.[3] Solidity[4] is the programming language used to define smart contracts in Ethereum. These smart contracts are compiled into EVM executable bytecode before they are deployed to the blockchain.

Proof of Work (PoW) is a promising consensus solution for decentralized networks where trust among participants cannot be guaranteed. Ethereum utilizes a PoW-based consensus, similar to Bitcoin's Nakamoto consensus [26], to ensure participants prove their contribution by finding a valid nonce. A valid nonce allows a participant to bundle transactions into a block candidate, which other participants verify before confirmation. However, PoW's race condition can create multiple block candidates, resulting in a fork. To address this, Ethereum uses the Greedy Heavies Observed Subtree (GHOST) solution [30], which finds the branch with the most computation power based on the number of blocks. Additionally, finalizing a block requires waiting for a chain extension with seven consecutive blocks.

3.2.2. Hyperledger fabric: Permissioned platform

Hyperledger Fabric, or simply Fabric, differs from other blockchain-based systems by adopting the execute-order-validate pattern instead of the order-execute structure. The Fabric transaction flow consists of three main phases: execution, ordering, and validation. A distributed application comprises a chaincode and an endorsement policy in this blockchain structure. A chaincode represents a smart contract implemented and executed in the execution phase. An endorsement policy is a static library that validates transactions in the validation phase. Initially, a client sends transactions to specific peers, who then execute and endorse them before returning the endorsed transactions to the client in the execution phase. Subsequently, the endorsed transactions move to the ordering phase, where a consensus mechanism orders them as a totally ordered sequence and packs them into a block candidate. In the validation phase, the block candidate is broadcast to all peers, who verify the endorsed transactions' operations following the endorsement policy and the execution's consistency.

Following the execute-order-validate architecture, Fabric distinguishes between three roles: clients, peers, and ordering service. Clients initiate transaction proposals as requests and queries to peers for execution and subsequently submit the responses of ordered transactions from peers to the ordering service. Peers are not only responsible for executing transaction proposals from clients but also for maintaining the blockchain ledger and smart contracts. Peers that execute the transaction proposals are referred to as endorsing peers or endorsers. Ordering services, known as orderers, utilize the well-known Raft consensus mechanism to establish a total order of transactions and form them into a block candidate.

[3] http://ethereum.org/en/developers/docs/evm/
[4] http://soliditylang.org

Raft [31] was introduced as a more understandable alternative to Paxos, a well-known consensus in many industries. Raft achieves this by separating its components, as proposed by Ongaro et al. [31]. There are two primary parts to Raft: leader election and log replication. In a leader election, participants are in one of three states: candidate, follower, or leader. Participants start a consensus round in the candidate state, waiting for a period to receive a notification or a vote request from the first awake participant. Once a participant receives a request vote, they switch to the follower state, and the first awake participant becomes the leader for the consensus round. The leader is then permitted to wrap and form a block candidate before propagating it to other followers in the log replication phase.

3.3. Trip planner

A trip planner, also known as a journey planner, is a transportation search engine. In the context of MaaS, it enables travellers to search for multi-modal transportation options from a source to a destination. OpenTripPlanner (OTP)[5] [32] is an open-source project licensed under LGPL[6] that provides passengers with transportation information through a multi-modal trip planner. OTP is based on a Java virtual machine and is actively developed for use on large operating systems such as Windows, Linux, and Mac. It has been used in various use cases in the USA and Europe.

The technology behind OTP is based on graph construction using data from OpenStreetMap (OSM)[7] or General Transit Feed Specification (GTFS).[8] GTFS is a format developed by Google for sharing public transit data, while OSM is a collaborative project for creating an accessible global geographic database. After considering various platforms, such as Google Maps API[9] and Here API,[10] we found OTP to be the most flexible for customizing travel plans. What sets OTP apart is its ability to allow service providers to create a graph based on their GTFS data, which can come from various sources through cooperation between providers. Using a third-party service, such as Google Maps or Here, would require providing data to those services, making transport providers reliant on a trip planner service.

4. Methodologies

Blockchain-based smart contracts offer a promising solution for addressing current challenges related to centralization in the MaaS system. Given the involvement of multiple parties and services, managing the system effectively and fairly while satisfying everyone's demands, especially the need for trust among entities, is crucial. By leveraging blockchain technology, the MaaS system can build trust without the need for intermediaries, potentially leading to its growth. Additionally, smart contracts' success in facilitating transparent and automatic collaboration between organizations offers the potential for self-organization within MaaS. Through the exchange of call requests among these smart contracts, the blockchain-based MaaS system can achieve autonomy.

4.1. Blockchain-based MaaS

The blockchain-based MaaS system can be broken down into three essential components: OTP/trip planner for generating trip solutions, machine learning or artificial intelligence for recommending the best solution based on historical data, and blockchain technology to support secure and decentralized communication. This work, however, focuses specifically on the blockchain component, which enables decentralization in the MaaS system.

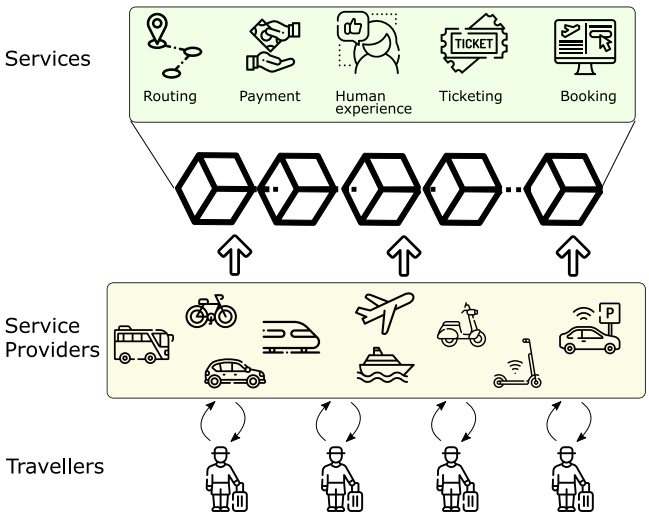

Fig. 3. An observation for a blockchain-based MaaS.

As illustrated in Fig. 1, a decentralized MaaS system based on blockchain technology has the potential to promote collaboration and trust among different transport providers. Collaboration is crucial to the development of MaaS and can be improved through transparency and standardized definitions among entities. As depicted in Fig. 3, each transport service provider maintains its blockchain for storing communication data. To this end, we proposed a blockchain-based MaaS [7] consisting of a trip planner, a transport data exchange protocol, and a blockchain protocol. The trip planner generates a set of possible transport solutions, while the data exchange protocol facilitates the exchange of transport data between service providers for the trip planner. The most critical component is the blockchain protocol, which is responsible for verifying tickets selected by travellers, maintaining unique data storage as a transaction history among service providers, and providing a smart contract platform for different subsystems to enable an autonomous and self-organized system.

In our foundational study [7], we introduced two key protocols, as shown in Fig. 4. The first focuses on the management of data routes, and the second is a blockchain-based protocol for validating tickets and coordinating operations between participants. Importantly, both protocols require synchronization among service providers to ensure data consistency. Specifically, route data synchronization aids in uncovering potential routes based on traveller requests. In contrast, the blockchain-based protocol synchronization ensures a uniform database across service providers for verification. Initially, as delineated in steps one through four of Fig. 4, the traveller requests route information to explore appropriate travel solutions. Following this, in step five, the traveller engages with smart contracts, utilizing the obtained route data, with the aim to reserve or procure the selected travel option. An in-depth examination and design of the smart contracts for blockchain-based MaaS are elaborated upon in the subsequent section.

Blockchain-based MaaS offers benefits for both service transport providers and travellers. By joining the platform, a service provider can proactively share data and passively receive information from other providers without having to request it. This voluntary exchange of information allows service providers

5 http://docs.opentripplanner.org/en/latest/History/
6 https://opensource.org/licenses/LGPL-3.0
7 https://openstreetmap.org
8 https://developers.google.com/transit/gtfs
9 https://developers.google.com/maps
10 https://developer.here.com

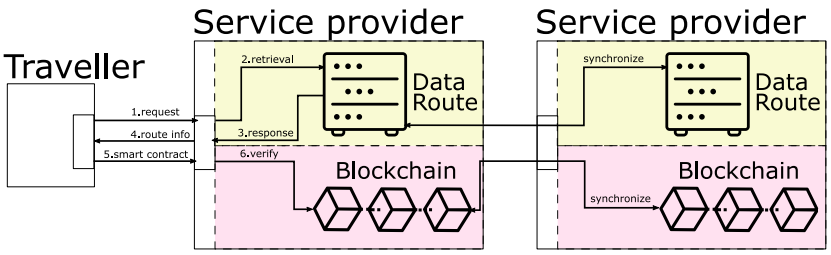

Fig. 4. An overall architecture with two main protocols.

to promote their services through advertisements rather than relying on a central database to store all transportation services. Instead, information is distributed among all participants, allowing for personalized routing and recommendations based on travellers' preferences. For travellers, routing is faster and more convenient, with the cooperation among transport providers made easier through the use of blockchain technology. This also reduces the need for infrastructure resources, as service providers can utilize each other's infrastructure for their operations.

