Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

A Distributed Ledger based infrastructure for Intelligent Transportation Systems

Relatore:
Chiar.mo Prof.
Stefano Ferretti

Presentata da:
Mirko Zichichi

Sessione I
Anno Accademico 2018/2019

Sommario

In questo lavoro viene presentata un'infrastruttura per Sistemi di Trasporto Intelligenti (ITS) basata sull'uso di Distributed Ledger Technologies (DLTs). Questi tipi di sistemi sono nati per migliorare l'efficienza dei sistemi di trasporto, l'incolumità in viaggio, la sicurezza dei veicoli ed il numero di scelte date a guidatori e passeggeri. Questi obbiettivi possono essere raggiunti grazie all'uso di procedure, sistemi e devices che permettono la raccolta, la comunicazione, l'analisi e la distribuzione di dati tra individui e veicoli in movimento, infrastrutture e servizi. In questo lavoro, una serie di applicazioni e protocolli sono stati utilizzati in comunione per ottenere un uso "avanzato" delle reti di trasporto alle quali siamo abituati. Ciò avviene senza l'ausilio di "intelligenza umana" perché le interazioni tra utenti nel sistema, o tra utente e veicolo o infrastruttura, include il meno possibile la presenza di intermediari fisici e viene portata a termine tramite l'ausilio di intelligenze artificiali, quando chiunque può avere fiducia in queste. Questi intermediari artificiali portano alla creazione di servizi innovativi che sono avvantaggiati da un processamento più veloce e da prestazioni migliori.
Le DLTs propongono un nuovo tipo di fiducia, rimuovendo il bisogno di terze parti e migliorando le transazioni tra utenti. In questo lavoro una infrastruttura decentralizzata viene presentata in termini di un ecosistema dove diversi ambienti interagiscono tra di loro per ottenere due servizi per gli utenti: Data Sharing e Smart Services. La visione dell'infrastruttura viene mostrata attenzionando questi due concetti in cui, il Data Sharing permette ad ogni utente di distribuire in maniera decentralizzata i suoi dati prodotti, mentre gli Smart Services sono servizi nei ITS basati su una computazione distribuita. L'idea principale dell'ecosistema è accompagnata da alcuni requisiti funzionali e non funzionali ed da alcuni scenari, in modo da analizzare il problema da risolvere. Il design dell'ecosistema è composto da quattro ambienti che includono diverse componenti inerenti a DLTs ed altre architetture. Un ambiente è dedicato all'acquisizione e archiviazione dei dati, che consiste nella comunicazione tra AU, ovvero il device personale dell'utente, e i veicoli permettendo di ottenere dati da salvare nella Tangle di IOTA o su IPFS. L'accesso ai dati è fornito, in un altro ambiente, da un Server di Autenticazione ad ogni utente che possiede i diritti di accesso, ottenuti attraverso l'uso degli Smart Contracts. Un terzo ambiente racchiude in sé una Second Layer Trust necessaria a validare la veridicità dei dati dell'utente, all'interno del quale vengono prodotti e rilasciati certificati. Infine gli Smart Services rappresentano un intero ambiente che comprende più componenti all'interno dell'ecosistema.
L'implementazione dell'ecosistema viene proposta focalizzandosi su ogni ambiente e sull'analisi dei componenti principali e delle loro operazioni. I sensori sono i primi componenti che agiscono all'interno dell'infrastruttura, dato che sono posti nei veicoli e negli AU e registrano i dati che l'utente può condividere. La Tangle di IOTA è usata per archiviare e validare i dati degli utenti attraverso l'uso di canali di Masked Authenticated Messaging (MAM). In questo processo l'applicazione dell'AU

agisce come un wallet che comunica con full nodes di IOTA per "attaccare" una transazione sulla Tangle. Quando i dati sono troppo grandi vengono salvati su IPFS e referenziati nei canali MAM. La blockchain di Ethereum concretizza la business logic dell'infrastruttura attraverso l'uso degli Smart Contracts. Alcuni contratti sono utilizzati per gestire l'accesso ai dati e la comunicazione con gli Smart Services, mentre altri sono usati all'interno dell'ecosistema per creare una moneta unica e per permettere pagamenti off-chain. Il Server di Autenticazione fornisce l'accesso ai dati agli aventi diritto rilasciandogli le chiavi d'accesso, mentre i certificati, rilasciati da devices fidati, permettono la correttezza di questi dati. Gli Smart Services vengono implementati tramite Smart Contracts gestiti direttamente dal fornitore dei servizi oppure consistono in servizi di trasporto standard che sfruttano la blockchain per eseguire transazioni senza bisogno di intermediari.
Infine, una serie di test inerenti alle prestazioni di IOTA vengono eseguiti per validare l'infrastruttura, mettendo a confronto transazioni standard con transazioni MAM e considerando diversi providers e devices. I risultati mostrano come trovare delle "tips" da referenziare durante il processo di attaccamento alla Tangle influenza maggiormente la latenza rispetto all'operazione di Proof of Work. Ciò implica che scegliere il giusto provider è fondamentale. Considerando le promesse di scalabilità di IOTA, viene anche mostrato come l'utilizzo di canali MAM per i ITS sia un task realizzabile.

Abstract

Intelligent Transportation Systems (ITS) are proposed as an efficient way to improve performances in transportation systems applying information, communication, and sensor technologies to vehicles and transportation infrastructures. The revolutionary growth in both research and industry that Intelligent Vehicles are experiencing is fundamental in the context of ITS, but still, some drawbacks arise in secure communication, trust and data accuracy. The great amount of vehicles produced data, indeed, can potentially lead to a revolution in ITS development, making them more powerful multifunctional systems.
To this purpose, the use of Vehicular Ad-hoc Networks (VANETs), classified as an application of Mobile Ad-hoc Networks (MANETs) for transportation, can provide comfort and security to drivers through reliable communications. Meanwhile, distributed ledgers in the form of blockchains have emerged in recent years radically evolving the way that we used to consider the finance, trust in communication and even renewing the concept of data sharing. Blockchain can be used to establish an autonomous, secured, trusted and decentralized system, hence being suitable to evolve the ITS infrastructure.
In this work an ITS infrastructure based on the combination of different emerging Distributed Ledger Technologies (DLTs) and VANETs is proposed, resulting in a transparent, self-managed and self-regulated system, that is not fully managed by a central authority. The intended design is focused on the user ability to use any type of DLT-based application and to transact using Smart Contracts, but also on the access control and verification over user's vehicle produced data. Users "smart" transactions are achieved thanks to the Ethereum blockchain, widely used for distributed trusted computation, whilst data sharing and data access is possible thanks to the use of IOTA, a DLT fully designed to operate in the Internet of Things landscape, and IPFS, a protocol and a network that allows to work in a distributed file system. Different approaches can be used to achieve a DLT-based ITS, especially regarding the use of new permissioned blockchains that enhance some useful aspects in VANETs, but the aim of this thesis is to create a ready-to-work infrastructure based on the hypothesis that every user in the ITS must be able to participate. To evaluate the proposal, an infrastructure implementation is used in different real world use cases, common in Smart Cities and related to the ITS, and performance measurements are carried out for DLTs used. Finally the architectural challenges will be shown, together with the distributed ledgers limits and how important concepts, such as privacy, are managed in these technologies.

Keywords – Intelligent Transportation Systems, Distributed Ledgers, Internet of Things, Smart Cities, Security, Ethereum, IOTA, IPFS

Contents

List of Figures

1 Introduction

In the last decades Intelligent Transportation Systems (ITS) have emerged as a way to improve transportation systems efficiency, travel safety, vehicle security and the choices given to drivers and passengers. These objectives can be accomplished through the use of procedures, systems and devices that allow data gathering, communication, analysis and distribution among moving individuals and vehicles, infrastructures and services. Intelligent Transportation Systems are defined in the European Union directive 2010/40/EU as "advanced applications which without embodying intelligence as such aim to provide innovative services relating to different modes of transport and traffic management and enable various users to be better informed and make safer, more coordinated and 'smarter' use of transport networks". Paraphrasing this definition, a series of key concepts explain causes and effects that lead to the realization of this thesis work. "Advanced applications" captures the most general definition of the infrastructure intended to bring in this work. A variety of applications and protocols are used in conjunction to obtain an advanced use respect to what we are used to for the traditional transportation network. The interpretation of "without embodying intelligence" brings out the real essence of an infrastructure of such type. The interaction process between two users or an user and a vehicle or infrastructure must include the least possible the presence of a human intermediary. The human intelligence must be changed in artificial intelligence, when this one can be fully trusted by anyone. This artificial intromission leads to the creation of "innovative services" that take advantage of faster processing and better performances. When there are no intermediaries, traditional processes become lighter, hence, by the combination of these lighter processes, new innovations may take place in what is considered a standard. Finally the concept "smarter use" gives an assist to present the other major subject of this thesis, because the adjective "smart" assumes a completely specific meaning in this context.
The blockchain, presented for the first time with Bitcoin [6], has changed radically the vision that we have of finance, trust in communication and even renewed the concept of contracts and digital democracy with Ethereum [4]. The decentralized computation enabled by the blockchain allows us to create self-managed structures that do not rely on a central control, avoiding the possibility of a single point of failure. Blockchain is already used for the Internet of Things [7] and Smart Cities [8][9], hence different works towards a blockchain-based ITS have already been made but also start-ups built around these concepts, such as DOVU [10] and CHORUS Mobility [11]. Especially worth to notice is MOBI [12], a consortium that promotes standards and accelerates the adoption of DLTs and related technologies in the mobility industry. The blockchain is a kind of Distributed Ledger Technology (DLT) between the many that are rising in recent years. The main scope of a DLT is to move trust from the intermediary that manages a transaction between two parties to a protocol that allows a transaction without the need of a third party. DLTs vary in what they provide and what they lack: for example Ethereum provides a distributed

Virtual Machine able to process any kind of computation but with constraints in scalability, while IOTA ledger [13] offers a more scalable structure with no possibility of distributed computation.
The aim of this work is to use DLTs to propose an infrastructure for the ITS. Recalling the key concepts that compose the definition of ITS, in this work two main views will be followed: Data Sharing and Smart Services. These two concepts do not form a partition, but they are combined in their use in order to exploit at full capacity the ITS. For Data Sharing will be defined how the data will be shared, but also how and from what will be acquired and the technologies used for these processes. As it is easy to think, DLTs play a major role in this case, but other technologies, such as IPFS [14], will be used to achieve different tasks. Smart Services embody completely the definition of ITS given above, being provided directly by users or devices thanks to the use of Ethereum Smart Contracts. In this context, VANET is taken as reference to provide a communication landscape in this work and other mechanisms and algorithms are used to provide standard requirements, such as Zero Knowledge Proof for privacy.
This thesis work is organized in six chapters where the entire infrastructure is explained starting from the core idea finishing to the discussion regarding the implementation. Chapter 2 is dedicated to the Background, where all the technologies used are presented with their structure and functions. Chapter 3 covers the Vision of the infrastructure, in which the main idea is showed together with system requirements and some scenarios. In Chapter 4 the infrastructure Design is presented, giving a definition of ecosystem that embraces different environments, representing components that allows Data Sharing and Smart Services. Based on the design, Chapter 5 is dedicated to the Implementation, where the environments components are examined in detail and technologies used are depicted. Chapter 6 presents the infrastructure Validation, in which performance tests and real world use cases are used to assess this work and finally in Chapter 7 a Discussion on remaining issues and concerns will be held.

2 Background

Year by year Intelligent Vehicles and Intelligent Transportation Systems seem to rapidly improve drivers comfort and security. However, emerging applications and services pose new challenges that require substantial changes in vehicular network models. Vehicular ad hoc networks (VANETs) are classified as an application of mobile ad hoc network (MANET) with the potential of improving road safety and travelers comfort [15]. In VANET driver properties depend on the communication between vehicles and infrastructure. This communication allows vehicles to share different kinds of information, such as safety data, vehicle data or traffic jams. Existing VANETs encounter many security issues in the dissemination of this data with malicious vehicles, hence blockchains can be used to resolve it.
Blockchain has been introduced firstly by Satoshi Nakamoto (fictional name) with his whitepaper on Bitcoin [6]. It serves as a base to allow the existence of Bitcoin as a decentralized cryptocurrency. Cryptocurrencies are a digital goods used as medium of exchange that make use of cryptography to secure transactions. They are considered as an alternative to traditional currencies since their control is decentralized and not bonded to banks. Bitcoin use the blockchain because it is a distributed computation architecture that allows to maintain an immutable ledger of transactions through a combination of cryptography algorithms that assure security against malicious third parts. The blockchain underlying Bitcoin is considered as the 1.0 version and multiple projects in the form of "alt coins" followed its path and enhanced its protocol. One for all is Ethereum that with its blockchain 2.0 led the way to the creation of decentralized application through Smart Contracts. Since the blockchain introduction, numerous alternatives have been presented as distributed ledgers. For instance a Direct Acyclic Graph is used in IOTA as a ledger, allowing the process of validation to be faster and feasible to IoT devices.

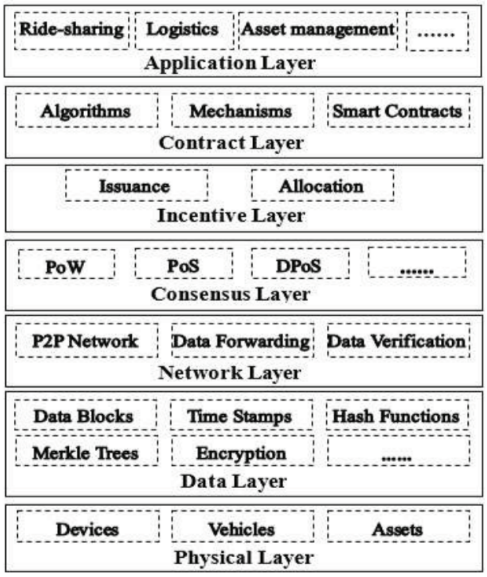


Figure 2.1: An ITS-Oriented Blockchain Model presented in [1]

Decentralization bring up different technologies that are easily integrable with DLTs for their characteristics. IPFS uses Distributed Hash Table to provide a decentralized file system accessible by anyone. This technology is useful to store data when is not convenient to do on DLTs. Zero Knoledge Proof is a technology that improve the privacy on ledgers using algorithms that proves something without revealing it. This become useful in a transparent environment such as blockchains.
Finally, through the use of decentralized ledgers and applications it is possible to provide an infrastructure as shown in Figure 2.1. Here from physical devices and data it is possible to provide applications for final ITS users through the use of distributed ledgers protocols. The layered structure presents a blockchain-based infrastructure where entities in the transportation system are able to communicate and transact in a trusted and secured way but also to reach consensus in decision making.

2.1 Vehicle Ad-hoc Network

VANET provides a wireless communication between moving vehicles, forming landscape where vehicle can communicate with other vehicles directly forming vehicle to vehicle communication (V2V) or with fixed equipment next to the road, called road side unit, forming vehicle to infrastructure communication (V2I). Dedicated short range communication (DSRC) is used and consists essentially in IEEE 802.11a and IEEE 802.11p that adds wireless access in vehicular environments (WAVE) for ITS , standardizing the whole communication stack by the 1609 family of standards. The different types of communications allow vehicles to share safety information, such as accident prevention, post-accident investigation and traffic jams, or other type of non-safety information, such as travelers information. The intention behind distributing and sharing this information is to provide a safety message to warn drivers about expected hazards in order to decrease the number of accidents and save people's lives, or to provide passengers with pleasant journeys [16].

2.1.1 Architecture

WAVE medium provides wide range of information and a comfortable driving to travelers and enables safety applications. This communication take place between different components: application unit (AU), on board unit (OBU) and road side unit (RSU). Typically the RSU hosts an application that provides services and the OBU is a peer device that uses the services provided. OBUs are connected to a series of vehicle sensors used to collect and process information that can be shared. Usually single or multiple AU are connected to a vehicle OBU to use services [2]. For authentication purposes, each network participant is equipped with a unique public/private key pair which usually resides in a tamper-proof-device (TPD) .

