HAL open science

An Open Source Solution for Smart Contract-Based Parking Management

Nikolay Buldakov, Timur Khalilev, Salvatore Distefano, Manuel Mazzara

To cite this version:

Nikolay Buldakov, Timur Khalilev, Salvatore Distefano, Manuel Mazzara. An Open Source Solution for Smart Contract-Based Parking Management. 16th IFIP International Conference on Open Source Systems (OSS), May 2020, Innopolis, Russia. pp.55-69, 10.1007/978-3-030-47240-5_6 . hal-03647276

HAL Id: hal-03647276

https://inria.hal.science/hal-03647276v1

Submitted on 20 Apr 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

An Open Source Solution for Smart Contract-based Parking Management

Nikolay Buldakov[1], Timur Khalilev[1], Salvatore Distefano[2] and Manuel Mazzara[1]

[1] Innopolis University, Russian Federation
{n.buldakov, t.khalilev, m.mazzara}@innopolis.ru
[2] University of Messina
sdistefano@unime.it

Abstract This paper discusses an open-source solution for smart-parking in highly urbanized areas. We have conducted interviews with domain experts, defined user stories and proposed a system architecture with a case study. Our solution allows integration of independent owners of parking space into one unified system, that facilitates parking in a smart city. The adoption of such a system raises trust issues among the stakeholders involved in the management of the parking. In order to tackle such issues, we propose a smart contract-based solution encapsulating sensitive information, agreements and processes into transparent and distributed smart contracts.

1 Introduction

It has been estimated that, with the current trend, 60% of people will end up living in urban areas [24]. The increasing pace of the urbanization process made the management of city services more challenging Among such services transportation has strategic importance since the increase of population leads to increased traffic congestion and pollution, which significantly impacts on the citizens' lives.
Several approaches have been adopted to address these issues, and in 1970s some countries decided to widen the roads and create new ones. For example, in Hague, Netherlands, many canals were drained and covered for this purpose [1]. More modern approaches focus on the factors affecting congestion and operate on them. This trend has led to smart management of streetlights [18], and the introduction of payed parking lots and roundabouts.
Several studies have also shown that searching for a free parking stall drastically affects the traffic as vehicles spend some time going around looking for parking [12]. Smart parking solutions can, therefore, significantly improve traffic conditions.
There are several open issues and challenges on the current parking management. First of all, in every district, there are slots of unused land, owned either by private or public entities. Providing easy means of renting out such properties for parking can simplify the process of creating new parking lots. Moreover, collection and aggregation of information about the current occupancy of all parking lots can significantly simplify the process of finding a free stall, thus decreasing traffic congestion.
Another significant issue is the trustworthiness of a potential parking system. With the current solutions on the market, such as [27] sensitive information is stored in a centralised way, which makes it vulnerable and simplifies malicious intrusion.
Finally, various parking providers need aggregation under one universal system, that will guarantee the unbiased treatment of all parking lots. Such strategy will allow potential seekers of parking lots to observe the entire situation in the city instead of seeing only partial information. This work aims at creating an open-source solution in order to tackle the aforementioned problems.

Problem Statement

Every city has governing authorities: city Hall, municipality and it can be divided up into districts with local councils. It can also be organized in terms of smart governance [25]. We will further refer to these authorities as an Administrator. The Administrator is responsible for authorising processes in the city. There are also people and companies possessing pieces of land that are suitable for parking. We refer to them as Landlords. A landlord can be a juridical person or a natural person; it does not matter for the system. A landlord wants to provide their land for parking in order to profit from parking fees. Besides, there are entities called Tenants. A tenant is a company or a person who will to whom a landlord rents out a subset of their parking land-based on some conditions.