4.2. Smart contracts for blockchain-based MaaS

Smart contracts for MaaS must determine the involved parties and their respective roles in the system. At a basic level, a mobility system involves at least four parties: travellers, service providers, insurers, and regulators. Although the workings of a mobility system can be complex, we simplify the critical players in a MaaS system. Traveller organizations represent individuals seeking travel information and tickets. Service providers are the main contributors to the MaaS system, offering transport services, while insurance organizations provide coverage to the parties involved in a transport system. Finally, the police organizations represent government policies that regulate transport across different regions, including border control. Travellers, as customers, interact with the system to access services. Service providers refer to transport providers such as bus, train, and flight companies and other transport services such as parking, car rentals, and gas stations. In addition to the two prominent organizations mentioned earlier, the MaaS system also includes an insurance party that provides insurance services for transport-related issues such as delays or cancellations. Furthermore, in the context of global considerations, border control is necessary to monitor and regulate movement based on restrictions. As illustrated in Fig. 5, our design requires travellers to create an account to access the system, add balance, and purchase tickets and insurance. Service providers are responsible for creating, verifying, and providing journey planning for tickets, as well as claiming insurance for transport issues. Once a traveller requests, the service provider creates the ticket, stores it in the blockchain, and verifies it through ticket controllers. Ticket verification may also require a connection to the police for criminal movement control, border checks, and pandemics such as COVID-19. Additionally, the service provider is responsible for confirming reimbursements for trip-related issues, such as cancellations, delays, or lost luggage. The system can also integrate with other services, such as the COVID-19 vaccination passport.

5. Experiments

The experiment section outlines the setup and results of performance evaluation methodologies based on two prominent blockchain platforms, Hyperledger Fabric and Ethereum. Also, the OTP platform is utilized in the implementation of the trip planner, allowing us to gauge the amount of data necessary for processing a ticket.

5.1. Adaptability and scalability measurement

The concept of adaptability in blockchain-based systems is broad and generally refers to the capacity to effectively utilize blockchain technology. Specifically, Conoscenti et al. [33] highlight the adaptability of blockchain in IoT in terms of scalability. This aspect measures the system's ability to rapidly respond to updates and fulfil real-time requirements, such as quickly adapting to market demands, product quality, or quantity changes in supply chain applications [34]. Other factors contributing to the adaptability of blockchain-based systems include flexibility in modifying term conditions, confidentiality, and security. Nevertheless, from a comprehensive performance perspective, the primary influence on adaptability measurement is often attributed to system scalability.

In assessing the performance of a blockchain-based system, scalability is a crucial factor to consider. This attribute is particularly important when the system involves a larger number of participants, as reaching consensus demands more coordination and effort among them [35,36]. Scalability reflects the system's capacity to accommodate an increasing number of participants without compromising its performance. A typical metric for measuring scalability is throughput, which indicates the number of requests or transactions the system can process per second. The system's ability to handle growing workloads without significant performance degradation can be assessed by evaluating throughput.

5.2. Experimental resources

The experiments were conducted using three main components: (1) a personal computer (PC), (2) a gaming Asus laptop, and (3) a ThinkPad laptop T15, as detailed in Tables 1 and 2, including their resources and main tasks. For device (1), we deployed six virtual machines for Hyperledger Fabric and three virtual machines for Ethereum. For device (2), we used four virtual machines for Hyperledger Fabric and two virtual machines for Ethereum. The client-side was handled by device (3), with sixteen terminals for sixteen clients as the default. Communication between the components was maintained using gigabit cables, a gigabit router (Asus AC 3200 Tri-band), and a gigabit switch (TP-link), ensuring a bandwidth of 1Gbps.

5.3. OTP: A trip planner

To set up a trip planner, OTP utilizes GTFS data to create a graph of public transport for possible transport solutions. Since OTP is implemented as a server for routing from source to destination, its setup requires a significant amount of memory to load as much data as possible to form numerous routes for requests. However, the main focus of OTP experiments is to investigate the amount of data required for a transport solution, including various means of transport. The experiment is based on public transport data regions of Tampere and Helsinki and railway data

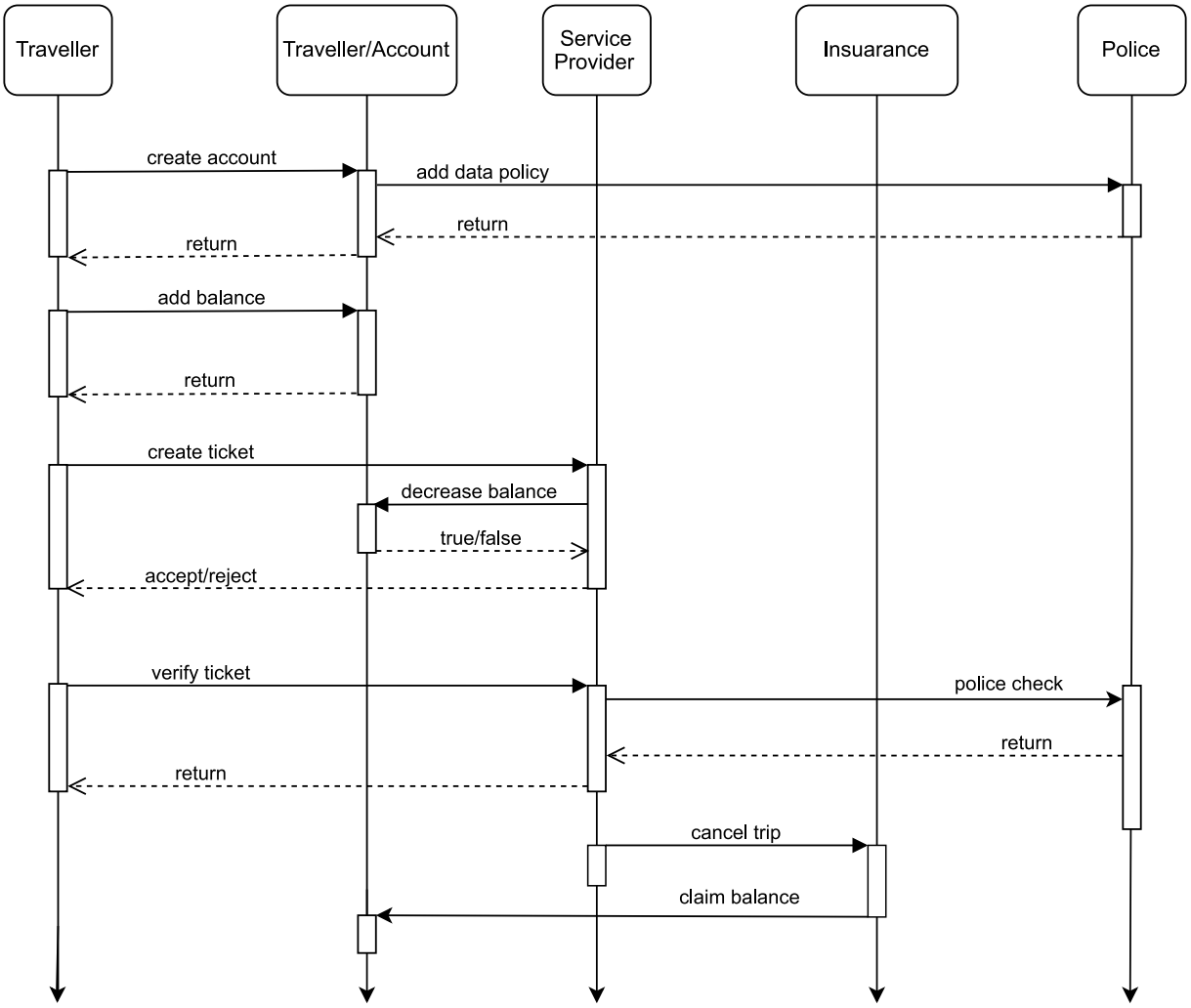

Fig. 5. The diagram for a smart contract with a blockchain-based MaaS among parties.