2.1.1.1 On Board Unit

OBU is a device mounted on-board of a vehicle used to exchange information with RSUs and other OBUs (hence other vehicles). In consists of a processor, a read/write memory, a user interface and a interface used to connect to other devices (based on IEEE 802.11 radio technologies). It can be used also to transfer data from/to different AUs in the vehicle. OBUs basically collect data from vehicle sensors to send it to other devices and receive information from the external context.

2.1.1.2 Application Unit

AU is a device within the vehicle that communicates with the OBU to use different provided applications. AU can be a dedicated device for safety and might even reside in the same physical unit or a personal digital assistant (smartphone). The AU runs applications that can utilize the OBU's communication capabilities

2.1.1.3 Road Side Unit

RSU is a device usually fixed along the road side or in dedicated locations such as at junctions. Main functions and procedures associated with the RSU are:

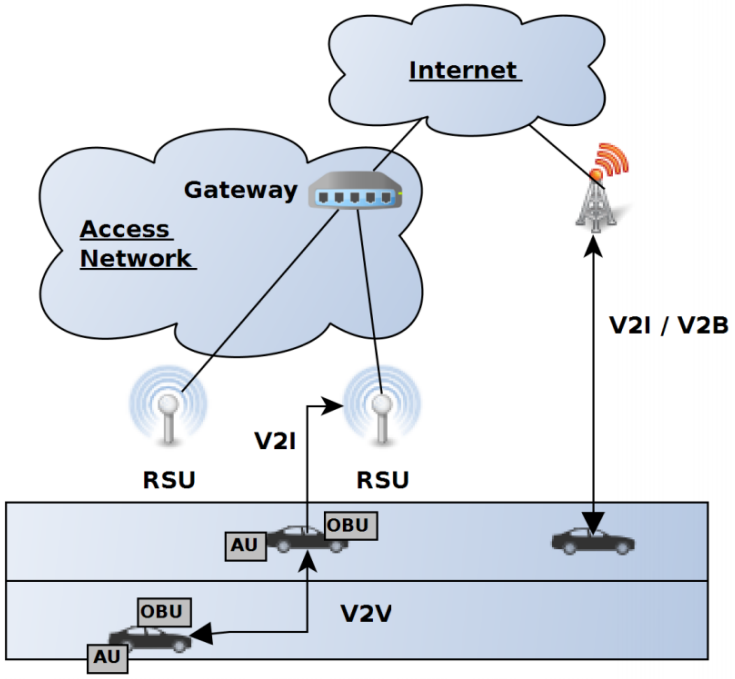


Figure 2.2: General VANET architecture, based on [2]

- Extending the communication range in VANET by re-distributing the information to other OBUs and RSUs in order to forward it to other network areas.
- Running safety applications using infrastructure to vehicle communication (I2V) and acting as an information source.
- Providing services to OBUs

2.1.2 Communication domain

Communication types in VANET can be categorized regarding the type of devices or the scope. It is possible to consider five general categories [3]:

2.1.2.1 Intra-vehicle

Intra-vehicle communication refers to the in-vehicle domain and consists of an OBU and one or multiple AUs. If OBU and Au do not reside in the same device, their communication link can be wireless or wired.

2.1.2.2 Vehicle-to-Vehicle V2V

Classified as the ad hoc domain, Vehicle-to-Vehicle communication consists in two or multiple OBUs communicating. The link can be direct through a wireless communication (single hop) or indirect when a dedicated routing protocol allows to forward data passing through multiple devices (multi hop).

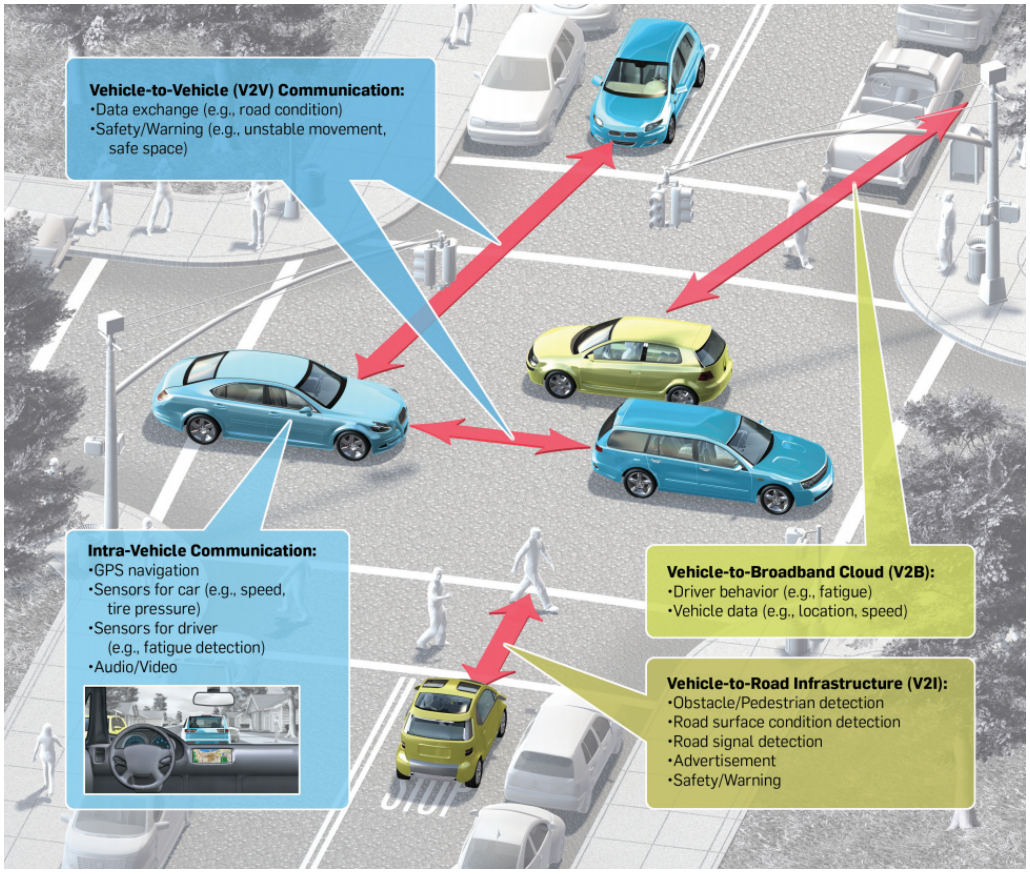


Figure 2.3: Communication types in VANET [3]

2.1.2.3 Vehicle-to-Infrastructure V2I

Also classified as the ad hoc domain in which a vehicle communicates directly with an RSU in order to increase the range of communication by sending, receiving and forwarding data.

2.1.2.4 Vehicle-to-Everything V2X

Vehicle to everything refers to the communication between vehicles, vehicles and roadside pedestrians, vehicles and infrastructures and in general a communication with a device.

2.1.2.5 Vehicle-to-broadband-cloud

Vehicle-to-broadband-cloud refers to the Infrastructural domain and consists in the RSU connecting to the infrastructural network, allowing OBUs to access the entire VANET or directly internet. OBU can communicate also with other hosts through AU using cellular radio networks (e.g. 5G).

2.2 Ethereum

Ethereum can be considered as the attempt to decentralize information access on internet, giving trust in digital world transactions, but especially as the acknowledgment of Bitcoin network limits, first one to use blockchain. Ethereum scope is to create an alternative protocol for decentralized applications development through a blockchain that integrates a Turing-complete language. Ethereum is an open-source project introduced by Vitalik Buterin via the whitepaper [4] at the end of 2013. No one controls Ethereum, however exists the Ethereum Foundation that controls the project development.

Blockchain 2.0 The difficulty to execute different operation apart from transferring currency in Bitcoin lead Buterin to propose Ethereum. In fact, Ethereum is capable of executing any kind of complex computation because of his Turing-complete blockchain.

Ethereum purpose Ethereum consists in a shared computer that belongs to everyone but is controlled by anyone. It is permanently available and no one can interrupt its execution or censor it. Different nodes connected in Peer-to-Peer style allow the execution of open source programs. Ethereum main purpose is to provide the ability to create anything inside the ecosystem, ideally exploiting decentralized consensus, in order to create new products and services not possible until now. Ethereum value increases with the number of applications based on itself, hence the main objective is to allow a development as easy as possible.

Ethereum Vision

- Multiple currencies: issue "tokens" that reside into the blockchain
- Decentralized Autonomous Organizations: manage shared resources through collective decisions
- Smart Contracts: rule through programmable code
- Smart Property: state the property of real world goods

Web 3.0 Among different definitions Web 3.0 can be seen as an interner where base services like DNS and digital identity are decentralized and individuals can engage financial interactions. Ethereum is particularly suitable to this vision of Web 3.0 and impose itself as a back-end for a secured, trusted and decentralized internet.

2.2.1 Structure

Ethereum is a cryptocurrency as well as an address system, a validators network, consensus algorithm, a ledger, a virtual machine, a set of programming languages, a complex economical structure and many other things [17]. Its structure can be divided into 4 layers:

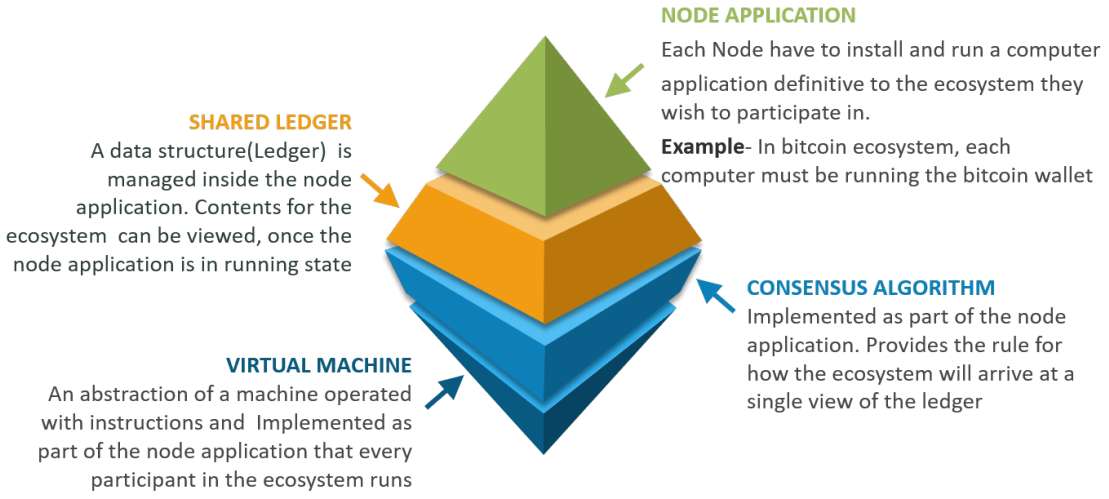


Figure 2.4: General structure of Ethereum represented by 4 Layers. https://www.edureka.co/blog/interview-questions/blockchain-interview-questions/

- **Wallet o Miner** - Application that interacts with the blockchain
- **Blockchain** - Distributed Ledger
- **Consensus Algorithm** - Proof of Work
- **EVM** - Virtual Machine

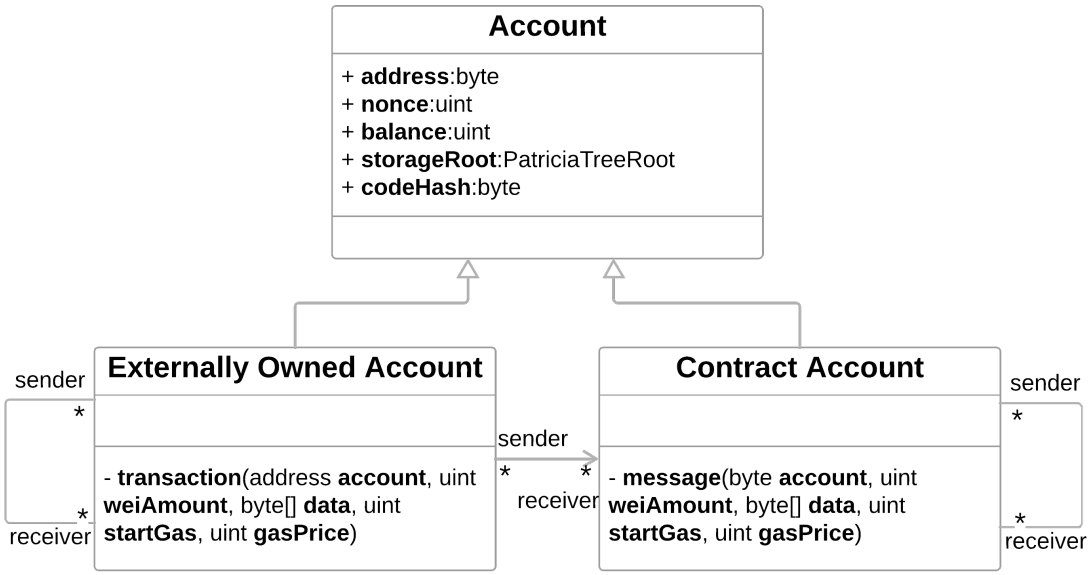
2.2.1.1 Wallet

Given the permissionless nature of Ethereum, every entity provided with an internet connection can access the blockchain, becoming a new node of the decentralized structure. Exist different kind of nodes but the simplest and most used one is the Wallet, an application that contains one or more accounts and that allows to send or receive information to the system.

Ether Ether is the main currency used in Ethereum, used to pay fundamental transactions in the system. The base unit is Wei and all the other units derive from this (1 Eth = 1^{18} wei). A table with all conversion rates is shown in appendix.

Accounts Accounts modify system state. Each account is represented by a 20 byte address and can be of two kinds: Externally Owned Account(EOAs) and Contract Account(CAs). Both maintain:

- **Nonce** - a counter used to process a transaction only once
- **Ether balance** - current ether balance
- **Storage** - a content storage
- **Code** - empty for a EOA but filled with code for a CA



Each account maintain a couple of keys, one public and one private. These two are used to encrypt and decrypt data and for digital sign, in order to grant anonymity and confidentiality during system execution.
The EOA is an account managed an entity external to Ethereum that can send messages into the platform through the creation of transactions signed using his private key. Whereas a CA is managed only by its code and is able to read and write to its internal storage, receive and send messages and create other CAs. It represents a Smart Contract.

Messages and Transactions Messages and transactions allows accounts to communicate and exchange data, but former ones are used only within the system, while the latter ones only from the outside.
Transactions are fundamental in Ethereum and are defined as data packets signed and sent by an EOA containing:

- Receiver account
- Sender signature
- Amount of Ether to transfer from sender to receiver
- Optional data
- STARTGAS and GASPRICE (values used to allow contract execution)

Messages are only exchanged between CAs and contain the same parameters as transactions but without signature.

2.2.1.2 Consensus Algorithm

The blockchain key concept consists in decentralization, hence the distributed ledger must be combined with a consensus mechanism that allows each participant to agree in the transactions order. This decentralized consensus requires that some nodes, called Miners, constantly try to produce valid blocks with new transactions received from other peers.

Miners For each new block received Miners immediately validates its transactions and try to create a new block to attach to this. In order to create a new block miners group new transactions received from peer nodes they are connected to. Steps are:

1. Determine transaction to use
2. Collect ommers, blocks created at previous step in the blockchain that were not selected
3. Apply block reward (amount of Eth gained by the creation of the block)
4. Compute the final state executing all the determined transactions
5. Compute Proof of Work

Proof of Work Proof of Work (PoW) is a value (nonce) cryptographically secure that proves without doubt that a computation work has been done, used to secure the blockchain. PoW function must be accessible by more individuals possible and cannot be possible to someone to profit more than anyone else with it. Formally the PoW function is:

$$PoW(H, H_n, d) = (m, n) \qquad \text{with} \quad m = H_m, n \leq \frac{2^{256}}{H_d}$$

- H is the block header without the two values *nonce* and *mixHash*
- Hn is the block header *nonce* value
- d is a dataset used to compute *mixHash*
- H_m is the block header *mixHash* value
- H_d is the difficulty for the block

PoW is a function that, given a block and a *nonce*, returns a value n. This value must be lesser or equal to a certain value $\frac{2^{256}}{H_d}$ determined by the difficulty H_d (from block 0 difficulty increases each time a certain number of blocks have been created and attached to the blockchain).
Security is assured because only a node that proves his work through the PoW is able to insert a block in the blockchain. Moreover, to incentivate the creation of blocks, a reward in Ether is given to the account that produced a block.