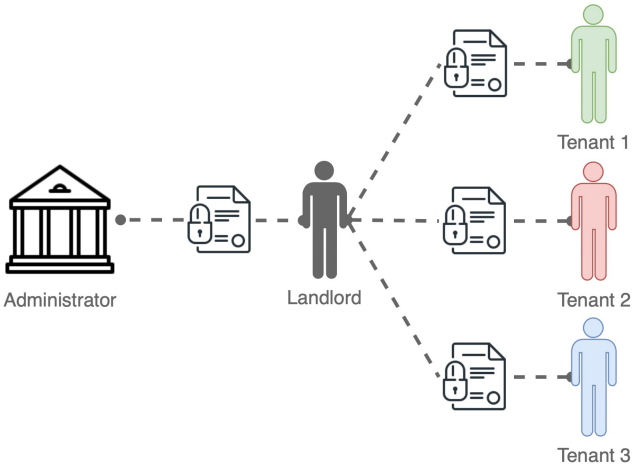

Figure1. Contracts' hierarchy

Now that all the roles are defined, one can observe the relations connection different roles in figure 1. In order to provide their land for parking, a landlord must first sign a contract with the administrator. There is always only one administrator for each area in the city. The process starting from a request to create such contract (from the landlord's side) up until its signing by both parties will be called *landlord registration*. This process involves checking whether the landlord is eligible for renting out their property, and negotiation on the details of the contract. Eventually, the signed contract contains such details as the tax which the landlord guarantees to pay, information about the registered land, duration of the contract and others. Both parties can further amend the contract, given that they are satisfied with the amendment.
The landlord, in turn, can not only provide their land for parking but also they can rent it out to tenants. In this case, the income from parking will not go directly to the landlord, however, they receive guaranteed rent payments and, possibly, a percentage from each parking transaction. Similarly, another contract preserved the agreement between a tenant and a landlord. This contract describes all the details of the rent. Such details are the duration of the contract, the rent fee, the frequency of payments, the landlord's share in each parking transaction and, most importantly, the exact borders of the area rented to each tenant. Again, all these details can be amended later, given the consensus of both parties.

Assumptions about parking lot

There is a set of preconditions, which this study assumes parking lots to hold. First of all, the parking areas have to be divided up into a set of spots. We refer to these spots as *parking stalls*. This partition can be represented by road marking. Moreover, there has to be a tracking device which will make the parking lot fully observable. A particular example can be a camera with an IoT-powered box, produced by Park Smart [2] as described in their patent [26]. This device has to run image processing algorithms in spot in order to tell at each moment which parking stalls are occupied (a particular implementation is provided in [5]) and what the plate numbers of the parked cars are. A visualisation of these assumptions can be found in Figure 2
A real example could be a mall with a big parking zone. Stalls can be partitioned among the businesses that are present in the mall. Big companies, like Ikea could rent the closest stalls in order to ease transportation of goods from the store to cars. Restaurants could want to offer reduced fares to their visitors for parking. There can also be stalls that are not rented by any company. In this case, a car parked there pays according to the pricing policy of the landlord, i.e. the mall.

Proposed solution

The proposed solution presented in this paper aims to tackle the problems described above by using blockchain smart contracts. It assumes a set of roles to be present is the city. One of them is a city hall, that will be responsible for the deployment of the system as

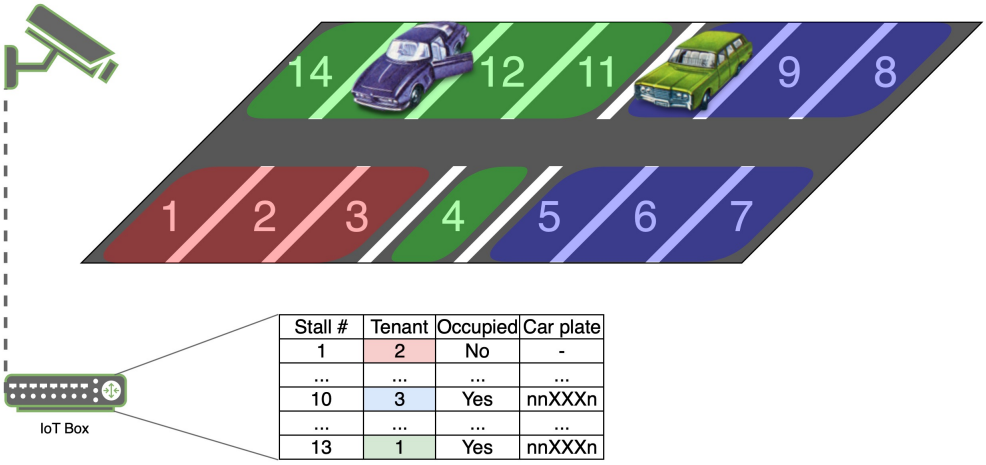

Figure2. Example of the assumed parking lot

well as setting several policies following the local legislation. Another role is a landlord who possesses a piece of land. A landlord can be a juridical person or a natural person, it does not matter for the system as long as their land satisfies the constraints describer in the Assumptions section. Such situations are common: a landlord does not want to maintain the land fully themselves and rents its parts out, in this case the proposed system allows a landlord to partition their land as they desire and set complex paying policies, that will ensure the agreed-upon conditions at the time of payment for parking. The last role is a driver interested in parking their car. They can observe free spots available at every parking lot and the pricing policy and choose the most suitable one. Payment is also covered conducted through the system, using a micro-payment channel, that makes it possible to enforce and guarantee the distribution of money among the landlords, the city hall and the tenants.