Table 1
The hardware resources for the experiment.

Devices	Processor	RAM	System	Tasks
(1) PC	i7-8700 CPU @ 3.20 GHz 3.19 GHz	32 GB	x64	3 Nodes
(2) Asus	i7-4720HQ CPU @ 2.60 GHz 2.59 GHz	16 GB	x64	2 Nodes
(3) ThinkPad	i5-10210U CPU @ 1.60 GHz 2.11 GHz	16 GB	x64	16 clients

Table 2
The hardware components for OTP, Hyperledger Fabric, and Ethereum setups.

Platform	Components	Resources	Setup Info
Fabric	Orderer	Process: 2, Memory: 2 GB	fabric-orderer:2.2.1 - 38.42MB
	Peer	Process: 4, Memory: 4 GB	fabric-peer:2.2.1 - 49.41MB
	Client	Process: 2, Memory: 4 GB	NodeJS vs SDK
Ethereum	Geth	Process: 4, Memory: 6 GB	Geth:1.10.12 vs Truffle:5.4.28
	Client	Process: 2, Memory: 4 GB	NodeJS vs web3
OTP	OTP	Process: 4, Memory: 10 GB	OTP-1.6.0 server

in Finland.[11,12] To be specific, a virtual machine is used to deploy the OTP-1.6.0 server via the VirtualBox tool, and it requires about 4 processors and 10 GB memory, as shown in Table 2.

The purpose of the trip planner is to find possible solutions for travellers. The experiments conducted with the planner focused on the amount of data required for each possible solution before sending an input request to the blockchain-based MaaS. After setting up the OTP with the public transport data for Tampere and Helsinki regions in Finland, we made several requests with longitude and latitude values for source and destination to obtain possible solutions using various means of public transport. The statistics for the results are shown in Table 3, which indicates that the number of available means of transport (# transports) affects the capacity of solution information, including transit information, directions, price, stops, time, and location. The amount

11 https://data.opendatasoft.com/explore/dataset/fi%40navitia/table/
12 https://transitfeeds.com/l/530-finland

Table 3
Capacity from a transport solution with the different number of transport means.

# transports	Details	Maximum capacity (KiB)
3	walk, light-rail, walk	6.2
5	walk, bus, walk, light-rail, walk	10.8
7	walk, bus, walk, train, walk, light-rail, walk	17.4
8	walk, bus, walk, train, train, walk, bus, walk	23.3

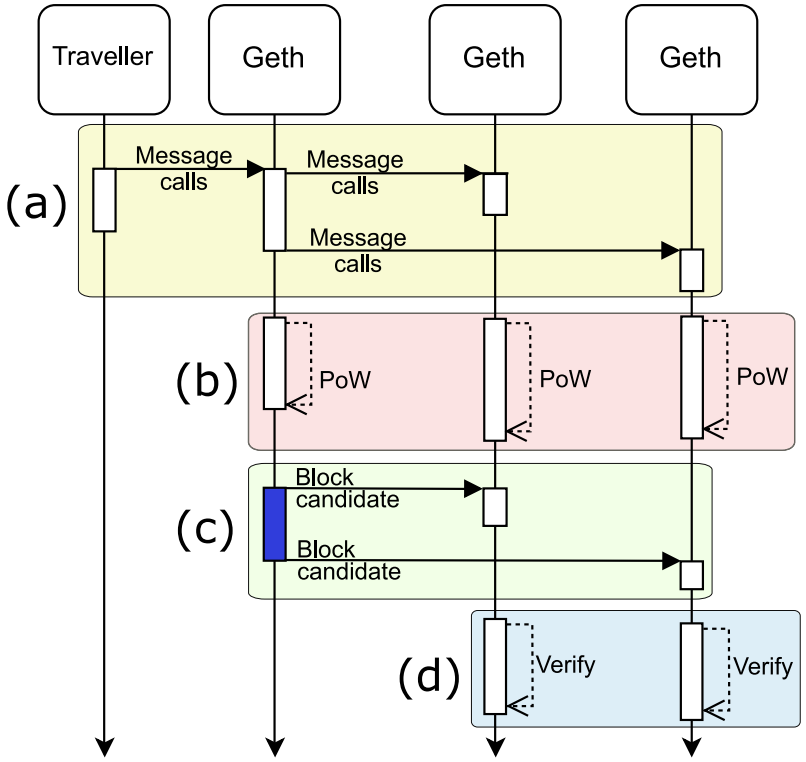

Fig. 6. Sequential diagram for Ethereum workflow with the three-participant network, including four components: (a) message call propagation, (b) PoW computing, (c) block candidate propagation, (d) verification of block candidate. The blue box is the process of turning to the leader from a basic node as the first participant, finding a suitable nonce for the consensus round.

of data for possible solutions also vary depending on the number of stops for each means of transport, even if the # transports are the same. We performed multiple attempts to obtain the same number of transports and selected the maximum value before recording the data in Table 3. For instance, in the last row of Table 3, where the number of transport means is one more bus than the second-to-last row, the additional data is approximately 7 KiB. Therefore, the length of the trip is not the determining factor for the amount of information but the number of stops for each means of transport.

5.4. Blockchain platforms

The following subsection describes the setup and experimental results for the blockchain platforms. We will introduce the network's number of nodes before delving into the details and analysing the obtained results.

5.4.1. Ethereum

The Ethereum network setup consists of several virtual machines, each with 4 processors and 4 GB of memory, as shown in Table 2. These machines operate on an x86/64 OS and are deployed with the Geth protocol version 1.10.12-stable for Ethereum. Smart contract deployment is done using Truffle version v5.4.28,[13] which offers a framework for compiling and migrating smart contracts to the blockchain network. The three-participant network is implemented using three virtual participants from device (1) in Table 1, while the five-participant network includes three virtual machines from device (1) and two virtual machines from device (2).

A blockchain-based MaaS system utilizing the Ethereum network is depicted in Fig. 6. The setup involves three participants running the Geth protocol. During the message propagation period, travellers' service requests are broadcast to all participants. At each consensus round, participants compete to find a valid nonce to become the leader and generate a block candidate, which other participants then verify. The system's performance is affected by the network parameters, such as the difficulty value for PoW-based consensus, the maximum gas limit for block size, and the number of participants. Therefore, experiments are conducted with varying parameters. The gas unit measures the amount of work required by Ethereum, such as smart contract execution complexity and block candidate size restriction. The parameters are explained in more detail below.

- *Gas limit* is about the restricted amount of gas that can be used in a block; hence, instead of limitation of block size, Ethereum desires to form limitation in the amount of used gas per block

[13] https://trufflesuite.com/

Table 4
Difficulty parameters of Ethereum's consensus.

# peers	3 peers			5 peers				
Block difficulty	375k	500k	625k	750k	1000k	1125k	1250k	1375k
Block creation (s)	1.015	1.173	1.600	1.774	1.630	2.146	2.380	2.487
Block uncle	53	21	13	10	59	38	38	31
Block lost	40	0	0	1	4	0	0	1

- *Difficult value for PoW-based consensus* determines the hard level that the required computation is to gain the permission to form a block candidate
- *Participants* are in the network or the number of peers in the network

By default, the Ethereum platform configures the difficulty for its PoW-based consensus mechanism in the genesis file, but this value adjusts over time to accommodate network growth. We sought to maintain a fixed difficulty value for our experiments, but selecting an appropriate value for three-participant and five-participant networks presented a challenge. To address this, our first experiment estimated practical difficulty values for Ethereum PoW on different participants, given the varying levels of computation within networks. The three-participant network demonstrated a range of block difficulty values between [250 : 1000], while the five-participant network utilized values between [875000 : 1500000]. Our findings, presented in the second row of Table 4, show the mean time it took to create a block (in seconds) after mining around 1000 blocks to find a suitable nonce for consensus. While a block uncle is not an orphan block in a general permissionless blockchain, Ethereum considers it a runner-up leader for the consensus round. However, uncle blocks can lead to network forks and segmentations, ultimately affecting the uniqueness of blockchain data. Thus, the system should avoid issues related to forks and reduce the number of lost blocks resulting from uncle blocks. Once again, lost blocks represent a waste of computation and energy in finding suitable nonces for consensus.

To ensure efficient block creation and performance in our Ethereum-based experiments, we have selected block difficulty values of 500k and 1125k for the three-participant and five-participant networks, respectively, based on the results from Table 4. While avoiding fork and segmentation is a complex configuration issue, we have taken steps to reduce the probability of these issues occurring. In general, we have observed that an increase in block difficulty leads to a higher rate of block creation per second, but this is inversely proportional to the occurrence of block uncle and block loss.