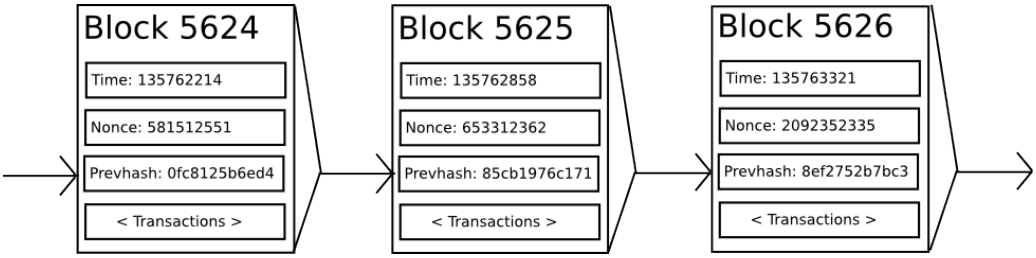


Figure 2.5: Blockchain representation in whitepaper [4]

2.2.1.3 Blockchain Architecture

Like Bitcoin, Ethereum is built upon a transactions ledger remotely validated in a distributed manner through the use of a Proof-of-work protocol. With this design each participant inputs unique data verified by the network of nodes. This data, represented by transaction, is used to generate blocks shared among the network that provide the history of what happened and create a truth verifiable by all participants. This ledger is called blockchain and is constantly growing with new blocks linked to the older ones forming a chain. Each Etherum node can send and verify transactions and take part to the competition of block creation: Mining.
The blockchain base mechanism consists in the fact that nodes accept a node received from another peer only if they can verify that each transaction in the block produce a valid system state after their code execution.

Block Besides transactions, a block contains an header with more informations:

- **parentHash**: previous block hash
- **ommersHash**: ommers or uncles are blocks created by the miner of this block at previous step that have ot been linked to the blockchain
- **beneficiary**: the account to which transfer the reward
- **stateRoot**: root of a tree structure that contains the system state after the execution of all transactions in the block
- **transactionRoot**: root of a tree structure that contains transactions
- **receiptsRoot**: root of a tree structure that contains transactions receipt (info on their execution)
- **logsBloom**: Bloom Filter composed by informations present in receipts
- **difficulty**: difficulty level
- **number**: block number
- **gasLimit**: total number of STARTGAS stated by transactions
- **gasUsed**: total number of used gas

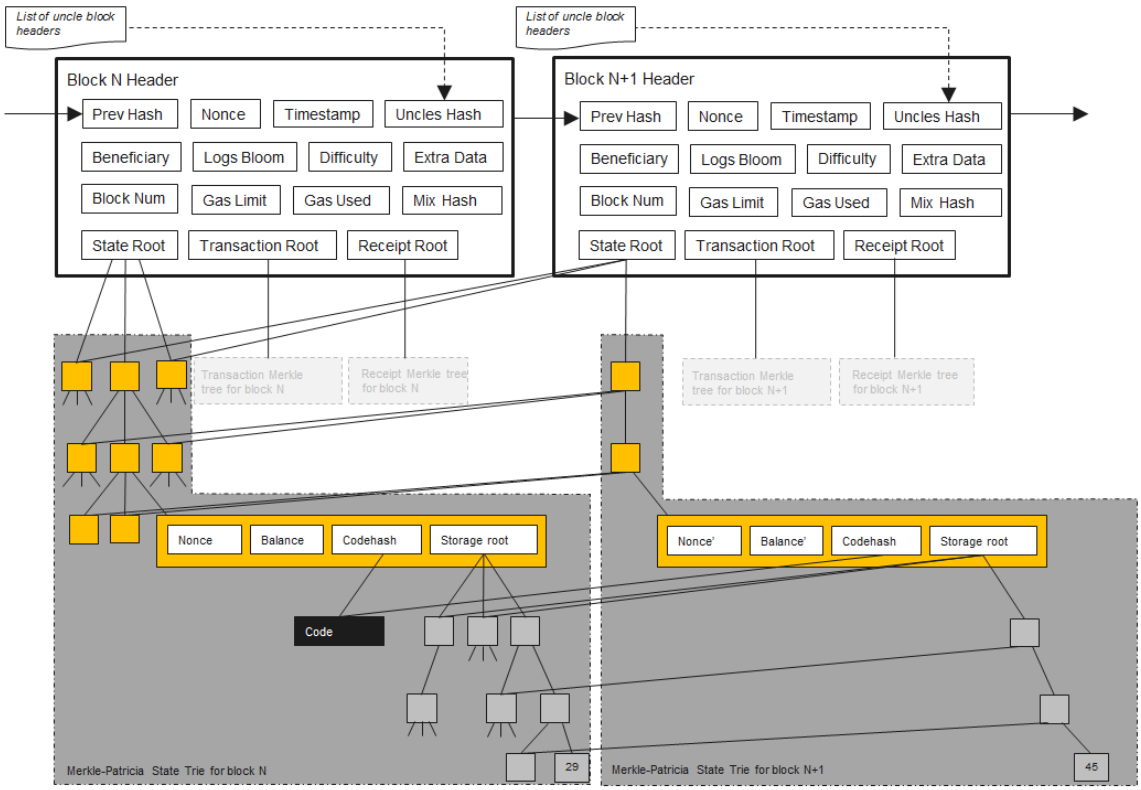


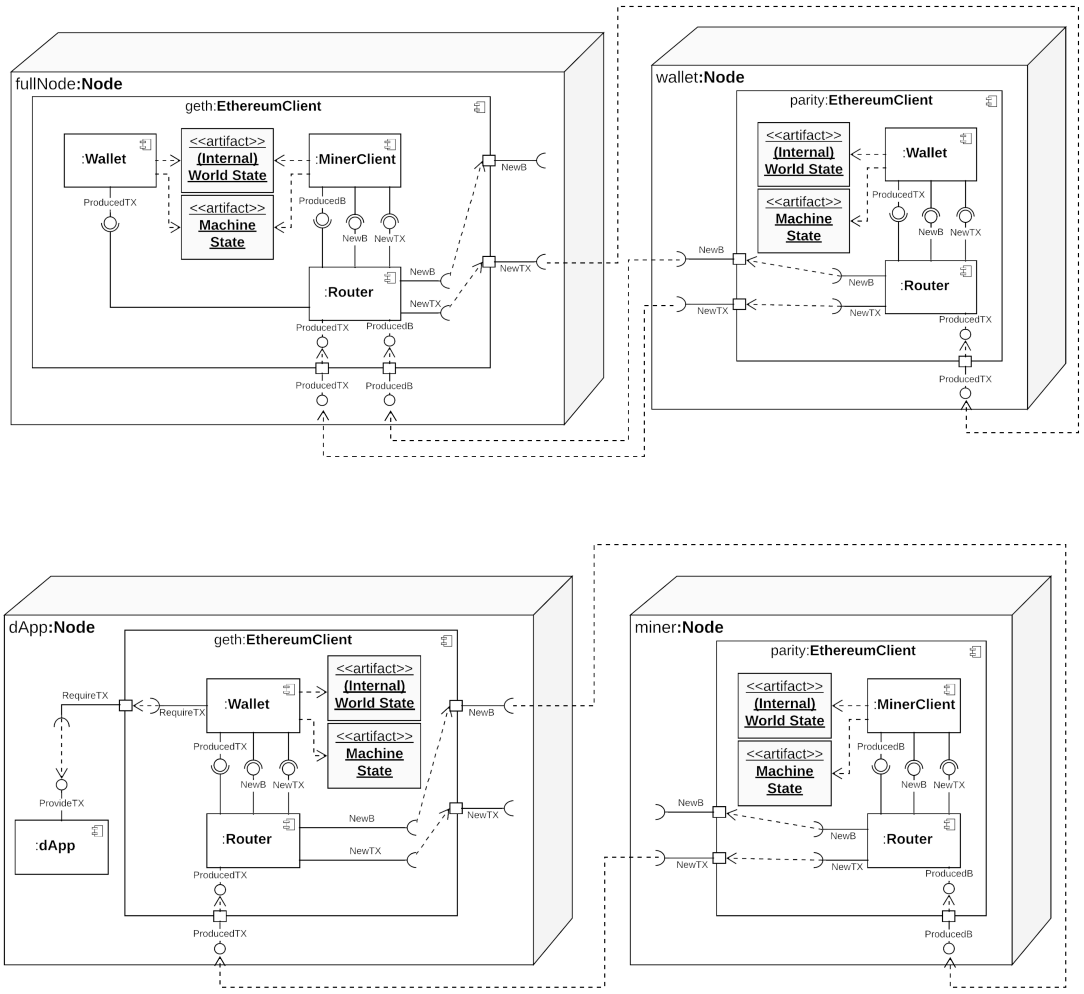
Figure 2.6: Block architecture in the Ethereum Blockchain https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture

- **timestamp**
- **extraData**
- **mixHash**: hash value mixed with the nonce to resolve the consensus algorithm problem
- **nonce**: value mixed with the mixHash to resolve the consensus algorithm problem

Opposed to the Bitcoin protocol, Ethereum does not limit the block dimension, instead it uses gas limit to overcome malicious attack. Another difference is that block time, which corresponds to the time interval between a block creation and its precedent, varies from 10 to 15 seconds instead of the 10 minutes required by Bitcoin.

Peer-to-Peer nodes architecture Ethereum follows pure p2p architectural style where the workload is equally shared between all the nodes in the network. In the Ethereum network there are different kind of nodes:

- **Miner Node** - store the entire blockchain and carry on Mining activity
- **Wallet Node** - do not store the blockchain, neither do Mining, only maintain a Wallet to exchange transactions



- **Full Node** - Miner Node with Wallet
- **dApp Node** - particular Wallet node that run an application that exploit Smart Contracts

Blocks Propagation When a block is successfully created by a Miner, it will be then broadcasted to the nodes connected to him. These nodes will verify the block and update their blockchain version. This propagation process requires time to reach furthest areas in the network, hence some nodes will have an outdated blockchain version. In the meantime another block can be broadcasted to these nodes, resulting in a network with two blockchain versions. This is called Fork and is solved imposing that the right blockchain is the longest one: nodes that do not follow this one will be left behind.

2.2.1.4 Ethereum Virtual Machine

Ethereum and Bitcoin blockchains are similar, the fundamental difference resides in the fact that transaction scripts are "stateless" in Bitcoin, i.e. state before and after the transaction script execution is not maintained, and "stateful" in Ethereum. Hence,

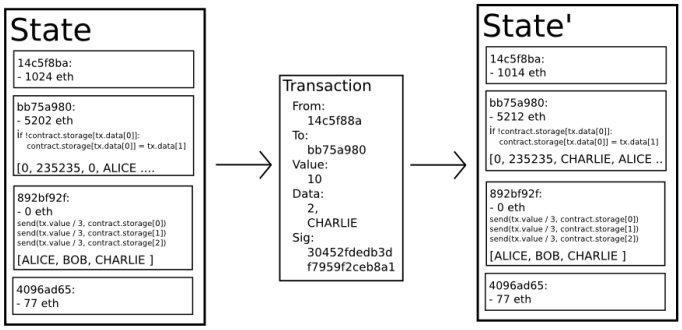


Figure 2.7: State transition representation in whitepaper [4]

Ethereum can be seen as a transaction-based machine that started with a "genesis" state and step by step executes transaction scripts to commute in a new state. This model name is Ethereum Virtual Machine (EVM), a *quasi*-Turing-complete machine, where the *quasi* indicates that computation in limited by a parameter: gas. EVM is a stack-based architecture virtual machine, where the stack elements are 256 bit words. Model memory is a simple volatile indexed array, this means that after the execution of a transaction script memoty is lost. Storage is non-volatile and maintained together with the system state in a tree structure called "Patricia tree" (a modified Merkle tree). Program code is saved in a virtual ROM.

Touring-Complete language Every system or programming language able to computer anything computable, given enough resources, is said Turing complete. Ethereum is Turing-complete because throug EVM and its low-level scripting language allows to write rules and execute programs identified as Smart Contracts. EVM is able to execute every computation, including infinite loop. To avoid malicious attack for this case, gas has been introduced.

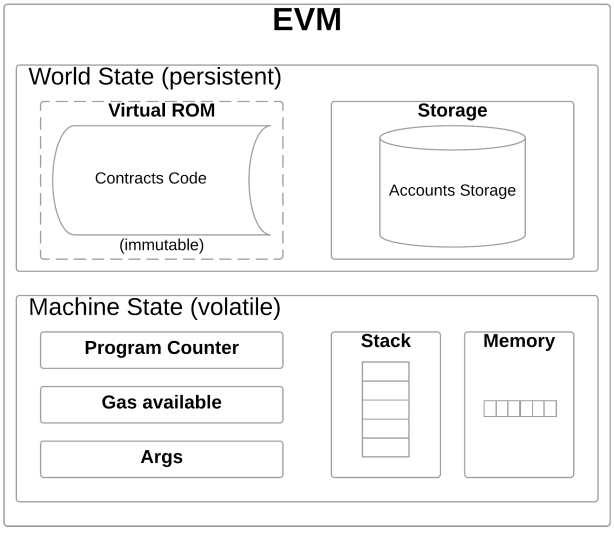


Figure 2.8: Ethereum Vitrual Machine stack-based architecture

Gas Computation in the EVM has a cost in gas. Each fragment of programmed computation, such as creating a contract or sending a message, has a universal recognized cost in terms of gas. Every transaction, indeed, must include a *STARTGAS* value that indicates the gas limit that the execution can reach. When the limit is overreached the transaction is canceled. Total gas units used during the execution are multiplied to another indicated value, the *GASPRICE*, in order to obtain the amount of wei that the transaction sender must pay as fee for the execution.

2.2.2 Functions and Use cases

Despite the complex architecture, projects built upon Ethereum show its real potential. Indeed, Ethereum created new concepts that aim to replace traditional financial and governance constructs.

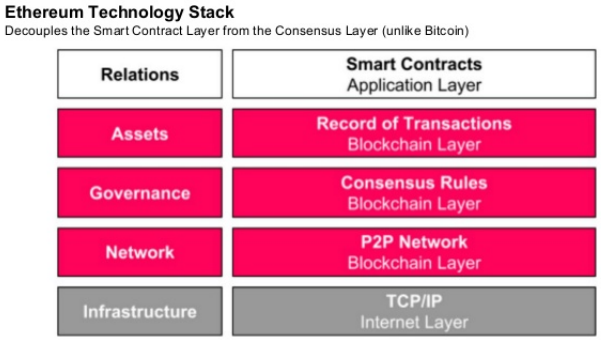


Figure 2.9: Ethereum Structure highlighting Smart Contracts. https://www.slideshare.net/SherminVoshmgir/ethereum-61549934

2.2.2.1 Smart Contracts

This new concept of contract is said to be "Smart" because it follows business logics executed semi-autonomously in order to move funds and force payments as a result of an agreement. Smart Contracts are fundamental to Ethereum because allow individuals to make exchanges on internet without the need of intermediaries. They are not governed by central authorities, nor need human presence, they are self-executed after terms settled between seller and buyer. Smart Contracts allow trusted transactions and agreements between individuals without the need of central authority, legal system or external mechanism of terms enforcement. Different languages are available in Ethereum to write code for Smart Contracts, but Solidity is mostly used. Solidity is an high-level object-oriented language, used to implement Smart Contracts. It has been influenced by C++, Python and JavaScript and designed for EVM to be compiled in low-level scripting code. It supports inheritance, libraries and complex used-defined types.
It can be considered reductive to associate Smart Contracts to simple object-oriented

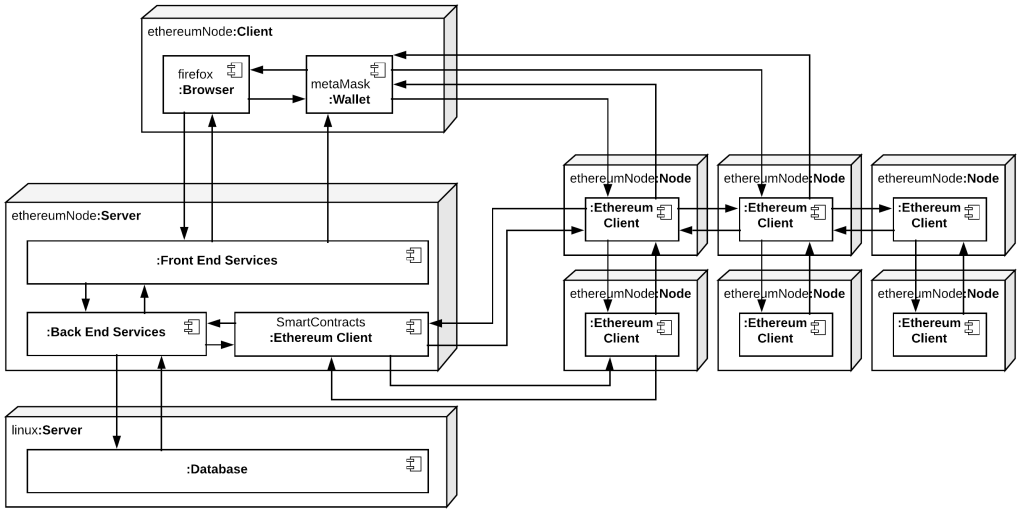


Figure 2.10: dApp hybrid architecture

classes because once their code is executed, becomes easy the task of transferring currencies.