2 Related Work

By the time of writing, various smart parking systems exist in the literature, that focus on various aspects of parking. In principle, we can divide them into two groups: centralized solutions and decentralised solutions.

Centralized solutions

There is a garden variety of solutions present both in the market and research, which attempt to prove a framework to manage smart parking. However, the major drawback of most of them is their centralised nature. Many of them rely on a particular supreme entity, their "source of truth". In [17] a parking system is proposed that relies on IoT devices and exchanges data by means of TCP/IP protocol. However, the application resides on only one server, that makes it vulnerable to issues with security, trustability and single point of failure.

Decentralized solutions

Other works propose architectures that utilise the blockchain solutions as their backbone. All centralised solutions share a common set of problems that make them undesirable for practical use. The most important of such issues is trustworthiness. All participants of such systems have to rely on a particular entity, that will possess the power of accessing, chaining and removing their data, with no guarantee at the time of signing an agreement or registering that the owner of the system will obey to the promised rules. Lack of transparency is another significant shortcoming. When the implementation is not exposed to the end-users, they cannot validate that a particular solution is guaranteed to function in accordance with their needs and that it does not have any back- doors. Finally, considering the ongoing competition

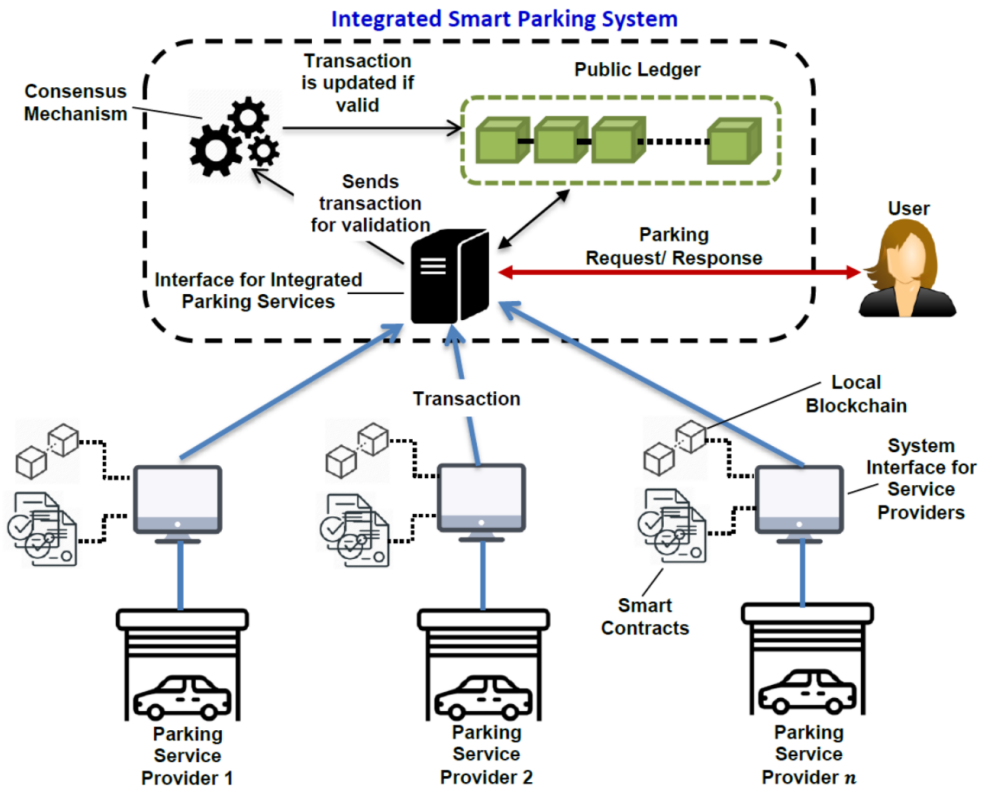

Figure3. Overview of Integrated Smart Parking System, taken from [6]

on any market, including parking, a centralised solution can be unfair to some participants if the owner is biased towards some companies. In [6] a system is proposed for incorporating blockchain into regular parking scenarios. This paper presents a layered architecture that standardizes the typical architecture of an integrated smart parking system. It also describes the major components of the system as well as provides an overview of possible work-flows of the solution. In this system, there are three major participants: parking service provider, blockchain network, and user. The network enjoys the standard blockchain characteristics: a consensus algorithm, a shared public ledger, running a local copy of the ledger at each parking area. The overview schema of the system can be seen in 3. The architecture in the described solution consists of four layers: application layer, network layer, transaction layer, and physical layer. This layout of technologies can be considered the current state of the art and is adapted in the solution, proposed in this thesis work. Finally, the authors of the paper describe two main workflows supported by their system: 1. Search and Request of a parking spot and 2. Parking Provider Operations in the system. Even though the paper describes a useful basic concept, it supports a minimal number of scenarios, and thus, one cannot consider it as realistic.