5.4.2. Hyperledger fabric

The Hyperledger Fabric network comprises various machines for Orderers and Peers. While setting up these services can be done with a single virtual machine, we prefer to separate them into distinct virtual machines. Hardware details are provided in Table 2. In terms of physical resources, as seen in Table 1, the first machine runs six virtual machines to establish a three-participant network, while the second machine adds four more virtual machines to expand the network to five participants. In line with previous studies on performance experiments [37], we found that multicast update messages can enhance efficiency. As a result, most configurations in Hyperledger Fabric utilize multicast communications from the client to the server.

As shown in Fig. 7, the three-participant network is configured with three machines serving as Peers and three machines as Orderers. The travellers initiate proposals or requests to peers for execution, and once they receive responses, they submit an endorsed transaction proposal to the Orderers. The three Orderers then conduct an election scheme to select the leader for the consensus round. The leader subsequently creates a block candidate and broadcasts it to the participants for verification. To optimize the system, we evaluated various parameters, such as the number of Peers and Orderers, as well as the block size. We also considered the division of peers into endorsing and committing peers, but we simplified the architecture by combining these tasks into a single peer. For the five-participant network, ten virtual machines are required for Peers and Orderers, distributed as shown in Table 1. Physical device (1) is used to deploy six virtual machines, equally split between Orderers and Peers. Meanwhile, machine (2) is used to distribute four virtual machines evenly to Peers and Orderers.

The two crucial aspects of blockchain consensus are communication and block formation restrictions. In Hyperledger Fabric, a batch or block can serve as a block candidate if it meets one of two conditions: (1) it times out, or (2) it reaches the maximum data size. While acknowledging the issues of liveness that may arise in blockchain consensus, we opted to maintain the default format for the Raft configuration, including TickInterval, ElectionTick, HeartbeatTick, MaxInflightBlocks, and SnapshotIntervalSize. This decision was made to avoid risks associated with election algorithms, which fall outside the scope of this paper. Therefore, the Hyperledger Fabric configuration is primarily focused on wrapping a block candidate as follows:

- *Preferred max bytes* as block size is the trigger to wrap a block candidate if the capacity of submitted transactions satisfies
- *Participants* are in the network or the number of nodes in the network

5.4.3. Performance comparison

Based on the results of our experiments, this subsection provides a performance comparison between Fabric and Ethereum in the context of permissioned and permissionless blockchain platforms. Table 7 presents the performance of Ethereum, while Table 6 presents the performance of Fabric. Both platforms were tested in the same physical environment, which included similar hardware components, network conditions, computational capacity, and other relevant parameters, such as the number of participants, block size, and transaction updates or queries.

- The numbers of peers are three and five to form three-participant and five-participant networks, respectively
- The transaction types are update and query. The former aim to change information via writing replaced data, while the latter is to read information from the blockchain
- Transaction size is about the diversity of transaction capacities in set {32, 64, 128} KB
- Block size shows the variety of block capacity in the set {128, 256, 512} KB

Although we adjusted the parameters mentioned above, we conducted our experiments using a ThinkPad T15, as specified in Table 2. We utilized 16 clients, each of which sent 100 requests for updating and 100 requests for querying information to the system. As a result, we deployed 16 smart contracts to facilitate the interaction between the 16 clients and the system.

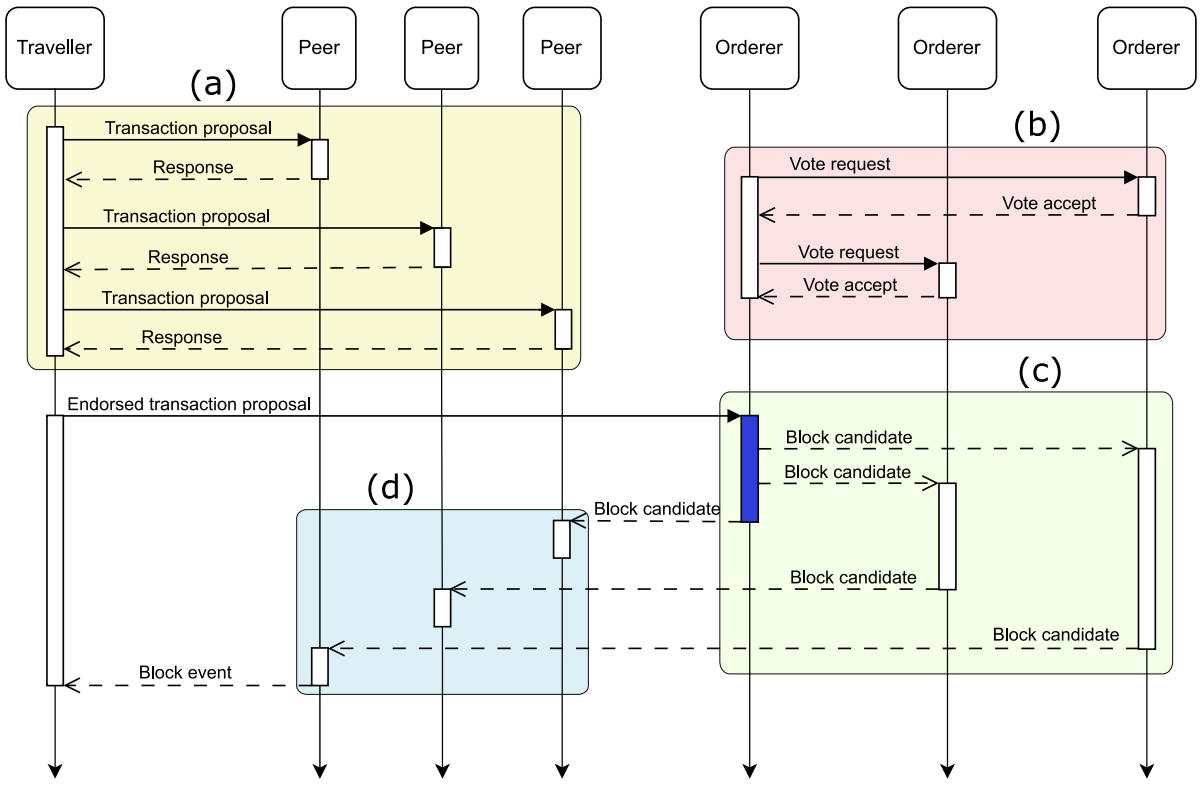

Fig. 7. The diagram for a blockchain-based MaaS with Hyperledger Fabric through a set-up on three Peers and three Orderers via four components: (a) execution phase, (b) leader election for the current consensus round, (c) order phase, and (d) validation phase. The blue box is the process of turning to the leader of the consensus round from RAFT.

Table 6 presents the results of our experiments on the performance of Hyperledger Fabric with varying block and transaction sizes. Our findings indicate that the average time to confirm a transaction increases with transaction size when the block size is held constant across both networks and types of transactions. For instance, in the three-participant network with a block size of 128 KB, the time to confirm an update request transaction increases from 5140.870 ms to 5727.82 ms and 6340.05 ms for transaction sizes of 32 KB, 64 KB, and 128 KB, respectively. Similarly, for query requests, the corresponding times are 55.412 ms, 68.005 ms, and 97.296 ms. These results suggest that as the amount of data being processed increases, more blocks are required, which in turn affects the system's performance. In addition, increasing the block size while keeping the transaction size fixed should improve performance. However, our results show that this is only true for transaction sizes of 32 KB and 64 KB in both networks and transaction types. For example, in the three-peer network, the confirmed period for updates at the 32 KB column reduces from 5140.870 to 2428.354 and then 1534.130 ms for block sizes 128, 256, and 512 KB, respectively. This trend is consistent across all transaction types and networks. This result is logical because transaction confirmation depends on block confirmation. Thus, increasing the block size reduces the number of confirmed blocks, leading to a doubled performance improvement. Another interesting observation is the slight increase in confirmed periods in the five-participant network's results compared to the three-participant network due to the more significant communication needs. Additionally, the query requests dominate the comparison between transaction types because queries are only read operations, whereas updates require a confirmation from multiple network entities.