2.2.2.2 Decentralized Autonomous Organizations

The combination of different kind of Smart Contracts can lead to the creation of Decentralized Autonomous Organizations (DAOs). A DAO is an organization managed through Smart Contracts and executed in a decentralized way in which the state is maintained via a consensus system. Contracts are used to implement transactions, currency flows, rules and rights inside the organization. Members are Ethereum accounts that can register and interact through contracts and their affiliation can be proved through the Blockchain immutability.

Properties and components The set of Smart Contracts used to manage a DAO usually enables:

- **Autonomy** - all DAO components must be implemented through the Blockchain and Open Source
- **Currency** - a dedicated token for internal payments
- **Proposals** - members can propose and vote operative decisions
- **Transparency** - internal processes and mechanisms must be transparent to all members
- **Consensus** - a democratic vote must be enacted to reach consensus
- **External Agents** - must be possible interact with external world

2.2.2.3 Decentralized Applications

Ethereum main scope consists in being the core base of Decentralized Applications (dApps). A dApp is an interface based on the Ethereum blockchain that allows final user to connect to a series of Smart Contracts. Usually dApps are web application based on Javascript and a core set of API (web3) that allows the communication with the EVM. dApps have their dedicated set of Smart Contracts used to conduct business logics and to persistently save their state.
The relation between a dApp and the Ethereum blockchain is similar to the relation between a traditional web app and a web server with a database.

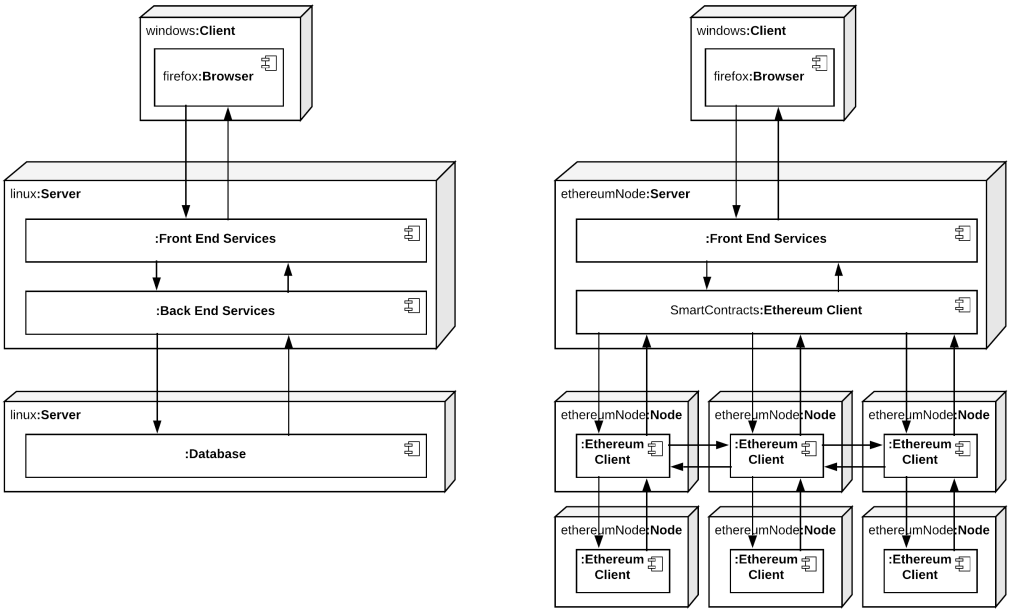


Figure 2.11: dApp architecture in comparison with a traditional 3-layer architecture

2.3 IOTA

IOTA [13] is a permissionless DLT that allows computers in an IOTA network to transfer immutable data and value among each other, specifically designed for the IoT industry. It is the first open-source distributed ledger that is being built to power the future of the IoT with feeless microtransactions and data integrity for machines, improving efficiency, increasing production, and ensuring data integrity in a machine-to-machine economy.
Clients send data and IOTA tokens to each other through IOTA nodes, which are nodes of a peer-to-peer system.To send and receive IOTA tokens, clients send packages of transactions called bundles to nodes. The transactions in a bundle instruct the node to transfer IOTA tokens from one address to another. These addresses are derived from a client's unique secret password called a seed. When the bundle is confirmed in the Tangle, the IOTA tokens are transferred.

Internet of Everything To solve some inefficiencies of the blockchain, IOTA is based based on a new concept of DLT, the Tangle, allowing the link between Internet of Everything (IoE) and Web 3.0. With the rapid development of IoT industry, it is increasingly important to allow machine-to-machine micropayments efficiently. In the case of Ethereum, for instance, it is possible that a transaction fee is larger than the amount of value being transferred but, it is not easy to get rid of fees in the blockchain since they serve as an incentive for the creators of blocks. To address this issue IOTA introduced the Tangle.

2.3.1 The Tangle

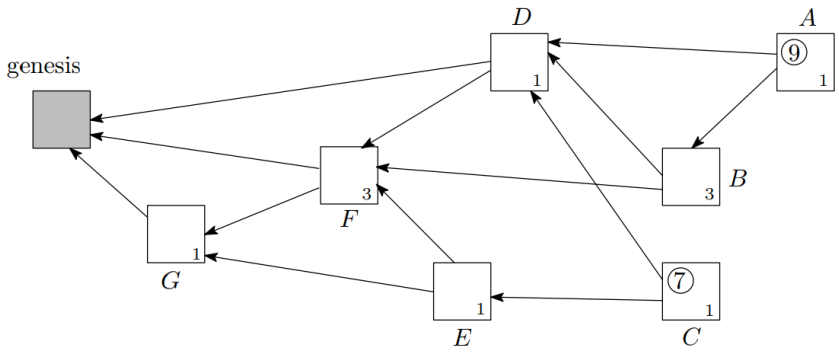


Figure 2.12: Tangle representation in whitepaper [5]

The Tangle [5] is a Direct Acyclical Graph (DAG), where vertices represent transactions, and edges represent approvals. The transactions issued by IOTA nodes constitute the site (vertex) set of the tangle graph, which is the ledger for storing transactions. In this environment, hence, nodes are entities that issue , but also validate, transactions. When a new transaction is issued, it must approve two previous transactions and the result is represented by directed edges (Figure 2.12). Transaction A indirectly approves transaction B if there is not a directed edge

between them, but there is a directed path of length at least two from A to B. At the beginning of the tangle, there was an address with a balance that contained all of the IOTA tokens and with a "genesis" transaction all these tokens were transferred to several other "founder" addresses. Genesis transaction, hence, is the first transaction and is approved either directly or indirectly by all other transactions. No tokens will be created in the future, and there will be no mining in the sense that miners receive monetary rewards "out of thin air".
As depicted also for the Ethereum blockchain, no one imposes any rules for choosing the transactions to approve. Instead, all the protocol is based on the fact that if a large number of nodes follow some "reference" rule, then for any fixed node it is better to stick to a rule of the same kind. To issue a transaction a node must:

1. choose two other transactions to approve according to the Tip Selection algorithm
2. check if the two transactions are not conflicting and do not approve conflicting transactions
3. solve a cryptographic puzzle similar to those in the Ethereum blockchain but simpler

Conflicting transactions The tangle may contain conflicting transactions, however, the nodes need to decide which transactions will become orphaned, i.e. transactions that are not indirectly approved by incoming transactions anymore. To decide which of the conflicting transaction become orphaned nodes run the Tip Selection algorithm many times and sees which of the two transactions is more likely to be indirectly approved by the selected tip. For instance, if a transaction was selected 97 times during 100 runs of the algorithm, it is confirmed with 97% confidence.

Transactions propagation Given the IOTA protocol, if a node does not issue transactions, resulting in being "too lazy", it will be dropped by its neighbors.Therefore, even if a node does not issue transactions, and hence has no direct incentive to share new transactions that approve its own transaction, it still has incentive to participate.

2.3.1.1 Weights and fundamental definitions

The weight of a transaction is a value 3^n, where n is a positive integer that belongs to a certain interval, proportional to the amount of work that the issuing node invested into it. By the sum of the own weight of a transaction with the sum of own weights of all transactions that directly or indirectly approve this transaction we obtain the cumulative weight. For instance in Figure 2.12 the boxes represent transactions, the small number in the corner of each box is the transaction own weight. In this case, transaction F is directly or indirectly approved by transactions A, B, C, D, E. The cumulative weight of F is $10 = 3 + 1 + 3 + 1 + 1 + 1$, i.e. the sum of F own weight and the own weights of A, B, C, D, E.

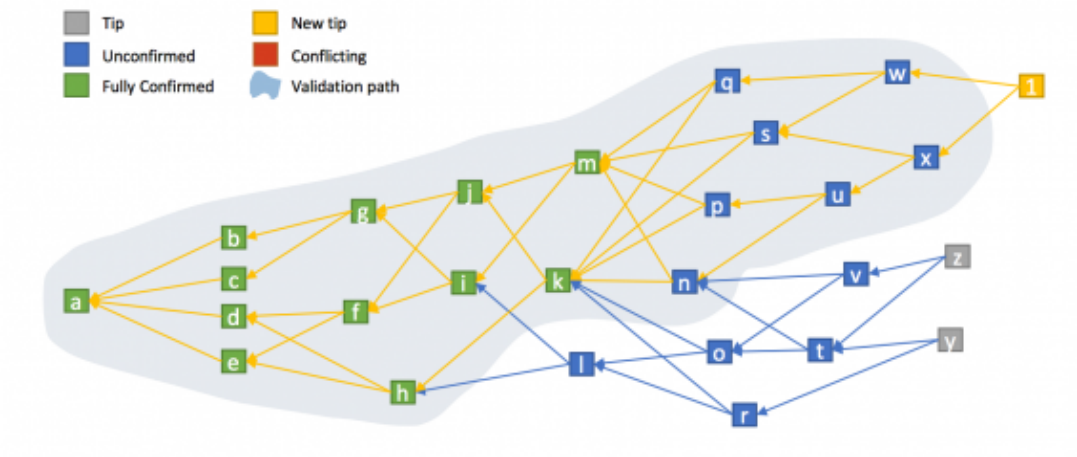


Figure 2.13: Adding a transaction http://untangled.world/iota-consensus/

Tips "Tips" are the unapproved transactions in the tangle, for instance A and C in Figure 2.12. When the new transaction X arrives and approves A and C, X becomes the only tip.

Height and Depth For a transaction site the height is the length of the longest oriented path to the genesis, whereas the depth is the length of the longest reverse-oriented path to some tip. In Figure 2.12 G has height 1 and depth 4.

Score The score of a transaction is the sum of own weights of all transactions it has directly or indirectly approved plus its own weight. In Figure 2.12 A approves transactions B, D, F, G, hence its score is 9=1+3+1+3+1.

2.3.1.2 Tip Selection algorithm

To select the two tips to approve the IOTA node selects a subgraph (also known as a subtangle) of the ledger and does two weighted random walks through it. Each weighted random walk returns a tip transaction. The idea is to place some random walkers on sites of the subtangle and let them walk towards the tips in a random way, thus using a Markov Chain Monte Carlo algorithm:

1. Select a subgraph considering all sites on the interval [W, 2W]
2. Independently place N walkers on sites in that interval
3. Let these walkers perform independent discrete-time weighted random walks "towards the tips", meaning that a transition from x to y is possible if and only if y approves x and its probability is given by the cumulative weight of y
4. The two walkers that reach any tip define the two tips that will be approved (first discard those random walkers that reached the tips too fast because they may have ended on one of the "lazy tips")

The transition probabilities of the walkers are defined in the following way: if y approves x ($y \rightsquigarrow x$), then the transition probability P_{xy} is proportional to

$$\exp\Big(- \alpha(\mathcal{H}_x - \mathcal{H}_y)\Big)$$

where $\alpha > 0$ is a parameter to be chosen and $\mathcal{H}$ is the current cumulative weight of a site. This algorithm select very unlikely lazy tips and parasite chain tips issued by malicious users.

2.3.1.3 Nodes incentive

The protocol does not enforce the tip selection algorithm, instead, nodes have an incentive to use it to have the best chance of their transactions becoming confirmed. It is impossible to check if a node used the tip selection algorithm or even changed it to return custom tip transactions for its own purposes.
In [18] the existence of ("almost symmetric") Nash equilibria is proved for the system where a part of players tries to optimize their attachment strategies and also showed that the "selfish" players will nevertheless cooperate with the network by choosing attachment strategies that are similar to the "recommended" one.

2.3.1.4 Coordinator

The current version of IOTA exists an unique dedicated program called Coordinator. The Coordinator only issues a normal signed transaction commonly known to as a "milestone". Currently, as long as another transaction is referenced by this milestone, the transaction is confirmed. The reason and purpose of the issuance of these milestones is to protect the Tangle at this early stage against hostile attacks, and defends against double spends or spending from non-existent funds.

2.3.2 Structure

The Tangle is similar in spirit to a blockchain, but overcomes some of its fundamental limits in scaling, because the Tangle does not have a built in maximum throughput, in fees, because there are no Miners, therfore no miner fees, and finally in Miners, because there are no incentives to slow the network down to raise fees and other conflicts of interest.
An IOTA network consists of nodes and clients. A node is a device that has read/write access to the Tangle, whereas a client is a device that has a seed. A seed gives a client access to addresses and these have a balance, which defines the amount of IOTA tokens in them. In order to operate, clients must send bundles of transactions to a node so that the nodes can validate the transactions and update their ledgers.

2.3.2.1 Trinary

IOTA represents data according to the trinary numeric system. Compared to binary, trinary computing is more efficient as it can represent data in three states rather then just two. In balanced trinary, data can be represented as 1, 0, or -1. These

values are called trits and three trits is equal to one tryte, which can have 27 (3^3) possible values. All data represented in IOTA is tryte-encoded, according to the tryte alphabet which contain all the English alphabet letters plus the character '9'.

2.3.2.2 Clients

Each client in an IOTA network has a secret password called a seed, which is used to derive addresses. An address in IOTA is a unique string of 81 trytes that are unique to each seed. A client on the network sends data or IOTA tokens to other addresses in bundles, which contain transactions.

Addresses and signatures Seeds are the master keys that allows clients to create pairs of address and private keys. Addresses are public and clients can send IOTA tokens and messages to them using the address field of a transaction. A private key is used to sign bundles to attach to the Tangle.