Another blockchain-powered system is presented in [15]. It aims to take into account the process of setting up the parking space for the land-owner. For this they a landlord has to go through government validated process and procedures such as Land Registry and Government mandate external system. The external control is conducted by presenting another entity, named Oracle. In addition to that, they tackle the problem of leasing lands to the so-called contractors by distributing non-fungible tokens among them for a particular period. These tokens are issued to the certified landlords by the government. The critical advantage of this solution is that it describes in great details peculiarities related to how governing authorities and lease of land can be integrated into a blockchain system. However, many processes are supposed to happen off-chain, which is a serious limitation for the true decentralisation of the system.

[16] presents registration, search, rent, payment modules built by the means of the BCOS smart contracts framework. The main novelty of this system is the use of one-time pad mechanism and a group-signature mechanism together with a union of privileged nodes in order to conceal the details of each transaction and thus preserve users' privacy while the

privileged nodes can still access users' data. One-time padding means that instead of receiving a real address for each transaction, users, instead, receive a newly generated address each time. This way, the users' addresses are not exposed. A group signature mechanism allows signing a transaction on behalf of a group, which hides the mapping between transactions and users. Other concepts implemented in this system are relatively similar to the works mentioned above, so the privacy mechanisms are the main beneficial feature of the paper.
Finally [11] introduces a gamification approach that simplifies the process of finding a parking lot. It encourages users to take part in the reporting process, by using the data history in order to calculate which places are more likely to be empty and by providing a framework that does not assume or require any particular installation of infrastructure. In this system, the concept of mobile crowdsensing plays a pivotal role.In order to ensure the involvement of drivers, the system assigns individual points to each driver for their collaboration. Examples of such collaborations are the installation of a special beacon or reporting on the number of free parking spaces in the street. Pedestrians can also participate by reporting through a mobile app. These points can be exchanged for free parking minutes. The authors claim their approach to bring a variety of benefits including a faster search for parking (leading to lower CO2 emissions), easy deployment of the solution as it does not require much an existing infrastructure. Although their reasoning is valid, there is almost no information in the paper how the blockchain technology was incorporated.

Discussion

Although the works mentioned in this literature review present a variety of parking solutions, a significant limitation is in the fact that all of them lack flexibility for the end users. There is a variety of possible configurations of the system present, for example with a governing agency involved [15] or without [6]. However, none of them allows the community to choose what type of configuration they need. The same issue arises with the pricing policy. All papers consider some pricing policy present and even fluctuating depending on some factors as time and demand [6]. However, the details are omitted of how exactly the pricing is organized as well as how shares are distributed among different entities in the system. Finally, the topic of leasing or renting out the land is not thoroughly defined, in particular, how exactly it can be done through the system. In [15] a token system is proposed but the details of how exactly the partitioning of a land will be tackled in the system are not presented.

3 System Design

The objective of this work is to take into account drawbacks of the current centralised parking solutions, as well as limitations of the decentralised parking systems and to design and build a system that would address those issues. The system will utilise smart contracts and take into consideration some realistic requirements gathered by interviewing people from the industry.

Interviewing Industry representatives

Any proper system-development commences with gathering important requirements of the system. Furthermore, as [13] states "one might use questionnaires or interviews to explore what problems members see as most important, how members place themselves in various classification schemes, etc." Thus, an interview has been conducted with a representative of Park Smart [2]. Park Smart is a smart parking solution company founded in Italy, Messina in 2014 and its main aim is to implement a software-based technology able to analyse, in real-time, the availability of space in in-street parking. The company has already been in the field for a few years. Therefore, was valuable to question them about their experience and to learn what are the main requirements for a parking system.
The interview has shown some essential aspects of the modern smart parking system and it has also revealed some common shortcomings of the existing system. For instance, if a company wants to use this solution for paid parking, they can provide their end point to Park Smart in order to receive money through the system. This approach has inspired the idea of service providers in our system. Moreover, the interviewee has confirmed our concern about the centralisation of current systems. Park Smart itself has a patent about blockchain architecture for their system, however, it has not been implemented yet. Besides, the importance of flexible pricing policies has been highlighted. Companies have different rates at a different time of the day, therefore, this part of functionally is to be flexible. Finally, we found out what particular information about users and processes is needed for such a system, and this has affected our decisions of what to store in the proposed solution.