The periods fluctuate in the 128 KB transaction size for both the three-participant and five-participant networks. The worst results occur when the block size is 128 KB, but the performance improves when the block size is 512 KB, for example, 5597.050 ms in the update type for the three-participant network. The fluctuation occurs because the number of transactions confirmed in a block changes. When the block size increases from 128 KB to 256 KB, the number of transactions involved in a block increases from one to two, but the processing becomes worse because processing two transactions is worse than one transaction for a block size of 128 KB. The same result is observed in both transaction types and networks. However, when the block size is 512 KB, which can wrap at most four transactions of 128 KB, the performance becomes much better. Therefore, despite the processing of four transactions in a block, the period of forming a block proves to have a repercussion on performance. However, in the case of fewer transactions in a block, for example, two transactions per block, the system is impacted by processing transactions in a block more than the number of blocks.

Table 7 displays the results of our Ethereum experiments, which involved varying messages and block sizes. Throughput for write and read requests were measured for update and query operations, respectively. One notable difference between Ethereum and Fabric is that block formation in Ethereum is based on the total gas of transactions, not on capacity units. As a result, gas-to-byte unit conversion was necessary. Interestingly, the performance trends observed in Ethereum closely mirror those seen in Fabric (Table 6). Specifically, performance improves when the block size is increased, and when transaction size is held constant, performance deteriorates as transaction capacity increases. These experiments highlight the similarities in performance behaviour between the two blockchain platforms.

Another exciting aspect is comparing the results between these platforms in the same experiments, which shows that most of the results from Fabric outperform Ethereum. However, in the case of a transaction size of 32 KB, the opposite is accurate,

Table 5
Comparison between Hyperledger Fabric and Ethereum.

Features	Fabric	Ethereum
Smart contract	Go, NodeJS, or Java	Solidity or Vyper
Identity	Certificate	Public-Private key
Access permit	Permissioned	Permissionless
Consensus	Raft	PoW
Data model	Key–value	Account base
Library interaction	SDK for NodeJS, Go, Java, and Python	Web3 for Python, Php, Java, etc.
Environment	Docker and others	EVM
Security	CFT	BFT
Application	-	Cryptocurrency (Ether)
Incentive strategy	-	Rewarding contributors and pay-as-benefits
Database	CouchDB or LevelDB	LevelDB
Client task	Submit function calls and endorsements	Submit function calls
Policy	Signature and implicit meta policies	-
Server task	Orderer and Peer	Geth

Table 6
Hyperledger Fabric (ms/tx) with various parameters.

# peers	3 Peers						5 Peers					
Transaction type	Update			Query			Update			Query		
Tx size (KB) / Block size (KB)	32	64	128	32	64	128	32	64	128	32	64	128
128	5140.870	5727.820	6340.050	55.411	68.005	97.296	5151.010	6316.530	8493.080	64.591	81.446	123.306
256	2428.354	5684.720	7989.760	54.870	64.5531	112.711	2628.410	6131.860	8579.560	67.457	80.585	125.017
512	1534.130	2907.780	5597.050	51.579	63.245	99.085	1693.560	3200.440	8648.480	68.2475	88.4694	122.972

Table 7
Ethereum (ms/tx) in various parameters.

# peers	3 Peers						5 Peers					
Transaction type	Update			Query			Update			Query		
Tx size (KB) / Block size (KB)	32	64	128	32	64	128	32	64	128	32	64	128
128	4890.700	8564.410	18744.960	52.975	309.664	489.421	6222.850	14816.848	26386.500	52.289	137.583	383.353
256	2303.160	4949.214	10045.900	103.416	284.690	635.639	3420.960	6684.250	15436.200	51.753	132.140	463.131
512	1436.010	3153.980	5822.260	49.445	222.056	642.314	2756.840	4337.340	8314.060	53.800	141.243	333.704

and the Ethereum results are better than Fabric for both request types. This difference can be attributed to the computational demand for the consensus mechanism in Ethereum, which is lower than Fabric due to its different technology that requires more communication. The differences in results between these platforms will be further discussed in Section 6.

From a business perspective, permissioned blockchain is suitable for MaaS ecosystems as it provides connectivity among service providers based on identity, reputation, authentication, and authorization, minimizing leakage of traveller information and enabling information retrieval by authorities. Permissioned blockchain also offers scalability, as seen in the domination of Fabric results in Table 6 compared to Ethereum performance in Table 7. In particular, Fabric outperforms Ethereum, especially with three to five participants, in our experiments with query requests.

6. Blockchain-based MaaS: A discussion

6.1. A technical framework for blockchain platform comparison

In analysing blockchain platforms, it is essential to note significant variations in design concepts across different platforms. These include differences in smart contract functionality, identity and access permissions, consensus mechanisms, data models, library interactions, environmental setups, security measures, applications, incentive strategies, databases, policies, and server-client tasks. This subsection provides an in-depth comparison of these design concepts and summarizes them in Table 5.

6.1.1. Smart contract

In Fabric, chaincode and smart contract concepts are used interchangeably to represent the business logic of interactions between parties via executable code. Chaincodes are written in Go, NodeJS, or Java. In contrast, in Ethereum, smart contracts are an account type in the system with a balance and address. Users interact with smart contracts by submitting transactions to call functions defined in the contracts. Smart contracts cannot be removed or modified due to the integrity of hash primitives. Solidity or Vyper is used to program smart contracts.

Discussion: As smart contracts in Fabric are built using well-known programming languages, it helps reduce issues related to programming languages, such as type checking in compiler design. However, the development of programming languages can still affect the Fabric system. Overall, the use of well-known programming languages reduces the complexity of programming language design but may present challenges in optimization. On the other hand, Ethereum uses a novel programming language to meet the specific requirements of its system. However, based on its development history, Solidity has faced several vulnerabilities that can compromise trust and security.

6.1.2. Identity

As a permissioned blockchain, Fabric uses a traditional public key infrastructure with a hierarchical model and X.509 certificate authority. The Orderer uses Membership Service Provider (MSP) to verify digital signatures signed by Peers and supports authorization based on participants' organizations. Fabric offers two policy types: signature policy, which involves arbitrary combinations of evaluation rules like AND, OR, and NOutOf, and implicit meta policy, which aggregates the result from evaluating policies in a configuration hierarchy. On the other hand, Ethereum's account identity is based on a key pair using the Elliptic Curve Digital Signature Algorithm, and its public address is derived by applying the Keccak-256 hash function to the last 20 bytes of the public key. The contract creator creates contract addresses.

Discussion: As a permissioned blockchain platform, Fabric employs an identity scheme based on organization and policy for entities, which benefits authorization and authentication from a security perspective. This identity scheme also helps establish trust in communication between participants. In contrast, Ethereum offers greater freedom, but it comes at the cost of optimization and security concerns.

6.1.3. Data model

Hyperledger Fabric has two distinct components: world state and blockchain. The world state represents the current view of the system, allowing for the retrieval of current values from any object in the system, while the blockchain records any changes that affect the current state. The world state is structured as a key–value model, similar to a database, and supports three basic operators: get, put, and delete, which interact with objects in the system. On the other hand, Ethereum follows a traditional account-centric data model where each account manages its own set of transactions, balance, codeHash, which represents the code of the account on the EVM, and storageRoot, which is the root node of a Merkle Patricia trie containing the account's storage content. Ethereum introduces two types of accounts: externally owned accounts, which anyone with a private key manages, and contract accounts that represent smart contracts on the network via controlled code.

Discussion: Fabric adopts a key–value model for its world state, similar to Ethereum's account model, but with a primary database storage scheme. However, Fabric has a unique limitation: it cannot accept multiple updates on the same object in a block, which is not the case for Ethereum. This limitation may arise from Fabric's execute-order-validate approach, while Ethereum trusts the person who solves the PoW for the current round to order transactions in a block and maintain the sequence.

6.1.4. Library interaction

Fabric provides an SDK environment with libraries supporting NodeJS, Java, Python, and Go programming languages for developing chaincode applications. Meanwhile, Ethereum supports several libraries based on different programming languages through the web3 library, including Python, PHP, and Java.

Discussion: Through its active community, Ethereum has gained extensive support and offers various libraries for different programming languages similar to Fabric. Additionally, the Ethereum community has developed many frameworks for interacting with and deploying smart contracts.

6.1.5. Environment

In Hyperledger Fabric, the concept of chaincode is similar to that of a smart contract in Ethereum. However, instead of being based on a virtual machine, Fabric creates a container, known as Docker, for each deployed chaincode. Ethereum's EVM, on the other hand, uses a transient memory and executes smart contract bytecode, making it responsible for the deployment and execution of smart contracts. Essentially, EVM serves as the runtime environment for smart contracts.

Discussion: As for the discussion, Fabric's use of containers allows for the deployment of different smart contracts in separate environments, which could improve the system's security. Meanwhile, Ethereum primarily deploys smart contracts in EVM.