2.3.2.3 IRI Nodes

The IRI (IOTA reference implementation) is open-source Java software that defines the IOTA protocol. Computers that run the IRI are called IRI nodes. Nodes keep a ledger of every transaction that they receive and the non-zero balances of all addresses in the network. All nodes communicates with each other in a peer-to-peer system in order to accomplish the following key functions:

- Validate transactions
- Store valid transactions in a ledger
- Allow clients to interact with the IRI and have their transactions appended to the ledger through a set of APIs

2.3.2.4 Transactions and bundles

A transaction is a single operation that you can send to a IRI node, then the node will validate the transaction and attach it to the Tangle. Transactions can withdraw/deposit IOTA tokens or send data. To send a node one or more transactions, you must package them in a bundle. Transactions can be either:

- **Input transaction** - withdraw IOTA tokens from addresses
- **Output transaction**
  - A zero-value transactions that contains only a message in the signatureMessageFragment field
  - A transaction with a positive value that deposits IOTA tokens into an address

Structure A transaction object contains the following fields:

- **signatureMessageFragment**: a signature or a message, both of which may be fragmented over multiple transactions in the bundle.
- **address**: contains a recipient's address if the transaction is an output, otherwise it contains an address from which IOTA tokens are being withdrawn
- **value**: amount of IOTA tokens to deposit to or withdraw from the address
- **obsoleteTag**: user-defined tag (soon to be removed)
- **timestamp**
- **currentIndex**: index of a transaction in the bundle
- **lastIndex**: index of the last transaction in the bundle
- **bundle**: hash of the bundle
- **trunkTransaction**: transaction hash of a parent transaction
- **branchTransaction**: transaction hash of a parent transaction
- **attachmentTag**: user-defined tag
- **attachmentTimestamp**
- **attachmentTimestampLowerBound**: lower limit of the attachmentTimestamp field (not currently used)
- **attachmentTimestampUpperBound**: upper limit of the attachmentTimestamp field (not currently used)
- **nonce**: trytes that represent the amount of times a transaction must be hashed to check the proof of work

2.3.3 Functions and Use cases

IOTA goal is to become the very fundamental layer of a world full of small IoT devices, where their small jobs, flow of microscopic data, and nano-payments come and go all over the globe. Different projects are developed towards this way.

2.3.3.1 Masked Authenticated Messaging

Masked Authenticated Messaging (MAM) is a second layer data communication protocol which adds functionality to emit and access an encrypted data stream over the Tangle regardless of the size or cost of device. MAM allows to maintain integrity and confidentiality in a data flow produced by a device.
MAM works through Channels, where the publisher and viewers meet. Viewers subscribe to channels to get data the data that channel owner publishes. This ownership is implemented and secured through the owner account seed. MAM is one of IOTA's most used modules and opens up a new field of use cases on top of IOTA. Being able to secure data's integrity and control its access management is a prerequisite for machine-to-machine communication.

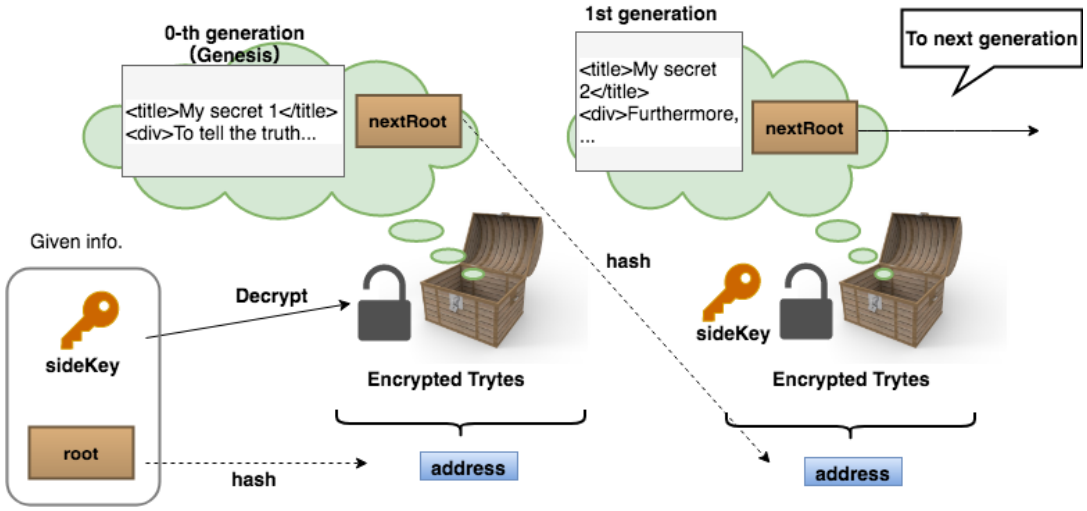


Figure 2.14: Graphical representation of a MAM channel in restricted mode https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

Channels Channels can be considered as message chains where each message, i.e. a transaction, is linked to the next one. Owners publish messages in this chain and viewers can access the stream by knowing the address of only one message. Address is obtained from a generated parameter called root and knowing a message address limits the viewer to access only messages that comes next, not previous ones. Three channel modes are possible:

- **Public** - root is used as the address of the transaction that the message is published to, but also to encrypt the data. Hence anyone that knows a message root will be able to access all messages published from that one on. It could be used for public announcements from a device with properties of immutability and data integrity.
- **Private** - root hash is used as the messages address and root to encrypt the message, hence a random user cannot decrypt a message if he knows only the address due to the fact that they are unable to derive the root from the hash. Private mode can be used for encrypted streams not meant for public consumption.
- **Restricted** - root is combined with an authorization key and the hash is used as the messages address, then the side key is also used to encrypt the data. The publisher could stop using the side key without changing their channel, so access could be revoked from subscribers if desired.

Merkle tree based signature scheme MAM uses the Merkle tree based signature scheme. Owner seed is used to create a Merkle tree and the root value is the root of this one. Merkle tree has two integer parameters: *start* and *size*. These represent the index of address being generated from seed by this way (without considering security levels):

$$private\ key = hash(seed,index)\ ,\ address = hash(private\ key)$$

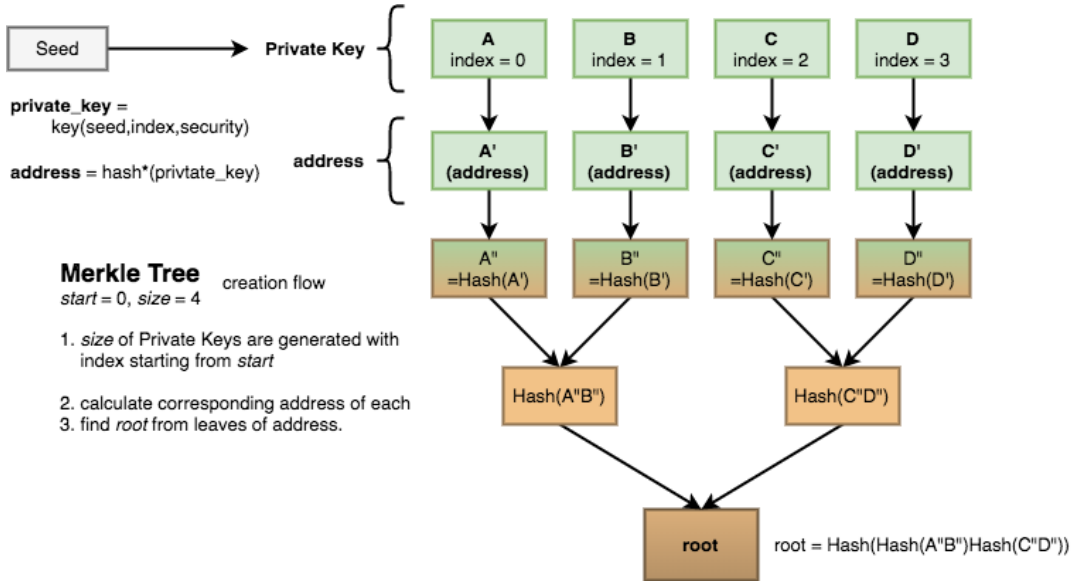


Figure 2.15: Graphical representation of the Merkle tree used in MAM channels https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

Then Merkel Tree is created by hashing all the $n(=size)$ addresses as leaves, up to root narrow them down by hashing combined of each pair.
One of the private key generated is used for the message signature.

Structure of MAM Bundle Various transactions can be included in a bundle to create a message to publish in a channel. In this bundle two sections are distinguishable: MAM section and Signature section. The Signature section contains a signature obtained by signing the MAM section whereas this one contains:

- **branchIndex** - the index corresponding to the private key used to sign
- **siblings** - the set of complementary hashes that, combined with given leaf address (obtained by validating the signature), can generate tree root. For instance in Figure 2.15, in the case of branchIndex = 0, address A' is obtained by validating the signature, then siblings are B" and Hash(C"D")
- **nextRoot** - the root of next message Merkle tree, used to create links in the chain

Validation Finally when fetching a message through a root (and eventually a side key) the first operation consists in validating the signature. The validation process return an address, which is used as a leaf address of index=branchIndex for a new Merkle tree and combined with siblings in order to calculate the tree root. If this new root equals the given root, then this unmasked message is valid.

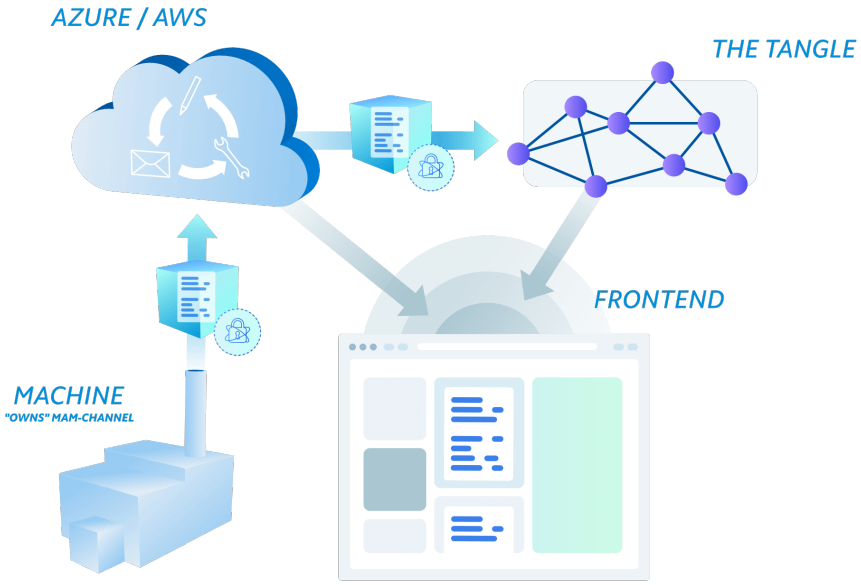


Figure 2.16: Diagram of data flow in IOTA Data Merketplace https://data.iota.org

2.3.3.2 Data Marketplace

Data Marketplace was launched by the IOTA Foundation in 2017, as a Proof of Concept and open innovation ecosystem. Nowadays Data Marketplace PoC evolved into a system that makes it possible to securely store, sell and access data streams. Data sources are accessible through the main website and referenced through a geomap or a list. These are available after approval from IOTA team. Visitors of the website can select data sources and access the data sets directly from the browser after purchasing the sensor data through IOTA Tokens.

Devnet As well as other DLTs, IOTA also has a separate ledger used for testing. It is called "Devnet" and it is a version of the Tangle meant for development and testing purposes. It allows developers to work on their apps without getting in their way or costing them any real tokens.

Masked Authenticated Messaging In the Data Marketplace implementation MAM is used and each data packet is encrypted with it is own randomly generated encryption key. All keys are securely stored in a cloud backend system.

Payment Data can be decrypted if payment is made. Price of the data stream is defined by sensor owner, and usually set between 1000 and 50000 IOTAs. But it is important to notice that IOTA Token in the Devnet are worth nothing, thus buyer can fund his wallet for free and spend tokens exclusively for purchasing sensor data.

Data storage in the Cloud The Data Marketplace PoC uses a cloud backend service provided by Google Firebase. All information is stored securely and confidentially. Google cloud services are used for user authentication and access rights management.

Data Streaming New data packets can be added with a recommended minimum time interval of 5-10 min. More frequent additions are possible, but since a publisher needs to conduct a small amount of Proof of Work to allow the data to propagate through the network, the delay could be up to 60 seconds for every new packet.

2.4 IPFS

InterPlanetary File System (IPFS) [14] is a protocol that allows to connect all computing devices in a peer-to-peer network with the same distributed file system. IPFS is similar to the Web, but it could be seen as a single BitTorrent swarm, exchanging objects within one Git repository. It creates a resilient system of file storage and sharing, with no single point of failure where network nodes do not need to trust each other. At the heart of IPFS is the MerkleDAG, a directed acyclic graph whose links are hashes and sites are objects. This MerkleDAG lets IPFS offer authenticated and permanent contents and an addressing based on these, but also the possibility for object to be served by untrusted agents, cached permanently and created and used offline.

Possession and participation World Wide Web is structured on ownership and access, meaning that a file can be accessed only from whoever owns them with their permission. On the other end IPFS is based on the ideas of possession and participation, where many people have each others' files and participate in making them available. That means IPFS only works well when people are actively participating. This allows IPFS to:

- Make it hard for a website hosted on IPFS to go offline, because website pages are stored somewhere else
- Make it harder for authorities to censor content
- Speed up the web when nodes are far away or disconnected from each other

2.4.1 Structure

IPFS has a stack of modular protocols, formed using a variety of successful projects. This stack is composed by five layers:

- **Network** - network stack for communication between peers, such as transport protocols, reliability mechanisms, NAT traversal techniques, integrity checks, authentication
- **Routing** - routing system that locate other peers and peers who store particular objects
- **Block Exchange** - block transport and replication, communication between nodes of what data a peer can provide and what data he is looking for
- **Merkle DAG** - data structure format
- **Naming** - a self-certifying PKI namespace (IPNS)

First three layers are implemented using libp2p, an extensible library developed by IPFS and Ethereum developers. Libp2p is a high-level, developer friendly, set of modules that contain implementations of network protocols that are suitable for p2p communication. Libp2p provides protocols such as: Transports, Stream

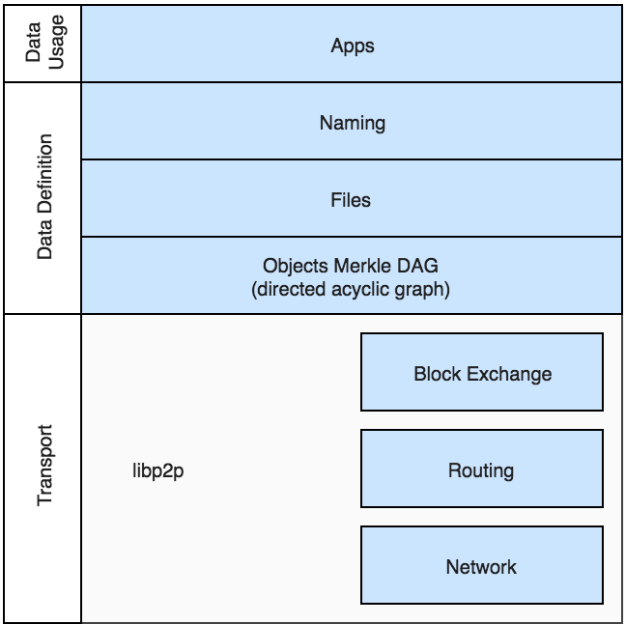


Figure 2.17: IPFS technology stack https://takahser.github.io/tldr/ipfs.html

Muxers, Crypto Channels, Discovery, Peer Routing, Record Stores, NAT Traversal, Connection.

2.4.1.1 Network

The network provides point-to-point transports (reliable and unreliable) between any two IPFS nodes in the network. Each node has a NodeId which is the cryptographic hash of its public key. When peers initialize a connection, they exchange their public key and check if the publick key hash equals the NodeId, If not, the connection is terminated. Nodes can change their NodeId anytime, but, they are incentivized to remain the same. The nodes store objects (of their interest) in their local storage.

2.4.1.2 Routing

The IPFS Routing layer serves two important purposes:

- **Peer routing** - to find other nodes
- **Content routing** - to find data published to IPFS

In order to find other peers' addresses and peers who can serve particular objects, a Distributed Hash Table is used. DHTs can be used to create a semi-persistent routing record distributed cache in the network. For IPFS, in fact, objects references are stored and paired with the NodeIds of peers who can serve these. All the nodes maintain this DHT.