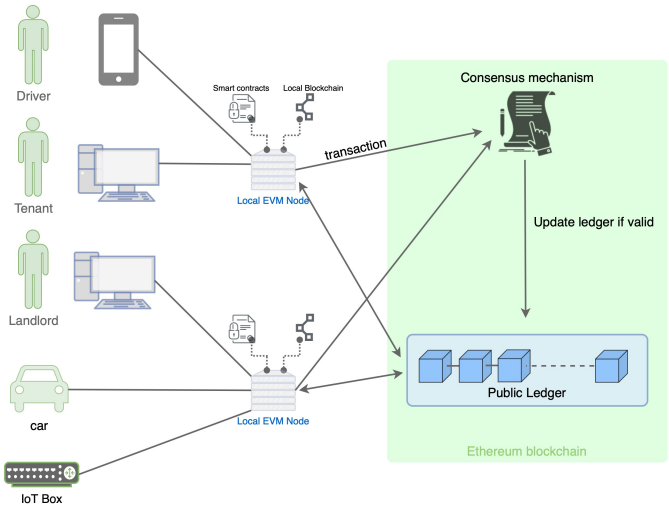

Figure4. System Overview

Product backlog

In order to comprehend the needs of various user roles in the system, user stories have been gathered. The roles that we considered are the following: *Administrator*, *Landlord*, *Tenant*, *Driver*, and *Service Provider*.
The user stories have revealed a particular set of features that every role needed. Here we extract the key functionality from the user stories. First of all, both driver and parking providers (landlords and tenants) require a safe, trustworthy way of conducting payments in such a way that both parties are guaranteed that the amount of funds transferred is equivalent to the provided service. Besides, landlords and tenants want to ensure that both parties respect the renting contracts they make among them and that such contracts can be extended or amended over time. Parking providers also want the system to postulate equal treatment for all parking lots so that there are no privileged ones, promoted by that system. Finally, the administrative authority requires to have privileged over the system (bounded by conditions of contracts made with other parties).
In total, the backlog is constituted by more than 30 user stories, which were considered during the design and implementation phases.

System Design Overview

There are several participants in the system: *landlord*, *tenant*, *driver or car*, *IoT box* installed in each parking system, *administrator* and as a governing authority. The parking system consists of thin clients (such as a mobile or a web app), and each participant can access their part of functionality through the thin client. Every thin client is connected to a local EVM (Etherium Virtual Machine) node, which in turn is connected to the rest of the Etherium network. The local node has an instance of the ledger, shared by all nodes, and also hosts smart contracts.
Whenever a participant wants to perform an action (e.g. initiate parking or update number of free parking sports), the thin client tells the EVM to invoke a corresponding method of some smart contract. This action is performed in the form of a transaction on the blockchain. The blockchain network contains a public ledger and updates the public ledger with the valid transactions only. A consensus mechanism is used to verify the transactions. Once the new transaction is verified, the public ledger is updated, and the local copies of the ledger in each node are synchronised and all participants can observe the new state of the parking system. This overview is visualised in 4

Smart contracts

Smart contracts are the back-end part of the system. The logic is distributed among several contracts, which correspond to logical components of the system. Each contract encapsulates its sensitive data and controls access to its functions. This way, only authorised entities can perform actions in the system, even though the contracts are publicly exposed. Actors of the

system deploy some of the contracts, other contracts are deployed automatically in existing contracts. The decision of whether a contract is deployed from the outside of the system or within contracts depends on two questions:

1. Is the nature of the contract static or dynamic?
2. Should the system validate and control the deployment process of a contract?

The nature of a contract matters because some contracts are deployed once and remain active in the system for a long time, whereas others are needed only for a short amount of time, and after their expiration, they cannot be used inside the system anymore. An example of a static contract could be the Parking System contract, which is deployed by the authorities and the life cycle of this contract defines the life cycle of the system itself. On the other hand, an example of a contract of a more dynamic nature could be a Payment Channel. Whenever a new parking process commences, the system automatically generates a payment channel. Such a contract has a short lifetime and cannot be reused. Thus the burden of its creation lies upon the shoulders of a static contract – Parking Provider.

The latter question tackles the problem of validation. Contracts can be inherited, and their behaviour can be changed by other parties. For example, the Parking Lot contract can be modified to surpass the tax obligation. That is why one requests the system to deploy such a contract. The system verifies all information related to the contract and only after that deploys it. For other contracts, on the contrary, extensions are allowed. For instance, Payment Policy defines a set of methods to be implemented to calculate the price of particular parking, however, it is up to a parking provider how exactly the price will be calculated. Thus, a parking provider is entitled to extend and deploy their payment policy and supply the address of this policy to the system.