6.1.6. Network topology

Hyperledger Fabric requires a configuration file to act as a gateway for connecting to the system, given the diversity of duties and participants involved. On the other hand, the Ethereum network topology does not necessitate communication from the client side. Instead, Ethereum offers Web3 as an SDK that provides protocols such as WebSocket and HTTP for interaction. Here are more details:

- Server task. In the Hyperledger Fabric platform, there are different types of nodes, such as orderers, endorsing peers, and committing peers. Endorsing peers are responsible for endorsing each request made by clients, but they do not update the ledger. The orderer collects these endorsements and creates a block, which is then propagated to all peers for validation and added to the ledger upon confirmation. To participate in the Ethereum system, users need to maintain a software called Geth, which allows them to become a peer of the network.
- Client task. To achieve execute-order-validate, clients in Hyperledger Fabric must create a transaction before submitting it to the endorsing peers. The client then waits for the endorsed transactions to be collected before submitting them to the orderers to create a block candidate that all peers subsequently commit. Meanwhile, in a conventional blockchain system like Ethereum, clients create transactions and send them to Ethereum nodes for execution and validation after the consensus round with a block candidate.

Discussion: Fabric introduces a decoupling of tasks between clients and servers. This decoupling allows for a flexible and innovative procedure called execute-order-validate, which separates consensus and execution tasks among participants. However, this approach comes with a cost in terms of communication between orderers, peers, and clients. As a result, Fabric's communication costs are likely to be higher than those of Ethereum, and Fabric clients have more responsibilities compared to Ethereum clients.

6.1.7. Application

While Hyperledger Fabric does not have detailed applications, Ethereum has its own native cryptocurrency called Ether, which enables the use of incentive mechanisms in the system.

Discussion: Unlike Ethereum, Fabric does not have an inherent cryptocurrency or default incentive strategy, as it does not initially target applications with cryptocurrency. Therefore, while Ethereum uses its cryptocurrency for an incentive strategy and other applications requiring transactions, Fabric does not have a default approach for incentivizing participants.

6.1.8. Incentive strategy

The Hyperledger Fabric protocol does not employ any incentive strategy. On the other hand, Ethereum utilizes an incentive strategy similar to the Bitcoin protocol. Specifically, the system rewards the proposer with a certain amount of Ether for various contributions, such as executing smart contracts, solving puzzles for block proposals, verifying, and disseminating block candidates.

Discussion: As Ethereum operates using a cryptocurrency, it can utilize incentive strategies by rewarding participants for contributing to the system's growth through tasks such as resolving puzzles, executing smart contracts, verifying, and propagating block candidates.

6.1.9. Consensus and security

Using the Raft consensus mechanism, Fabric employs a leader-based approach where the leader for each round is chosen randomly, providing Crash Fault Tolerance (CFT) to address network participant failures. However, Fabric cannot protect against arbitrary failures. On the other hand, Ethereum utilizes a PoW consensus mechanism, allowing anyone to participate and propose a block candidate by solving a computational puzzle. While Byzantine Fault Tolerance (BFT) is achievable, Ethereum faces challenges with performance and security due to race condition conflicts. To address this, a latency period is required before confirming a block proposal, for example, a block is only confirmed if it has six consecutive successors in both Ethereum and Bitcoin.

Discussion: Fabric and Ethereum use different consensus mechanisms for their blockchain systems, with Fabric using Raft to support CFT and Ethereum using a PoW concept to satisfy BFT. While Fabric keeps the same leader until a crash occurs, Ethereum requires the leader to solve a puzzle to prove correctness before forming a block candidate. Ethereum also faces issues related to conflicts between leaders, requiring a confirmation period. In comparison, Fabric incurs two main communication costs from clients to Peers and Orderers, while Ethereum has two idle periods for finding a leader and block confirmation.

7. Blockchain-based MaaS: Challenges, open questions, and opportunities

Although blockchain has many benefits in enhancing MaaS, there are still challenges that need careful consideration for practical implementation. This section discusses open questions for future directions in blockchain-based MaaS, including the usage of trip planners and blockchain.

7.1. Trip planner: Challenges, open questions, and opportunities

The implementation of blockchain technology within MaaS platforms presents a unique set of challenges due to the diversity of components involved, each introducing distinct issues. We aim to dissect and discuss the challenges in two corresponding subgroups of journey planner and blockchain technology in forming a blockchain-based MaaS.

7.1.1. Challenges and open questions

Examining the intricacies of trip planners is a critical step towards fostering the reality of the blockchain-based MaaS. Given that the primary role of a trip planner is to generate potential or finalized plans based on user requests, it necessitates access to a substantial volume of traffic information, particularly from an array of service providers within a decentralized framework. This decentralization leads to another significant challenge — the standardization of routing information across these service providers, which is crucial for the adaptability of the trip planner's algorithm. Furthermore, real-time updates or modifications due to unforeseen events present an additional challenge, as any change in traffic information can have a considerable impact on the trip planner. It is important to note that these challenges primarily stem from the system's decentralized nature, where there is an absence of a central authority overseeing the collection and normalization of routing information.

In light of these challenges associated with trip planners, several pertinent questions arise when considering the development of a blockchain-based MaaS system. The foremost concern is the establishment of an effective protocol for exchanging traffic information, which is critical for connectivity among service providers. Thus, the question arises — what could be an appropriate protocol to facilitate this exchange? Alongside this, another question pertains to the standardization of route information among service providers, which would enhance the adaptability of trip planners' routing and exchange algorithms. Lastly, another aspect to be addressed is the capability of trip planners' algorithms to support real-time updates of traffic routes. For instance, the OTP requires traffic graph reconstruction with every update, which can have implications for system performance, computation, and even the latency and throughput of the blockchain.

7.1.2. Potential solutions

In the pursuit of standardizing travel information, it is essential for a trip planner to accommodate a diverse range of data types. Alternatively, it should have in place pre-processing methods for converting between different formats of travel information. This adaptability can facilitate participation from various service providers. Moreover, adopting a set of standard data types such as NeTEx,[14] TransXChange,[15] SIRI,[16] OpenStreetMap,[17] DATEX II,[18] and OpenLR[19] can establish a baseline for compatibility.

The challenge of providing real-time travel updates is more complex than mere data standardization. Rather than rebuilding the entire traffic graph to accommodate changes, the trip planner should be capable of modifying the existing graph. Additionally, the use of a gossip protocol may prove a potent solution for disseminating information across service providers efficiently.

7.2. Blockchain technology: Challenges, open questions, and opportunities

7.2.1. Challenges and open questions

Leveraging blockchain technology encourages and enhances connectivity among entities via trust established in a general transport ecosystem or a specific MaaS. However, following recent works on blockchain challenges [38], blockchain also introduces many drawbacks, including performance issues, confidentiality concerns, and security issues related to data conflicts from decentralization.

One of the biggest weaknesses of blockchain deployment is the confidentiality of sensitive information of participants and users. In detail, from the confidentiality perspective, travellers require a secure protocol that ensures that other entities cannot access or disclose private information associated with transactions. However, leveraging blockchain technology discloses the users' travelling information among network participants; hence, the adversaries can track the travellers for aiming evil behaviours. Another challenge with blockchain technology is performance. While it enhances query speed and availability through decentralization, update requests require agreement among the majority of participants, leading to increased communication and computation depending on the chosen technological platform.

7.2.2. Potential solutions

Depending on the type of travel, a customer's identity may need to be verified to issue a ticket, as in the case of cross-border travel and movement restrictions. An individual's identity can be verified either during account creation or later, but it is not revealed to outsiders through any system operation. Every transaction between a traveller and a service provider via a smart contract is encrypted and authenticated. Additionally,

[14] https://netex-cen.eu/

[15] https://www.gov.uk/government/collections/transxchange

[16] https://www.siri-cen.eu/

[17] https://wiki.openstreetmap.org/wiki/OSM_file_formats

[18] https://www.datex2.eu/

[19] https://www.openlr-association.com/index.html

each participant in the blockchain is represented by a pseudonym corresponding to a public key, and a single pseudonym represents a single account that is not linked to the actual identity of an individual.

Possible strategies for enhancing the performance of a blockchain-based system could arise from improved connectivity amongst blockchain-based subgroups. Sharding is a technique that enables a blockchain system to be dispersed across multiple subgroups, thereby reducing the extensive communication needed when broadcasting system changes. Alternatively, leveraging advanced distributed ledger technology could further enhance performance. For instance, the Directed Acyclic Graph (DAG) concept, which allows the system's structure to evolve in a graph-like pattern of parallel growth, provides a viable alternative to the linear development typically associated with blockchain. This approach can lead to significant improvements in processing speed and overall system performance.