2.4.1.3 Block Exchange

The IPFS Block Exchange takes care of negotiating bulk data transfers, consisting in blocks that compose IPFS objects. IPFS uses BitSwap, a protocol inspired by

BitTorrent, that requires peers to provide the fields *want list* (the data a peer is looking for) and a *have list* (the data a peer can serve). BitSwap acts as a persistent marketplace, where peers can exchange their data. Once nodes know each other and are connected this exchange protocol governs how the transfer of blocks occurs.

2.4.1.4 Objects Merkle DAG

A Merkle DAG is used for storing and distributing data blocks composing objects quickly and robustly. IPFS is essentially a P2P system for retrieving and sharing IPFS objects. An IPFS object is a data structure with two fields:

- **Data** - unstructured binary data of size less than 256 kB
- **Links** - array of Link structures where each structure links to another IPFS object. A Link structure has three data fields:
  - **Name** - name of the Link
  - **Hash** - hash of the linked IPFS object
  - **Size** - cumulative size of the linked object, including following its links

Content Addressing IPFS objects are normally referred to by their Base58 encoded hash. A content identifier(CID) is a label used to point to material in IPFS. It doesn't indicate where the content is stored, but it forms a kind of address based on the content itself. The hash is actually a multihash, meaning that it specifies the hash function and length of the hash in the first two bytes of the multihash.

Files The data and named links gives the objects collection the structure of a Merkle DAG, in which sites are Data and edges are Links. The IPFS filesystem is then also modelled on top of this Merkle DAG.

- **Small Files** - A small file, i.e. less than 256 kB, is represented by an IPFS object with data being the file contents and no links
- **Large Files** - A large file, i.e. greater than 256 kB, is represented by a list of links to file blocks that are less than 256 kB, and only minimal Data specifying that this object represents a large file. The links to the file blocks have empty strings as names.
- **Directory** - A directory is represented by a list of links to IPFS objects representing files or other directories. The names of the links are the names of these files and directories

Versioning MerkleDAG allows IPFS to represent the data structures used by Git to allow for a versioned file systems. A Commit object has one or more links with names *parent0*, *parent1*, etc... pointing to previous commits, and one link with name object (corresponding to the Git tree) that points to the file system structure referenced by that commit.

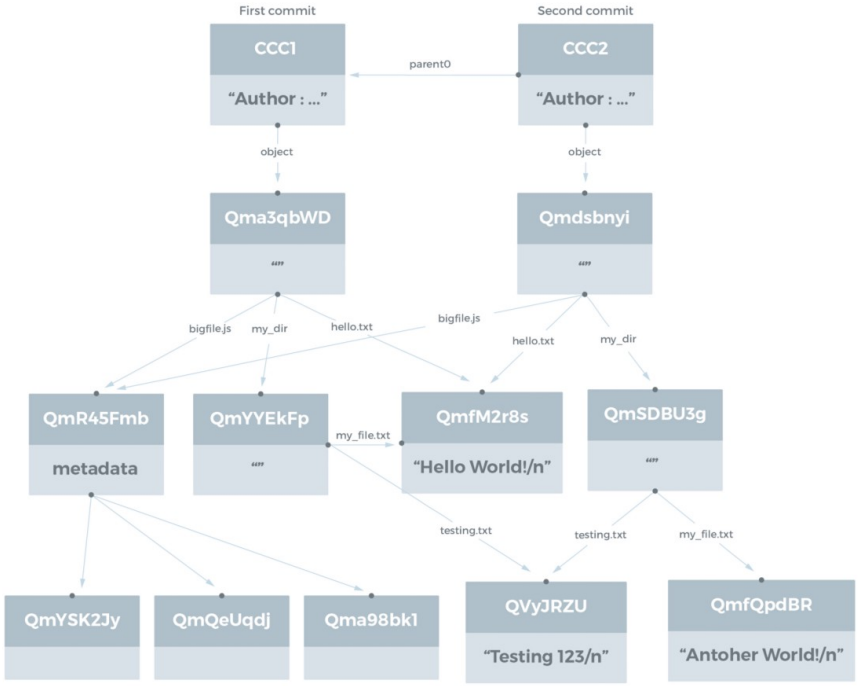


Figure 2.18: IPFS file system directory structure, with versioning https://medium.com/@ConsenSys/an-introduction-to-ipfs-9bba4860abd0

2.4.1.5 Naming

While objects in the merkle DAG are immutable, the naming is mutable. Changing an object would change its hash – and thus its address, making it a different object altogether. The IPFS naming layer (IPNS) handles the creation of mutable pointers to objects and human-readable names. IPNS is based on Self-certifying File System, a PKI namespace where a name is simply the hash of a public key. Whoever controls the private key controls the name. Records are signed by the private key and distributed anywhere in IPFS, via the routing system.

2.4.2 Functions and Use cases

IPFS provides a generic way to share files and data in a peer to peer fashion optimizing delivery. Some of the use case scenarios are:

- Sharing files and huge datasets
- Serving websites
- Using as a data source or sink in data processing workflows
- Using as a Mounted filesystem

2.4.2.1 Pubsub

Publish-Subscribe (pubsub) is a pattern often used to handle events in large-scale networks. Publishers send messages classified by topic or content and subscribers receive only the messages they are interested in, all without direct connections between publishers and subscribers. This approach offers much greater network scalability and flexibility. IPFS has its own version of pubsub.

2.4.2.2 OrbitDB

OrbitDB is a serverless, distributed, peer-to-peer database that uses IPFS as its data storage and IPFS Pubsub to automatically sync databases with peers. OrbitDB provides various types of databases for different data models and use cases:

- **Log** - an immutable (append-only) log with traversable history
- **Feed** - a mutable log with traversable history. Entries can be added and removed
- **Key-value** - a key-value database
- **Docs** - a document database to which JSON documents can be stored and indexed by a specified key
- **Counter** - Useful for counting events separate from log/feed data

2.5 Zero Knowledge Proof

In cryptography, a zero-knowledge proof is a method by which one party, called Prover, can prove to another party, called Verifier, that he knows a value x without giving any information except the fact that he knows x. The goal of zero-knowledge proofs is for the Verifier to be able to convince himself that the Prover possesses knowledge of the secret parameter x, called a witness. The challenge is to prove such possession without revealing the information itself or any additional information.

Definition A zero-knowledge proof must satisfy three properties:

- **Completeness** - if the statement is true, the honest Verifier will be convinced of this fact by an honest Prover
- **Soundness** - if the statement is false, no malicious Prover can convince the honest Verifier that it is true, except with some small probability.
- **Zero-knowledge**: if the statement is true, no Verifier learns anything other than the fact that the statement is true

2.5.1 zk-SNARK

The acronym zk-SNARK stands for zero-knowledge Succinct Non-interactive ARguments of Knowledge [19] and consists of a zero-knowledge proof where no interaction is necessary between Prover and Verifier. This constitutes the backbone of the Zcash protocol [20], one of the most used cryptocurrency that provide enhanced privacy for its users compared to Ethereum and Bitcoin.

2.5.1.1 Definition

A zk-SNARK consists of three algorithms G, P, V defined as follows:

Program A program C:

$$C(x, w) \longrightarrow \{0, 1\}$$

takes as input a public value x and a secret witness w and returns a boolean value.

Generator A Key Generator G:

$$(pk, vk) = G(C, \lambda)$$

takes as input a program C and a secret parameter λ and generates two public keys, a proving key pk and a validation key vk. These keys are public parameters that only need to be generated once for a given C. The λ parameter is called "toxic waste" because anyone who knows this parameter can generate fake proofs, hence must be destroyed during the process.

Prover A Prover algorithm P:

$$proof = P(pk, x, w)$$

takes as input the proving key pk, the public input x and the secret witness w and generates a $proof$ that the Prover knows a witness w and that the witness satisfies the program.

Verifier A Verifier algorithm V:

$$V(vk, x, proof) \longrightarrow \{0, 1\}$$

takes as input the verification key vk, the public input x and the $proof$ and returns true if the Prover knows a witness w satisfying $C(x, w) == 1$, false otherwise.

2.5.2 Zero Knowledge Proof of Location

Proof of Location (PoL) has been recently revised in conjunction with DLTs in order to achieve a decentralized consensus about the position of events or agents in space and eventually in time. Platin Zero Knowledge Proof of Location [21] allows a Verifier to test whether position committed by a Prover is inside or outside the radius of a service area, without revealing exact location.
The main idea consists in considering $A(x_{,0}, y_0, z_0)$ and $B(x, y, z)$ as two points in the space, where A is some reference point and B is the Prover's location, Zero Knowledge PoL allows the Verifier to verify that B is a point inside the sphere with center A and radius d. This statement corresponds to the inequation:

$$d^2 - (x - x_0)^2 - (y - y_0)^2 - (z - z_0)^2 > 0$$

To prove that exists a positive value equal to $d^2 - (x - x_0)^2 - (y - y_0)^2 - (z - z_0)^2$ the Prover uses the Lagrange theorem, which states that any positive number may be represented as the sum of four squares:

$$d^2 - (x - x_0)^2 - (y - y_0)^2 - (z - z_0)^2 = a_1^2 + a_2^2 + a_3^2 + a_4^2$$

(such a 4-tuple does not exist for a negative difference, i.e. a point outside the range).

To hide the secrets (x, y, z, a_1^2, a_2^2, a_3^2, a_4^2) a discrete logarithm is used.
Let p and q be primes, $n = pq$. Then for some (properly chosen) positive $g < n$, the function f is a one-way function if p or q are unknown:

$$f(u) = g^u \pmod{n}$$

Prover only needs to report values $f(x)$, $f(y)$, $f(z)$, $f(a_1^2)$, $f(a_2^2)$, $f(a_3^2)$, $f(a_4^2)$, allowing the Verifier to verify the distance through the equation:

$$g^{d^2-(x-x_0)^2-(y-y_0)^2-(z-z_0)^2} = g^{a_1^2+a_2^2+a_3^2+a_4^2}$$

2.6 Related Work

In [1] the authors conducts a preliminary study of Blockchain-based ITS, giving the basis for a new ITS-Oriented Blockchain Model. In this work the focus in on the blockchain potential to help establish a secured, trusted and decentralized ITS ecosystem, hence advantages and research issues are showed to be used as reference in other works. In [22] Leiding et al. combine VANETs and Ethereum to depict a decentralized system able to provide service and enforce rules in an ITS. This work lead the way to the CHORUS Mobility startup [11], specialized in enabling services for drivers in the Vehicular Network. In [23] various use cases are shown by authors to validate blockchain applications in the ITS.
Different works provide blockchain-based solutions for inter vehicle communication, for instance in [24] a this communication is supported by a Trust value. Authors in [25] presented a Privacy-Preserving Blockchain-Based application used for incidents and traffic announcements, using a token as incentive, whilst in [26] messages exchanged are validated through the use of Proof of Location.
In an environment such as the ITS where data sharing is fundamental, how to obtain this data is crucial and could involve different paradigms and technologies compared to the standard ones. These works [27] [28] follow exactly this work, proposing, in the context of ITS, blockchain based architectures where data is the main focus. In fact, they use content-centric networking instead of conventional transmission control protocol/Internet protocol (TCP/IP) routing.
Data sharing implies a Marketplace in the case of access restricted by payment and in these works [29] [30] authors present Data Marketplace systems backed by blockchains for IoT infrastructures and Smart Cities.
Finally in [31] is introduced the concept of Cooperation as a Service in VANETs with the deployment of a publish/subscribe interaction scheme in which participants can act as subscribers who express their interest in an event and publishers who submit information regarding those events to the system.
All of these works constitute the basis for this thesis work with their concepts or applications intended to provide relevant blockchain use cases in the ITS.

3 Vision

The aim of this chapter is to define the main concepts that will lead to the realization of the system meant to be developed in this thesis work. In section 3.1 is discussed the idea at the core of the system, providing a description of all major components. System functional and non functional requirements are listed in section section 3.2, whilst in 3.3 a study on the feasibility will be carried out. Finally in section 3.4 various scenarios are described in order to better understand system dynamics.

3.1 Idea

The main idea consists in building DLT-based ecosystem for mobility in the ITS. Applications in this ecosystem are decentralized because they are designed to depend the least possible from central entities, hence giving the user maximum control over his data and transactions. Decentralization is the primary reason to use a DLT based approach in order to allow anyone to be able to participate to the system without depending on any central authority. The aim of this work is to provide a solution ready to be implemented with few services provided from central authorities and the majority of them provided by users. Distributed technologies, indeed, offer an always available interaction with no or few fees. Trust in transactions is offered by these technologies covering an important quantity of transportation services in a blockchain-based ITS. Furthermore, other kinds of Trust can be built on these technologies, e.g. Proof of Location (Section 2.5.2).

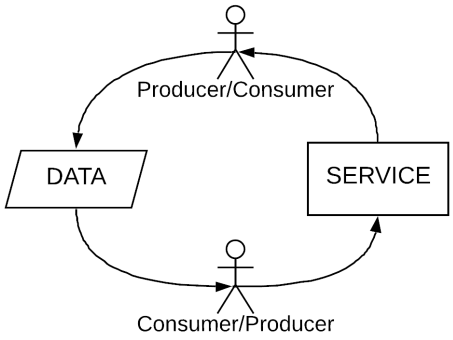


Figure 3.1: User cycle in the ecosystem

Data Sharing One of the two main abilities in the ecosystem is Data Sharing. This enables the creation of an implicit Data Marketplace in which anyone can access anyone's data after the execution of a transaction between the two parties. The system is always available and no censors can be made because no one controls data after it has been published. Anyone that finds himself in the ITS is able to share the data coming from a vehicle or from his personal device (e.g. smartphone) becoming a "Data Provider". Subsequently, anyone else is able to access this data with permissions granted through a payment or a priori, becoming a Data Consumer.

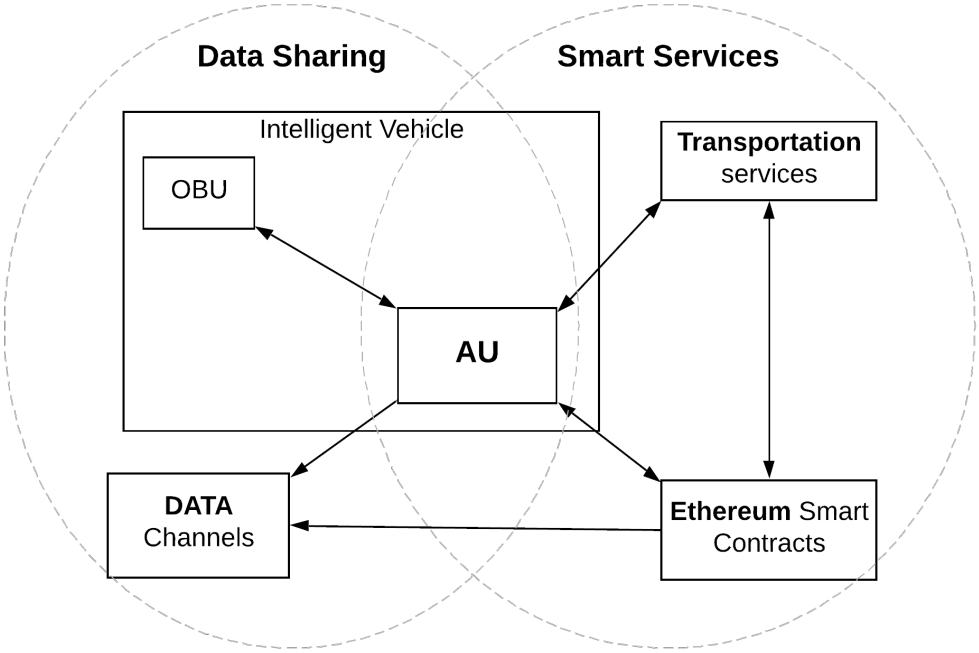


Figure 3.2: General ecosystem schema

Smart Services The second main ability that the ecosystem allows is the employment of Smart Services. These include services that follows economic transactions and also general transactions regarding communication between vehicles or infrastructure. All of them are backed up by the common way of use of Smart Contracts in Ethereum. This ability is used by transportation services that directly interact with users, by vehicle-to-vehicle or user-to-user services, and by third parties that operates in the ITS. This kind of user could be both Data Consumer and Service Provider enabling a service for an user that is Data Producer and Service Consumer.