Parking System

Parking System is the first contract to be deployed to launch the system. It is deployed by an administrative authority (e.g. a city hall) and it implements the main managing functionality of the system. The parking system contract is responsible for the registration of new parking lots and cars as well as for storing addresses of all other entities in the system. Whenever some request cannot be fulfilled automatically by the contract, it saves the request and pends administration's approval. Moreover, this contact provides a means of control and management of the whole system. Thus, the system cannot exists without this contract.

Parking Provider

Parking provider is an abstract contract that cannot be used on its own but should be extended in order to be used. It implements the logic of the parking process and stores information about what parking stalls it possesses and what cars are parked at the stalls. In other words, whenever a car wants to park, the code of this contract is used. In order to park a car sends funds to this contract. It also specifies a stall number and till what time it wants to be parked. The contract calculates the price, checks if the funds sent are sufficient and whether a parking stall is free and after that creates a payment channel. Optionally parking provider can also have a set a service providers, in this case a car can specify which service provider it is using and a fraction of the payments will go to the service provider.

Car

This contract preserves general information about a car. This information includes its plate number, account address of its owner and the rating of the car in the system. In addition to that, it stores information about parking. That includes whether the car is parked now or not, the address of the payment channel, parking history. If one needs some information about a car, this is the contract to address.

Parking Lot

Parking lot extends Parking Provider contract and is deployed by the parking system contracts at a request of a landlord and with approval of the administrator. This contract adds functionality to partition the parking lot among several tenants. In order to do so, a tenant creates a request to this contract. If the landlord approves the request, the contract creates a renting contract between the landlord and the tenant. It also contains general information about the parking lot, such as its rating, address and so on.

Tenant

Tenant also extends Parking Provider contract. Unlike Parking Lot contract, this contract does not need permission from the system to be deployed as it interacts only with a parking lot. A tenant can set its payment policy and it also stores the address of its contract with a parking lot.

Service Provider

Conceptually a service provider is a third-party entity that collaborates with a parking provider in order to place information about the parking provider's stall on their platform. Thus, the contract only facilitates the interaction between a service provider and a parking provider. It contains information about a service provider and about parking provider with whom it collaborates.

Renting Contract

Renting contract implements a relationship between a parking lot and a tenant in such a way that both parties are guaranteed that their agreement will be respected in the system. The contract is deployed when the parking lot contract approves registration of a new tenant. After that, the tenant reviews the contract and if conditions are satisfactory, the renting process begins. The contract also contains logic for penalties in case of delayed payment. Finally, it supports the introduction of amendments from both parties. In the case of mutual agreement, such amendments can modify the behaviour of the contract.

Payment Channel

Payment channel implements the logic for micropayments between two parties, in such a way that both parties are guaranteed to receive the promised funds. One party opens the contract and sends funds to it. After that, the funds are locked in the contract, and neither of the parties can access them. Micropayments are conducted off-chain, therefore, they are not part of this contract. Once the payment process is completed, the receiving party sends the last encrypted message to the contract, and the contract uses encryption algorithms to verify that the party is entitled to receive the funds. If that is the case, funds are sent to the receiver's address. The detailed description of the principles behind this contract can be found in 3

Payment Policy

Payment policy is a protocol that defines two methods: get payment rate at a particular time and get a total price for parking given the start and the end time. It is up to parking providers to define their parking policies, depending on their business goals. A particular implementation of this protocol is provided. It implements a policy that can have a specific rate depending on an hour and a day of the week.

Parking system front-end

The front-end part of the parking system is a distributed application in the form of web, mobile or embedded software installed on mobile phones and computers of the Administrator, landlords, tenants and drivers as well as in IoT boxes at parking lots. In future they will be referred to as think clients. Every thin client provides a user interface or an API for interaction with the system. User interfaces are used by landlords, administrator, tenants and drivers. The API is used by the IoT box or a smart car. The front-end part is responsible for translating users' actions into requests to the smart contracts.

Communication between the smart contracts and the parking system

In order for the front-end distributed application to communicate with the EVM, there has to be a specific data-interchange format. This format is called JSON-RPC [14]. It is utilised through a JavaScript library called Web3.js [1], which allows us to interact with a local or remote Ethereum node using a HTTP or IPC connection [4]. As depicted in Figure 5 Every front-end module of the distributed system communicates mostly with their own smart contract and this contract, in turn, is responsible for the interaction with the Parking System module and other modules.