The concept of off-chain data storage presents a viable solution for enhancing the performance of a blockchain-based system. This approach essentially entails the external management of data, allowing the blockchain to maintain a more streamlined operation by primarily dealing with the hashes of transactions. This not only minimizes the extensive communication capacity otherwise required but also ensures the integrity of the system through thorough auditability. One effective technological implementation of this concept is the InterPlanetary File System (IPFS),[20] a protocol designed to facilitate distributed storage independent of the blockchain. By leveraging IPFS, a blockchain-based system, especially blockchain-based MaaS, can achieve greater scalability while preserving the inherent decentralization and security aspects of blockchain systems.

8. Conclusion

Recently, there has been a growing need to improve travellers' experience and preferences in transportation through studies on emerging technologies, such as big data and graph theory. The Mobility-as-a-Service (MaaS) concept, which connects service providers in a transport ecosystem, has become a leading platform in ITS. However, the traditional MaaS lacks trustworthiness in collaboration among service providers. Blockchain technology can contribute to the growth of trustworthiness in connectivity among service providers through immutability and transparency. This work emphasizes the formation of a blockchain-based MaaS via realistic deployments and experiments with architecture and proposed smart contracts, primarily based on OpenTripPlanner, Hyperledger Fabric, and Ethereum. The study found that reducing the number of blocks by increasing block capacity is an efficient solution, but only if a block warps at least four transactions. The performance of Fabric was found to be superior to Ethereum in general view and fair comparisons. The study discusses possible blockchain-based MaaS via dissimilar technologies and the performance of blockchain types from the proposed technical framework for blockchain platform comparison. The work concludes with open challenges, highlighting opportunities and future directions in the development of blockchain-based MaaS.

A notable limitation of this study pertains to the absence of thorough consideration of privacy aspects. Given the sensitive nature of travellers' information, the employment of blockchain technology, which inherently necessitates the disclosure of data for verification, can raise serious privacy concerns. This hurdle represents a significant challenge not only for the expansion of blockchain-based MaaS but also for blockchain systems more generally. That being said, the ongoing advancement of zero-knowledge proof schemes, which can validate transactions without revealing the underlying data, offers a promising avenue for addressing this privacy challenge.

Given the current limitations of both blockchain technology and trip planners, future research stemming from this work should focus on their respective advancements. Specifically, implementing techniques such as sharding and off-chain solutions could significantly improve the performance of blockchain-based MaaS systems. Furthermore, harnessing the power of innovative trip planners presents another avenue for expansion from the current study, which could further facilitate the integration of blockchain technology in such systems. All these enhancements would be instrumental in increasing the adoption and efficacy of blockchain usage in the field of MaaS.

CRediT authorship contribution statement

Tri Nguyen: Conceptualization, Methodology, Resources, Investigation, Validation, Writing – original draft, Writing – review & editing. **Huong Nguyen:** Investigation, Validation, Writing – original draft, Writing – review & editing. **Juha Partala:** Writing – original draft, Reviewing and editing. **Susanna Pirttikangas:** Conceptualization, Reviewing and editing, Project administration, Funding acquisition, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Tri Nguyen reports financial support was provided by University of Oulu. Tri Nguyen reports a relationship with University of Oulu that includes: employment and funding grants. All authors are employed by the same organization named the University of Oulu.

Data availability

No data was used for the research described in the article.

Acknowledgements

This research is done in a strategic research project *TrustedMaaS* under focus institute Infotech Oulu, Finland, University of Oulu, Finland, Academy of Finland, 6G Flagship, Finland (grant 318927), and ECSEL JU FRACTAL, Finland (grant 877056). A personal grant by the Nokia Foundation and Tauno Tönning Foundation for Mr. Tri Nguyen.

References

[1] L. Tan, K. Yu, L. Lin, X. Cheng, G. Srivastava, J.C.-W. Lin, W. Wei, Speech emotion recognition enhanced traffic efficiency solution for autonomous vehicles in a 5G-enabled space–air–ground integrated intelligent transportation system, IEEE Trans. Intell. Transp. Syst. 23 (3) (2022) 2830–2842, http://dx.doi.org/10.1109/TITS.2021.3119921.

[2] G. Li, Y. Yang, S. Li, X. Qu, N. Lyu, S.E. Li, Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness, Transp. Res. C 134 (2022) 103452, http://dx.doi.org/10.1016/j.trc.2021.103452.

[3] S. Chavhan, D. Gupta, C. Nagaraju, A. Rammohan, A. Khanna, J.J.P.C. Rodrigues, An efficient context-aware vehicle incidents route service management for intelligent transport system, IEEE Syst. J. 16 (1) (2022) 487–498, http://dx.doi.org/10.1109/JSYST.2021.3066776.

[4] P. Jittrapirom, V. Caiati, A.-M. Feneri, S. Ebrahimigharehbaghi, J. Alonso González, J. Narayan, Mobility as a service: A critical review of definitions, assessments of schemes, and key challenges, Urban Plan. 2 (2) (2017) 13–25, Copyright - Copyright Cogitatio Press 2017; Last updated - 2019-04-11.

20 https://ipfs.tech/

[5] T. Storme, J. De Vos, L. De Paepe, F. Witlox, Limitations to the car-substitution effect of MaaS. Findings from a Belgian pilot study, Transp. Res. A 131 (2020) 196–205, http://dx.doi.org/10.1016/j.tra.2019.09.032, Developments in Mobility as a Service (MaaS) and Intelligent Mobility.

[6] A. Polydoropoulou, I. Pagoni, A. Tsirimpa, A. Roumboutsos, M. Kamargianni, I. Tsouros, Prototype business models for mobility-as-a-service, Transp. Res. A 131 (2020) 149–162, http://dx.doi.org/10.1016/j.tra.2019.09.035, Developments in Mobility as a Service (MaaS) and Intelligent Mobility.

[7] T.H. Nguyen, J. Partala, S. Pirttikangas, Blockchain-based mobility-as-a-service, in: 2019 28th International Conference on Computer Communication and Networks, ICCCN, 2019, pp. 1–6, http://dx.doi.org/10.1109/ICCCN.2019.8847027.

[8] Y. Yuan, F.-Y. Wang, Towards blockchain-based intelligent transportation systems, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems, ITSC, 2016, pp. 2663–2668, http://dx.doi.org/10.1109/ITSC.2016.7795984.

[9] F. Kitahara, K. Kera, K. Bekki, Autonomous decentralized traffic management system, in: Proceedings 2000 International Workshop on Autonomous Decentralized System (Cat. No.00EX449), 2000, pp. 87–91, http://dx.doi.org/10.1109/IWADS.2000.880891.

[10] K. Mori, Autonomous decentralized systems technologies and their application to a train transport operation system, in: V.L. Winter, S. Bhattacharya (Eds.), High Integrity Software, Springer US, Boston, MA, 2001, pp. 89–111, http://dx.doi.org/10.1007/978-1-4615-1391-9_5.

[11] V. Buterin, A next-generation smart contract and decentralized application platform, in: White Paper, Vol. 3, No. 37, 2014.

[12] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, Z. Zhang, CreditCoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles, IEEE Trans. Intell. Transp. Syst. 19 (7) (2018) 2204–2220, http://dx.doi.org/10.1109/TITS.2017.2777990.

[13] D. Schwartz, N. Youngs, A. Britto, et al., The ripple protocol consensus algorithm, in: Ripple Labs Inc White Paper, Vol. 5, No. 8, 2014, p. 151.

[14] A. Karinsalo, K. Halunen, Smart contracts for a mobility-as-a-service ecosystem, in: 2018 IEEE International Conference on Software Quality, Reliability and Security Companion, QRS-C, 2018, pp. 135–138, http://dx.doi.org/10.1109/QRS-C.2018.00036.

[15] E. Bothos, B. Magoutas, K. Arnaoutaki, G. Mentzas, Leveraging blockchain for open mobility-as-a-service ecosystems, in: IEEE/WIC/ACM International Conference on Web Intelligence - Companion Volume, in: WI '19 Companion, Association for Computing Machinery, New York, NY, USA, 2019, pp. 292–296, http://dx.doi.org/10.1145/3358695.3361844.

[16] P. Thakkar, S. Nathan, B. Viswanathan, Performance benchmarking and optimizing hyperledger fabric blockchain platform, in: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS, 2018, pp. 264–276, http://dx.doi.org/10.1109/MASCOTS.2018.00034.