User From both previous paragraph it is clear that the user is at the center of an ecosystem built around his data and his operations. In Figure 3.2 it is depicted a general schema that summarizes the idea of this work. The AU is placed at the center because it represents the interface an user is able to interact with and the way through he operates. This ecosystem is intended to give the control to the user by bringing him in through his data sharing and then by letting him execute an active role through smart services. The user is always both producer and consumer.

No Centrality In the context of multiple devices interacting between each others, such as the ITS or Smart Cities, the control of one or more central entities can result in positive or negative results. The increasing number of available devices that are able to depend only on their computational power, together with the development of powerful peer-to-peer architectures, incentives decentralization and its ability to give the right amount of "power" to users: power in the ability to manage their assets and to decide how to transact.

3.2 Requirements

The system must be realized following a decentralized architecture with a set of decentralized applications (dApps) and entities interacting between each others. The system consists of operative nodes acting as wallets and provider nodes acting as full nodes in the context of Decentralized Technologies. Operative nodes are the ones that play an "active" role in the system enacting operations and invoking decentralized nodes, which play the "passive" role of dialogue with the decentralized technologies.

3.2.1 Functional Requirements

- The system provides a direct communication with sensors placed on the user personal device or placed in a vehicle, thus obtaining environmental data coming from these ones.
- The system allows an operative node that obtained this data to communicate with different decentralized nodes that are peer in a network. This is achieved in order to store, validate and authorize the data. Hence the data is indexed and obscured, but made accessible for other users.
- The system determines whom can access a specified datum after a transaction between the requester and the server or after a specified declaration of the server.
- The system supports the execution of transaction in a blockchain and the use of Smart Contracts. To be able to do this an operative node, managed by an user, communicates with a passive node that is a peer in the blockchain peer-to-peer network and has the ability to publish transactions. This latter node accepts each transaction provided by the former node and participates to the blockchain consensus algorithm.
- The system enable users to create, invoke, update and destroy Smart Contracts or init existing ones, through transactions. By invoking specific Smart Contracts the system accept currency exchanges, also in "micro" quantities.
- The system enact exchange of configurable services between operative nodes with or without the use of currency transactions.
- The system allows to consume data shared by users to generate new services for users.
- The system allows any user to generate services through the data obtained by other users: service providers can be indistinguishably simple users or trusted control organisms.
- The system validates the data (e.g. location) of an operative node that requests for it.

It is possible to distinguish four main environments operating in the ecosystem. The following definition is not assumed to be limited by the technologies assumed, but, instead, other similar components can be added or can replace existing ones (e.g. Ethereum Swarm instead of IPFS).

3.2.1.1 Intelligent Vehicle

The user interface to the system is an application designed to work mainly inside a vehicle, possibly connected to the OBU and directly accessible by the user. The application, hence, resides into an AU, which represents a personal device (e.g. smartphone), and can communicate internally with the AU sensors to acquire data registered during the travel. Within the vehicle environment the application directly communicates with the OBU to receive vehicle sensors data. This environment is, basically, dedicated to acquire travel data coming from multiple sensors an registering various security and comfort events.

3.2.1.2 Data Channels

User data is indexed and accessed thanks to the use of personal channels that are created for each kind of data the user application is capturing. For each user an index is maintained and updated every new session with links to new channels for each sensors data kind. The use of these channels is intended to validate the data itself through immutability, but the storage is not always contemplated. Channels can contain actual data when the dimension allows to, otherwise the data can be referenced from another storage.

3.2.1.3 Ethereum Smart Contracts

The Ethereum blockchain is fundamental to let users to transact without the need of central authorities. One Smart Contract for each sensors data feature is strictly bonded to the user, represented in the ecosystem by an address. Then other general contracts are used, e.g. a token contract used in the ecosystem or a Micropayments contract, that are useful to anyone and not related to any specific user. Users can pay other users' data directly through Smart Contracts and then access those through Data Channels, each contract points directly to the index channel.

3.2.1.4 Transportation Services

Transportation Services here is used to depict a general environment in the ITS where services to drivers are provided by one or different entities that can physically operate into the vehicular network (e.g. motorway tolls, electric charging stands) or that can offer other related operations (e.g. Traffic Levels Indications). RSUs can be seen as the primary physical device that offer various services to drivers as shown with VANET implementations and users can make use of those through a direct devices communication. Trusted RSUs and devices can be also used to release a location certificate that a driver can use to validate users data.

3.2.2 Non Functional Requirements

3.2.2.1 Security

- **Data Confidentiality** - Data owned by an user and all messages are encrypted and/or signed using public/private key methods
- **Access Control** - Data access is controlled and only who has the rights can visualize and use the data
- **Anonimity** - Any user is identified only by his public key created randomly

3.2.2.2 Usability

- **Invisible Processes** - Data sharing is invisible to the user, which only starts or stops it and receives the (monetary) benefits of it
- **Transactions** - It is possible to easily execute transactions in DLTs with standard wallet clients
- **Execute Contracts** - Creating and using smart contracts allows to run every computable code that represents a contract without any intermediary

3.2.2.3 Reliability

- **Hardware Failure** - A failure from the user side is critical in transactions but it managed through a rollback. In the decentralized technologies p2p networks a single node failure does not halt the execution because of the replication.
- **Large number of Demands** - The decentralization allows multiple distinct nodes to receive requests for DLTs operations. Hence it is feasible to handle large number of demands
- **Data loss/corruption** - Data is replicated in every node, hence if a node corrupt some data all the other nodes will not

3.2.2.4 Performance

- **Data Throughput** - Data must be shared with a throughput high enough to allow the development of transport services using it
- **Transactions Validation time** - In Ethereum a block of transactions is produced each 15 sec (avg) and each block requires 6 blocks ahead to be considered valid (with a probability near to one)
- **Transactions Throughput** - 25-30 transactions/sec for Ethereum

3.2.2.5 Supportability

- **Protocol Modification** - Modifications in Decentralized Technologies must be adopted from nodes in order to participate into the network

3.2.2.6 Accessibility

- The system must be accessible by anyone that can participate in the ITS

3.2.2.7 Scalability

- System must allow every user to be able to share data and transact. With decentralization, scalability in terms of number of nodes is well supported because the number of nodes can always increase. In terms of number of transactions Ethereum cannot handle an always increasing number of transactions

3.3 Feasibility Study

Consensus algorithm are employed by nodes in decentralized networks in order to provide a shared, replicated distributed ledgers and to update them consistently over time. Types of DLTs affect the choice of the consensus algorithm and, since there are requirements regarding performance, accessibility and scalability, the system should rely on a public permissionless DLTs.

Public Accessible Data Data sharing is based on the principle that anyone can access anyone else data simply by obtaining the access rights. Not considering for the moment how to get this rights, the focus is directed on the data itself. This definition implies that data must be public but obfuscated, hence, since the system does not rely on a central entity, public DLTs are perfectly suited for this use.
To grant the property that anyone that plays a role in the ITS can enter the system, a permissionless DLT is required. A public permissioned DLT could be also used, with the hypothesis that one or multiple central authorities control this DLT (e.g. National Transportation Authorities), but in this case proper decentralization in the means of full power to control data and transactions is more difficult to achieve. With public permissionless DLTs any user is equal, with the same rights as the others and with full control over his operations, determined by shared protocols and deterministic actions (i.e. Smart Contracts).

Access to Services Smart services can be either local or accessible through internet. The assumption is that every user in the ITS has access to internet thanks to cellular radio technologies (e.g. 5G) or through a dedicated hotspot infrastructure in the VANET. Granted this assumption, every service in the system must be accessible by anyone. The Ethereum blockchain fits perfectly in this role because each Smart Contract can be always accessed and can be bonded to these smart services. Local services, where local means that consumer and provider are physically near and can communicate directly, must be handled with standard protocols for direct communication and errors recovery.
Smart Contracts are powerful tools, and their involvement in the ITS implies that every transaction in the system between two parties can be ruled and fully transparent for both of them. Trust in a central entity is substituted by trust in the Smart Contracts, i.e. trust in the consensus algorithm and cryptographic mechanisms.

Nodes in p2p networks Nodes defined above "passive" are the ones that act as full nodes in the context of Decentralized Technologies used. User nodes, i.e. the "active" ones, interact with the remaining part of the system sending messages to these decentralized nodes. This interaction is clearly of type request-response. User nodes send messages to nodes which replies accordingly. Decentralized nodes are meant to be geographically distributed and controlled by independent entities and the trust in them is given by the consensus protocol. There must be enough decentralized nodes in the system to allow user nodes to operate with the same transactions throughput in every condition. Adding a node is always feasible tanks to decentralized protocols used by DLTs.

Transactions throughput in Data Sharing In [32] authors study the IOTA feasibility in the use of vehicular applications. They conclude that the Tangle exhibits smaller transaction delays than existing public blockchains and that the performance of MAM channels are comparable with regular Tangle transactions. This result shows the feasibility of using IOTA MAM channels for data sharing in this thesis work because these support vehicular communications with negligible latency overhead. An important parameter is the dimension of the transaction to publish: to overcome a bigger delay it is possible to use IPFS as effective storage, with IOTA transactions pointing to IPFS objects.

Transactions security To ensure both authenticity and integrity of transactions, asymmetric cryptography is used. Users must be provided with a pair of cryptographic keys and sign transactions with their own private key. Decentralized nodes, when receiving a transaction, can verify the signature with the public key of the issuer and then send it to their peers in the network. The association with keys should not only be limited to users but also to devices and service providers that need to be certificated in the system. Hence a Public Key Infrastructure (PKI) is needed in this ecosystem and Certificate Authorities are likely to be Transport Authorities already present in the Transportation Network.

Distinction between user nodes and decentralized nodes A node participating to the Ethereum blockchain or IOTA Tangle consensus algorithm, i.e. a decentralized node, is limited by the Proof of Work. This means that not any user device representing a user node is able to execute a Proof of Work in a reasonable quantitative of time because of its limited resources. Hence an user device cannot be a blockchain or Tangle node, but it can only publish into them indirectly. Different is the case of decentralized nodes for IPFS. These are not limited by the computational power resource but by the internal storage resource. Hence it is rational to think of an user device that uses a portion of its storage to participate directly to the IPFS network. This case is ideal to work with when there is already an efficient implementation and a broad use of it, but actually a separated IPFS node is needed and should be treated as the other decentralized nodes (Ethereum and IOTA).

3.4 Personas

3.4.0.1 First Persona

Actors	Alice:User, Bob:Assurance Company Employ
Scenario	Alice starts his car and chooses to connect his smartphone via Bluetooth to the vehicle central unit. Alice has previously selected to share vehicle speed, steering position, brake pressure data and facial expressions recognition data from the app in her smartphone. Hence she opens the application and places the smartphone in front of herself. During her travel Alice's vehicle is hit from another vehicle. In a second moment Bob is able to access data from Alice's incident through his desktop client application. Bob reports directly data found at the time of the incident, stating that Alice's attention was sufficient and her eyes were completely open, and then also that she was respecting speed limits and traffic signals.
Analysis	Data coming from the vehicle and from the smartphone is automatically published in the corresponding MAM channel. Bob is able to access this data at the specified time of the incident because the assurance company address is stored in the Smart Contract relative to the channels.

3.4.0.2 Second Persona

Actors	Charlie:User, Diana:User, ChitChatCar:Smart Service
Scenario	Charlie starts his car and connects his smartphone via Bluetooth to his vehicle central unit. Charlie, then, selects his travel destination in the on board navigator and starts driving. Charlie was already subscribed to the ChitChatCar service, hence Diana is able to see Charlie's destination in her ChitChatCar app and requests a pickup from Charlie to get there. Charlie receives a notification from the app, sees Diana's actual position and accepts the request. When Diana and Charlie arrive to destination, the former pays the latter through a micropayment happening in their devices and a certain amount of currency is directly sent to ChitChatCar.
Analysis	When Charlie selects his destination, the coordinates are immediately sent to the specific MAM channel. The smart service is always listening to Charlie's channel thanks to a subscription made through Smart Contracts and has also a direct access to the datum. ChitChatCar shows Charlie destination in a map for other users that have to get there like Diana. A similar procedure happen for Diana position. The payment is made through another dedicated Smart Contracts for micropayments that automatically send a percentage to the Smart Service address.

3.4.0.3 Third Persona

Actors	Elliot:User, Moover:Smart Service
Scenario	Elliot opens the Moover app in his smartphone and picks a destination. The Moover app shows him a list of public transportation vehicles that can bring him to his destination. Elliot chooses a bus and takes the directions that Moover shows to get to the nearest bus stop. Moover also shows where is located the bus and the estimated time of arrival. Elliot takes the bus and connects his smartphone via Wi-Fi to the bus central unit. Using a wallet app Charlie pays the bus ticket through a micropayment.
Analysis	Moover smart services takes data from other users like Elliot that are on board of public transportation vehicles. The communication happening intra-vehicle is characterized by an exchange of micropayments and certificates. The latter is provided by the vehicle OBU to validate that an user is on board.

3.4.0.4 Fourth Persona

Actors	Frank:User, Gloria:Truck Company Employ
Scenario	Frank is in his car truck and has his smartphone connected via Bluetooth to the truck central unit. Gloria works for the same company of Frank, but she has a desk work. She analyzes Frank's truck data through a desktop program, especially to check the temperature in his container. When Gloria finds an anomaly and promptly alerts Frank. In the meantime Frank is driving by a motorway toll and is able to pay with a micropayment through the Wallet app in his smartphone.
Analysis	Data shared by Frank is directly accessed through the company account, hence Gloria can easily supervise the truck data searching for anomalies. Frank is also able to directly communicate with a motorway toll through WiFi, exchanging messages for the payments.

4 Architecture Design

Requirements established in the previous chapter work as driving factors for the architecture design presented in this chapter. Here a general schema of the system architecture is presented and each section is dedicated to all different environments in the ecosystem. The term ecosystem is used instead of system because of the presence of different environments that are not strictly comparable to sub-components. An ecosystem maintains the relationship between environments, but these ones can change their characteristics over time.

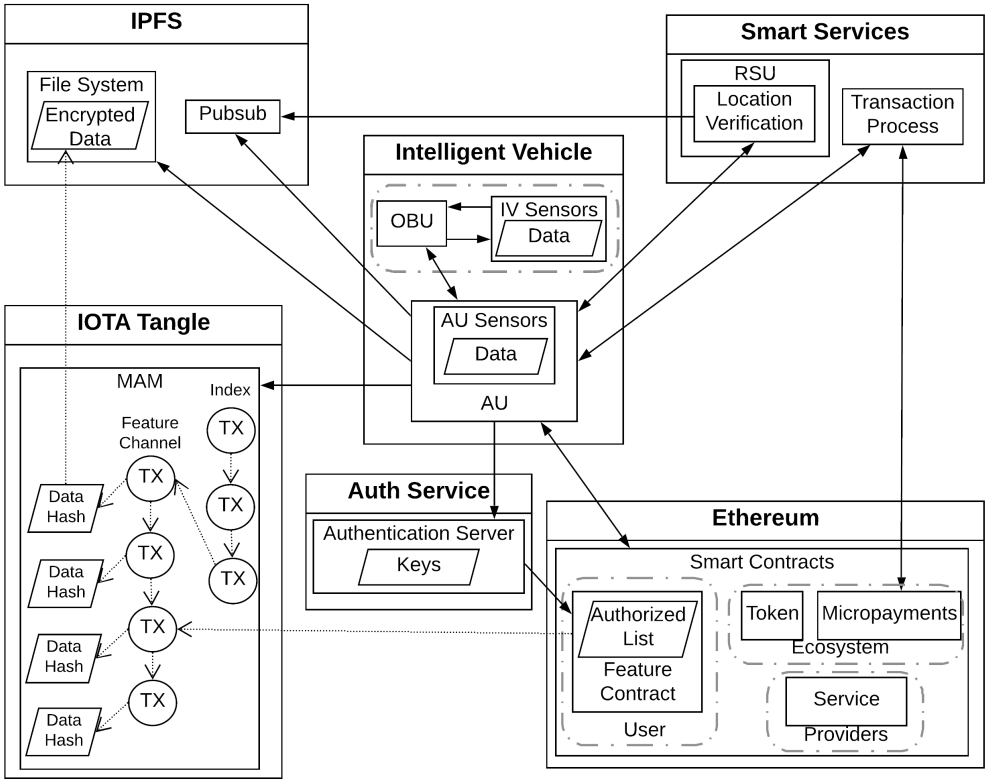


Figure 4.1: Ecosystem

The architectural style is based on decentralization, hence peer-to-peer. Ethereum, IOTA and IPFS work on a network of peers and nodes in the ITS communicate with these to participate to the network. The user's AU, indeed, is placed at the center of the schema in Figure 4.1 because it represents an ITS node controlled by the user itself. Thus the user is able to share his data and use smart services through the AU dialogue with decentralized networks peers and service providers. The schema is intended to give a first glance to the relations in the ecosystem and will be further explained in detail in the following.
Driving factors discussed above lead to a situation in which multiple distributed technologies must be used in conjunction both for Data Sharing and Smart Services.