[1] https://github.com/ethereum/web3.js/

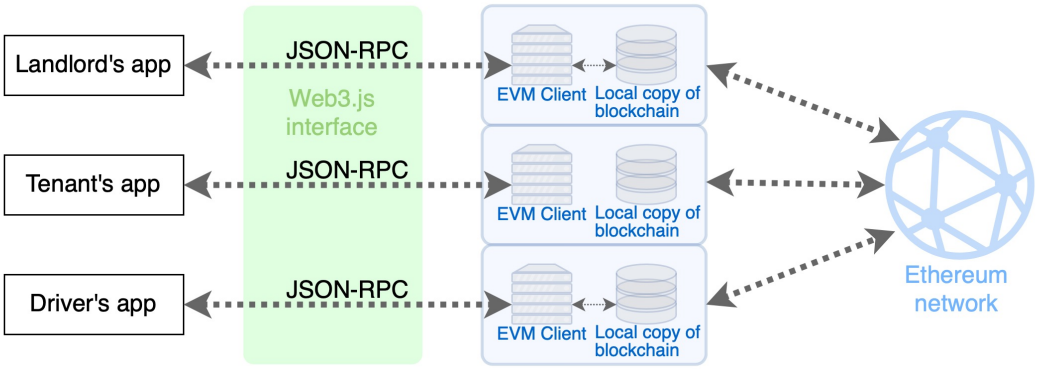

Figure5. Communication with the blockchain

Micro-payment channel

Whenever a car gets parked at a parking lot, there is a mutual agreement between the parking lot and the car: the car receives a service and transfers money for parking. In a traditional system, the payment is conducted in advance for a particular period, [19]. Such system lacks trustworthiness, as a car has to blindly trust the parking provider, that the service will be provided sufficiently and will not be interrupted earlier, the parking provider, in turn, has to ensure that the car is not parked for longer than the paid period, this requires additional control that brings an additional cost. Finally, parking is not a discrete process, unlike the pricing policy in traditional parking systems, which results in drivers oftentimes paying more than they should.
This problem can be solved by implementing a micro-payment channel through a smart contract and direct signed messages. The high-level picture of such payment includes three steps:

1. Driver deploys a contract with locked funds of some amount that exceeds any possible parking period.
2. Driver emits messages directly to the service provider. The messages are the "micro-transactions" containing how the amount is transferred. The parking lot can check the validity of the messages as they are signed by car.
3. After the parking process is over, the service provider can use the last message to retrieve the claimed amount of funds from the contract. The remained of the locked funds are sent back to the car.

It is essential to mention that in this scenario only steps 1 and 3 happen on the blockchain and the most intense process of micro-payments is conducted off-chain through a peer-to-peer communication [3]. Thus, the parties do not have to conduct costly blockchain transactions (gas in Etherium terminology [8]) for each payment. Only two transactions are required: to deploy the contract (by the car) and to claim the end of the payment period (by the parking lot). Moreover, the transaction fee will be paid by the service provider, thus removing any additional fees from the car. This process is visualised in 6

4 Conclusions

A significant amount of research has been conducted on smart parking. In this paper, we have examined some of the recent work related to this topic and propose a solution. Interviews have been conducted with domain experts, user stories defined and a system architecture has been proposed with a case study. Our solution allows independent owners of parking space to be integrated into one unified system, that facilitates the parking situation in a smart city. The utilization of such a system raises the issues of trust and transparency among several actors of the parking process. In order to tackle those, we propose a smart contract-based solution, that brings in trust by encapsulating sensitive relations and processes into transparent and distributed smart contracts. From the architecture point of view services and, in particular, microservices [9,22,7] and their composition, orchestration [20,23], reconfiguration [21] and recovery [10] have not been discussed. All these are open issues that need to be investigated in future.