[17] C. Melo, F. Oliveira, J. Dantas, J. Araujo, P. Pereira, R. Maciel, P. Maciel, Performance and availability evaluation of the blockchain platform hyperledger fabric, J. Supercomput. 78 (10) (2022) 12505–12527, http://dx.doi.org/10.1007/s11227-022-04361-2.

[18] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, A.V. Vasilakos, Latency performance modeling and analysis for hyperledger fabric blockchain network, Inf. Process. Manage. 58 (1) (2021) 102436, http://dx.doi.org/10.1016/j.ipm.2020.102436.

[19] S. Pongnumkul, C. Siripanpornchana, S. Thajchayapong, Performance analysis of private blockchain platforms in varying workloads, in: 2017 26th International Conference on Computer Communication and Networks, ICCCN, 2017, pp. 1–6, http://dx.doi.org/10.1109/ICCCN.2017.8038517.

[20] T.T.A. Dinh, R. Liu, M. Zhang, G. Chen, B.C. Ooi, J. Wang, Untangling blockchain: A data processing view of blockchain systems, IEEE Trans. Knowl. Data Eng. 30 (7) (2018) 1366–1385, http://dx.doi.org/10.1109/TKDE.2017.2781227.

[21] D.A. Hensher, C.Q. Ho, C. Mulley, J.D. Nelson, G. Smith, Y.Z. Wong, Chapter 2 - what is maas and how it fits into the transport landscape, in: D.A. Hensher, C.Q. Ho, C. Mulley, J.D. Nelson, G. Smith, Y.Z. Wong (Eds.), Understanding Mobility As a Service, MaaS, Elsevier, 2020, pp. 13–33, http://dx.doi.org/10.1016/B978-0-12-820044-5.00002-6.

[22] C. Mulley, Mobility as a Services (MaaS) – does it have critical mass? Transp. Rev. 37 (3) (2017) 247–251, http://dx.doi.org/10.1080/01441647.2017.1280932.

[23] G. Lyons, P. Hammond, K. Mackay, The importance of user perspective in the evolution of MaaS, Transp. Res. A 121 (2019) 22–36, http://dx.doi.org/10.1016/j.tra.2018.12.010.

[24] D.A. Hensher, Future bus transport contracts under a mobility as a service (MaaS) regime in the digital age: Are they likely to change? Transp. Res. A 98 (2017) 86–96, http://dx.doi.org/10.1016/j.tra.2017.02.006.

[25] D.A. Hensher, C.Q. Ho, C. Mulley, J.D. Nelson, G. Smith, Y.Z. Wong, Chapter 3 - global debate and experience with MaaS, in: D.A. Hensher, C.Q. Ho, C. Mulley, J.D. Nelson, G. Smith, Y.Z. Wong (Eds.), Understanding Mobility As a Service, MaaS, Elsevier, 2020, pp. 35–58, http://dx.doi.org/10.1016/B978-0-12-820044-5.00003-8.

[26] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, in: Working Paper, 2008.

[27] R.C. Merkle, A digital signature based on a conventional encryption function, in: C. Pomerance (Ed.), Advances in Cryptology, CRYPTO '87, Springer Berlin Heidelberg, Berlin, Heidelberg, 1988, pp. 369–378.

[28] F. Tschorsch, B. Scheuermann, Bitcoin and beyond: A technical survey on decentralized digital currencies, IEEE Commun. Surv. Tutor. 18 (3) (2016) 2084–2123, http://dx.doi.org/10.1109/COMST.2016.2535718.

[29] G. Wood, et al., Ethereum: A secure decentralised generalised transaction ledger, in: Ethereum Project Yellow Paper, Vol. 151, No. 2014, 2014, pp. 1–32.

[30] Y. Sompolinsky, A. Zohar, Secure high-rate transaction processing in bitcoin, in: R. Böhme, T. Okamoto (Eds.), Financial Cryptography and Data Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 507–527.

[31] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: 2014 USENIX Annual Technical Conference, USENIX ATC 14, USENIX Association, Philadelphia, PA, 2014, pp. 305–319.

[32] B. McHugh, The OpenTripPlanner Project, 2011, pp. 12–16.

[33] M. Conoscenti, A. Vetrò, J.C. De Martin, Blockchain for the Internet of Things: A systematic literature review, in: 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications, AICCSA, 2016, pp. 1–6, http://dx.doi.org/10.1109/AICCSA.2016.7945805.

[34] A. Sheel, V. Nath, Effect of blockchain technology adoption on supply chain adaptability, agility, alignment and performance, Manag. Res. Rev. 42 (12) (2019) 1353–1374, http://dx.doi.org/10.1108/MRR-12-2018-0490.

[35] K. Zhang, H.-A. Jacobsen, Towards dependable, scalable, and pervasive distributed ledgers with blockchains, in: 2018 IEEE 38th International Conference on Distributed Computing Systems, ICDCS, 2018, pp. 1337–1346, http://dx.doi.org/10.1109/ICDCS.2018.00134.

[36] J. Kalajdjieski, M. Raikwar, N. Arsov, G. Velinov, D. Gligoroski, Databases fit for blockchain technology: A complete overview, Blockchain Res. Appl. (2022) 100116, http://dx.doi.org/10.1016/j.bcra.2022.100116, URL https://www.sciencedirect.com/science/article/pii/S2096720922000574.

[37] H. Nguyen, T. Nguyen, T. Leppänen, J. Partala, S. Pirttikangas, Situation awareness for autonomous vehicles using blockchain-based service cooperation, in: X. Franch, G. Poels, F. Gailly, M. Snoeck (Eds.), Advanced Information Systems Engineering, Springer International Publishing, Cham, 2022, pp. 501–516.

[38] T. Nguyen, N. Tran, L. Loven, J. Partala, M.-T. Kechadi, S. Pirttikangas, Privacy-aware blockchain innovation for 6G: Challenges and opportunities, in: 2020 2nd 6G Wireless Summit, 6G SUMMIT, 2020, pp. 1–5, http://dx.doi.org/10.1109/6GSUMMIT49458.2020.9083832.

Tri Nguyen was born in Ho Chi Minh, Vietnam, in 1993. He received the B.Sc. degree in computer science from the University of Information Technology - Vietnam National University, Vietnam in 2015, and the M.Sc. degree in computer science from the University of Pisa, Italy, in 2018. Since 2018, he has been a doctoral student in the Center for Ubiquitous Computing, University of Oulu. His research interests include distributed systems, blockchain technology, and information security.

Huong Nguyen was born in Ha Noi, Vietnam in 1995. Since August 2022, she has been a Ph.D. student at the Center of Ubiquitous Computing (UBICOMP) at the University of Oulu, Oulu, Finland. Before that, Huong completed her M.Sc. degree in Computer Science and Engineering at the University of Oulu in the same year and got a B.S in Computer Science at the Posts and Telecommunications Institute of Technology (PTIT), Ha Noi, Vietnam in 2018. Her main research interests are robot vision, intelligent transportation, Internet of Things, vulnerabilities in distributed systems, and edge computing.

Juha Partala was born in Oulu, Finland in 1980. He received the M.Sc. and D.Sc. (Tech.) degrees in computer science from the University of Oulu, Oulu, Finland in 2005 and 2015, respectively. Since 2015, he has been a postdoctoral researcher in the Center for Machine Vision and Signal Analysis with the Faculty of Information Technology and Electrical Engineering, University of Oulu, Oulu, Finland. He has published in a wide range of topics related to information security such as theoretical and applied cryptography, steganography, security of biomedical systems and blockchain. He serves as a reviewer in top journals in information security. His research interests include cryptography, theory of computation, algebra and security and privacy of biomedical systems.

Susanna Pirttikangas was born in Kemi, Finland in 1973. She received the M.Sc. degree in theoretical mathematics from the University of Oulu, in 1998 and the D.Sc. (tech.) degree in embedded systems in University of Oulu in 2004. Dr. Pirttikangas works as a research director in the Center for Ubiquitous Computing at the University of Oulu. She made her postdoctoral visits to Waseda University, Japan (20052006), Tokyo Denki University, Japan (2008) and Tsinghua University, China (2011). Her research team Interactive Edge develops adaptive, reliable and trusted edge computing. She is an active member of the international research community as a workshop and conference organizer, as well as serving as a reviewer and PC member in top journals and conferences in her field. Dr. Pirttikangas also works as a freelance lead AI scientist in a Finnish company Silo.AI.