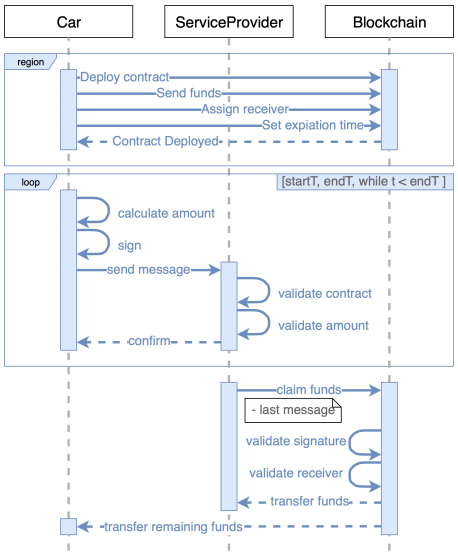

Figure6. Sequence diagram of micro-payments over blockchain

References

1. Mvrdv unveils plans to reopen the hague's forgotten canals. https://www.designboom.com/architecture/mvrdv-hague-canals-reopen-09-19-2019/. Retrieved 2019-09-19.
2. Parksmart home page. https://www.parksmart.it. Retrieved 2019-09-30.
3. The peerjs library documentation. https://docs.peerjs.com. Retrieved 2019-12-01.
4. Web3.js documentation. https://web3js.readthedocs.io/en/v1.2.4/. Retrieved 2019-09-30.
5. Debaditya Acharya, Weilin Yan, and Kourosh Khoshelham. Real-time image-based parking occupancy detection using deep learning. In *Research@ Locate*, pages 33–40, 2018.
6. Sabbir Ahmed, Mohammad Saidur Rahman, Mohammad Saiedur Rahaman, et al. A blockchain-based architecture for integrated smart parking systems. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 177–182. IEEE, 2019.
7. Antonio Bucchiarone, Nicola Dragoni, Schahram Dustdar, Patricia Lago, Manuel Mazzara, Victor Rivera, and Andrey Sadovykh, editors. *Microservices, Science and Engineering*. Springer, 2020.
8. Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.
9. Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch-Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering*. Springer, 2017.
10. Nicola Dragoni and Manuel Mazzara. A formal semantics for the ws-bpel recovery framework. In Cosimo Laneve and Jianwen Su, editors, *Web Services and Formal Methods*, pages 92–109, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
11. João Carlos Ferreira, Ana Lúcia Martins, Frederica Gonçalves, and Rui Maia. A blockchain and gamification approach for smart parking. In *First International Conference on Intelligent Transport Systems*, pages 3–14. Springer, 2018.
12. Tullio Giuffrè, Sabato Marco Siniscalchi, and Giovanni Tesoriere. A novel architecture of parking management for smart cities. *Procedia-Social and Behavioral Sciences*, 53:16–28, 2012.
13. Joseph A Goguen and Charlotte Linde. Techniques for requirements elicitation. In *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 152–164. IEEE, 1993.
14. JSON-RPC Working Group et al. Json-rpc 2.0 specification, 2013.
15. Jennath Hassan Sabeela, Nikhil V Chandran, et al. Parkchain: A blockchain powered parking solution for smart cities. *Frontiers in Blockchain*, 2:6, 2019.
16. Jiaxi Hu, Debiao He, Qinglan Zhao, and Kim-Kwang Raymond Choo. Parking management: A blockchain-based privacy-preserving system. *IEEE Consumer Electronics Magazine*, 8(4):45–49, 2019.

17. Abhirup Khanna and Rishi Anand. Iot based smart parking system. In *2016 International Conference on Internet of Things and Applications (IOTA)*, pages 266–270. IEEE, 2016.
18. Sei Ping Lau, Geoff V Merrett, Alex S Weddell, and Neil M White. A traffic-aware street lighting scheme for smart cities using autonomous networked sensors. *Computers & Electrical Engineering*, 45:192–207, 2015.
19. Todd Litman. Parking pricing implementation guidelines how more efficient parking pricing can help solve parking and traffic problems, increase revenue, and achieve other planning objectives. 2018.
20. Manuel Mazzara. *Towards Abstractions for Web Services Composition.* Ph.D. thesis, University of Bologna, 2006.
21. Manuel Mazzara, Faisal Abouzaid, Nicola Dragoni, and Anirban Bhattacharyya. Design, modelling and analysis of a workflow reconfiguration. In *International Workshop on Petri Nets and Software Engineering*, pages 10–24, 2011.
22. Manuel Mazzara, Antonio Bucchiarone, Nicola Dragoni, and Victor Rivera. Size matters: Microservices research and applications. In *Microservices, Science and Engineering*, pages 29–42. 2020.
23. Manuel Mazzara and Sergio Govoni. *A Case Study of Web Services Orchestration*, pages 1–16. Springer Berlin Heidelberg, 2005.
24. Gordon McGranahan and David Satterthwaite. *Urbanisation concepts and trends.* Iied London, 2014.
25. Albert Meijer and Manuel Pedro Rodríguez Bolívar. Governing the smart city: a review of the literature on smart urban governance. *international review of administrative sciences*, 82(2):392–408, 2016.
26. Billy Jay Smart et al. Allocating an area to a vehicle, November 20 2014. US Patent App. 14/344,947.
27. Jihoon Yang, Jorge Portilla, and Teresa Riesgo. Smart parking service based on wireless sensor networks. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 6029–6034. IEEE, 2012.