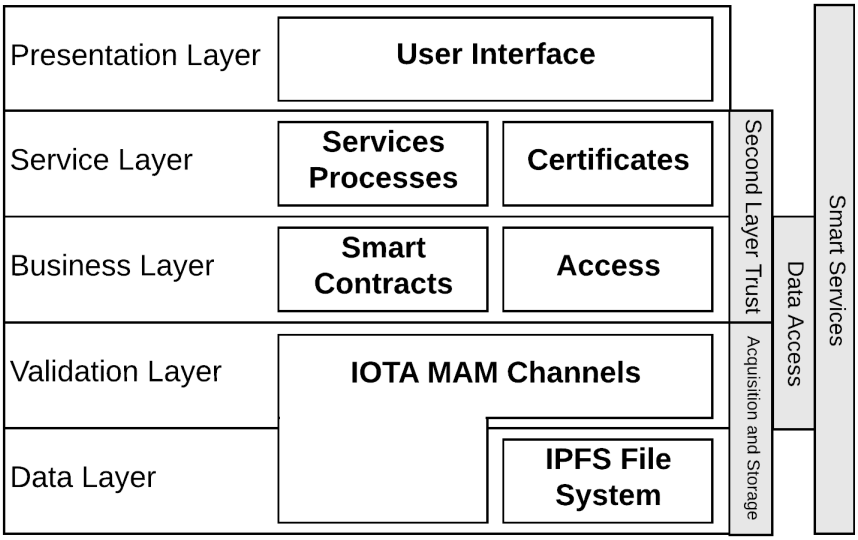


Figure 4.2: Ecosystem layered architecture

Here it is possible to distinguish four environments that explicate what has been showed as the main idea (section 3.1):

- Data Acquisition and Storage
- Data Access
- Second Layer Trust
- Smart Services

These environments are transversal to a layered architecture of the ecosystem, showed in figure 4.2. At high levels, the ecosystem can be partitioned in five layers that include all the components varying from Data Storage to User Interaction, where each layer depends from the layers below.

- **Data Layer** - Includes the IPFS File System where data is stored in a decentralized way and also part of the data stored in MAM channels
- **Validation Layer** - Dedicated to the data validation using IOTA Masked Authenticated Messaging channels
- **Business Layer** - Includes Ethereum Smart Contracts and the Data Access component
- **Service Layer** - Components regarding the certificates release process and the execution of services offered by third parties
- **Presentation Layer** - Includes the applications the user interfaces with to interact with smart services and operational settings

Later in this chapter will be also showed the point of view of a Data Consumer, which is a user that may not be physically accessing the ITS, but only access data produced by other users.

4.1 Data Acquisition and Storage

The core of the ecosystem is the Data Acquisition and Storage environment because it enables data sharing to every user in the ITS. At the time of writing, the most used data sharing services are based on a centralized services where the central entity has the power over data: power to access, give access and censor data without the user participation. On the contrary, in this work data is managed and shared directly by the user using Decentralized Technologies to store and access it.
In this environment two phases are distinguishable: Acquisition and Storage. Different kinds of data are acquired from the Intelligent Vehicle sensors or directly from sensors placed on the AU. After the acquisition, data is stored using IOTA Tangle and IPFS.

4.1.1 Acquisition

As explained early, user become a producer, in particular Data Producer. This data is directly obtained from the actions he performs in the ITS and the state of the vehicle he finds himself into. In this context the AU is considered as a gateway that receives information from all the sensors available.

- **Vehicle Sensors** - Vehicles are able to produce a great quantity of data through their sensors placed all over their structure. For instance, among all sensors, the ones related to vehicle status could be linked to a repair service, or sensors related to the environment could be used to weather or pollution measurements. Vehicle sensors are directly connected to the OBU through a physical connection and the OBU has the task to redistribute all the data produced.
- **AU Sensors** - AU sensors can be used in conjunction with vehicle ones or independently. These sensors are not directly linked to the vehicle itself but to the user. Sensors such as camera, microphone, GPS can track user's status for safety reasons or can substitute vehicle ones when it cannot be accessed (e.g. localization). AU sensors are directly placed in the device and data is acquired as raw from the application.

Sensors data acquired is directly collected by the AU application and it is ready to be stored. Data is then grouped as "features", meaning that an interpretation is given to each datum, e.g. vehicle position, tires pressure, facial expressions ...
The OBU and AU separation could be physical or logical: AU and OBU could reside in the same device, hence sensors would be exactly the same. In the architecture, thus, the connection technology between the two is not specified.

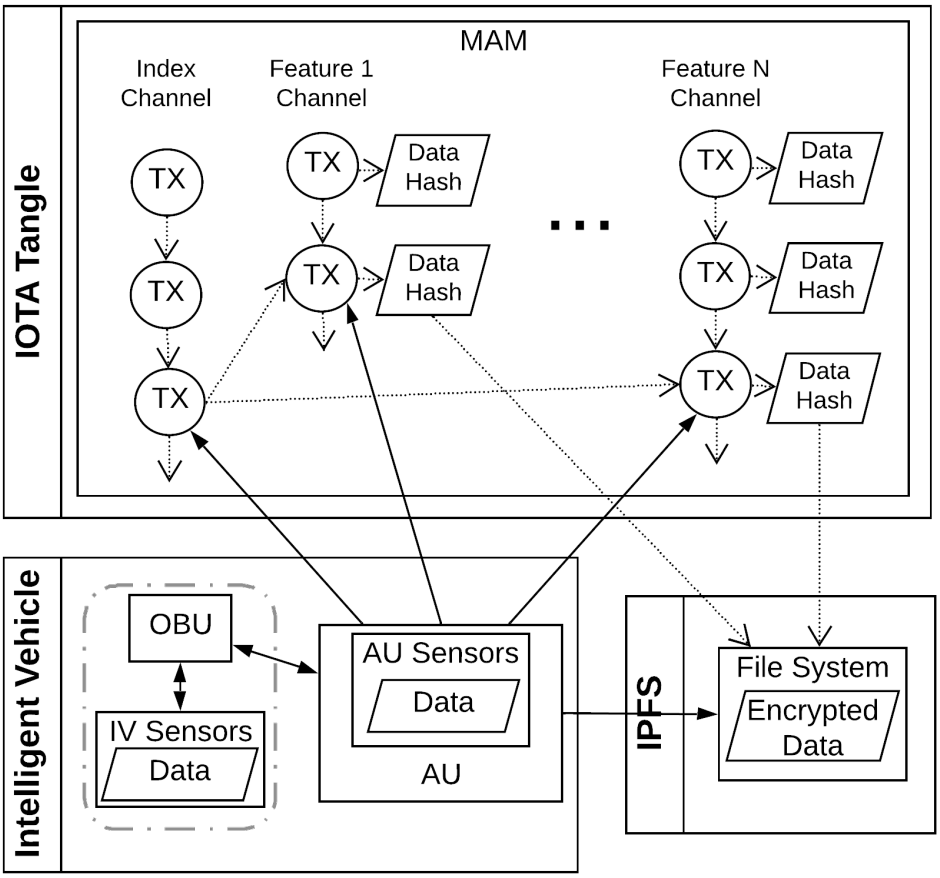


Figure 4.3: Data Storage Architecture

4.1.2 Storage

Data acquired by the sensors must be stored in a way that allows it to be reached and accessed by anyone that, in some way, obtained the rights after an agreement with the owner. The storage and validation of this data is achieved through the use of IOTA and IPFS. Both technologies duplicate the data in multiple peer nodes in a network, in order to decentralize the access. This gives the user the ability to really manage his production and to not depend from any central entity.

4.1.2.1 IOTA Masked Authenticated Messaging Channels

Data is published to the IOTA Tangle in order to be validated. The Tangle stores immutable informations that cannot be censored nor removed and will always remain online as long as the Tangle is online. Everyone can access this data, hence is necessary to obfuscate it to whom has no access rights. Masked Authenticated Messaging is a second layer protocol in which multiple channels are used to store information accessible by a third party. In the ecosystem each feature extracted by sensors data has its own MAM channel and all of this channels are indexed by a single channel that updates itself when new channel or sessions are initiated.

Index Channel In general, the entire MAM structure could be seen as a database, but the index channel in particular has the ability to convey the right informations to someone that needs to access them. Transactions in this channel contains references to the other channels in the form of strings. Each channel is indexed by the root of one of its transactions, whether it is the first one or a subsequent one. It can be seen as equivalent to a tree object in git.

Feature Channel A feature channel is a list of data or metadata in a chronological order. It is a log in which each transaction has the same structure and contains the data itself or an hash link to it. When data dimension is low enough (e.g. latitude and longitude) it is stored directly into the transaction. On the other hand when data dimension is not low enough, the transaction contains only a link to an IPFS object. Channels are obscured in order to provide access only to the righteous users, in fact each transaction in the channel is encrypted with a different key. Hence each user will have different MAM channels related to the kind of data they shared and an index channel (IOTA database) public to anyone that needs to access a specific information.

4.1.2.2 IPFS Objects

IPFS is actually used as the raw data storage. Each datum is represented by an IPFS Object, hence is referenced by its hash. No need of PoW implies that once an IPFS node stores an objects it is immediately available by all the peers in the network. This means that the datum is firstly published as an IPFS object and then referenced through its hash in a MAM transaction in an asynchronous process. The AU can easily process the IPFS object hash and others can use this information to get it: the object will be available for everyone immediately after it is published.

Fast Transmission In the case of a quasi-real-time data transmission, the use of MAM channels should be avoided because of their delays due to PoW. OrbitDB can be used instead of these to obtain data indexing and the creation of a log data structure similar to MAM channels. An user, then, directly uploads data to OrbitDB without the need of PoW and a third party can access it.

4.2 Data Access

Every user in the platform is able to share his data with various levels of access and everyone else that is eligible can access this data. In this context each datum access can be acquired through a monetary transaction or an agreement with the owner. The data access environment is composed by Ethereum Smart Contracts and an Authentication Service that act as intermediary for the user. Data owner may never see any interaction with other user that want to access his data, but only the transaction benefits, i.e. reward.
It is important to notice that this architecture pones the authentication service provider in a position in which he can access all the user data. A solution may be

the fact that the user fully trusts the provider (Traffic Network National Authority) or the use of more complex and less performing keys distribution mechanisms.

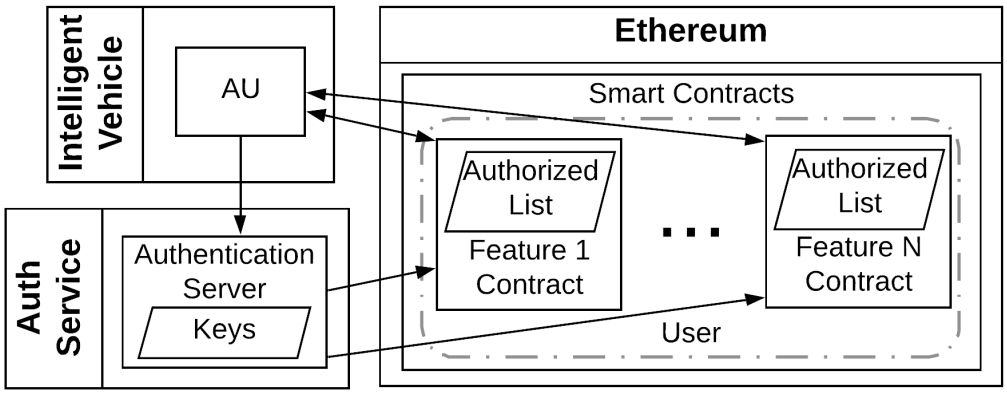


Figure 4.4: Data Access Architecture

4.2.1 User Smart Contracts

For each user a series of Smart Contracts concerning the data he shared are uploaded on the Ethereum blockchain. For each feature a Smart Contract may be implemented to control the access flow to the data itself. These Feature Contracts contain a "Menu" managed by the user that indicates the type of access and the costs he requires for that feature. But the most important structure in the contract is the list of authorized users. This is a list that associates an Ethereum address to a bundle of data and indicates that the address has the rights to access that data. This implies that the smart contracts allows a third party to directly obtain the rights to access a user data and store this information in a public list: hence anyone is able to check access rights.

Events Smart Contracts have the ability to emit events notification in broadcast to all the listeners. These events are fired once a change in the state of the contract happen or after an action is executed. The user's AU listen to the events broadcast of all user's contracts in order to notify the user or act for proper instructions.

4.2.2 Authentication Service

The Authentication Service consists in the simple process of checking the access rights in an Ethereum Smart Contract and then release access keys to who possess them. The architecture behind this process is the only one that assume a client server style communication. Here a server receives requests from different clients that want to access a specific user data, answering with a set of keys used to for MAM Channels and IPFS Objects.

Central Entity The use of a central entity is necessary in this decentralized context to unload the user's AU from the great amount of requests coming for keys, since it is not possible to encrypt a message through Smart Contracts without revealing the message itself.

4.3 Second Layer Trust

DLTs are based on the trust in the consensus algorithm, i.e. PoW, in the case of Ethereum and IOTA. This trust is at the core of data validation and transactions, because it is built in the main mechanisms underlying DLTs. When trust is given directly to a third party authority and when it is used to verify data that cannot be validated through the consensus algorithm it can be referred as a second layer trust. This mechanism can be used for intrinsic properties of the ITS that are possible to verify with proofs.

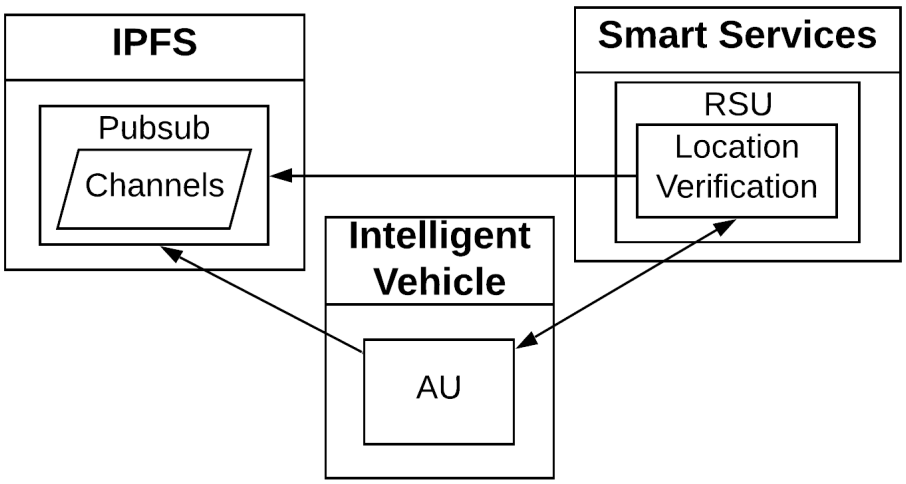


Figure 4.5: Second Layer Trust Architecture

One of the most important feature needed in the ITS is the location: knowing the user (or the vehicle he is into) position enables a series of context aware services. An important proof, indeed, is the Proof of Location that verifies the correctness of a user's claim to be in a certain position at a certain time.
In general Proofs or Certificates can be used to prove a certain user state property in space and time, providing a second layer of trust to the DLT trust. These proof, and in general the trust in what a user publish, allows the creation of services in a decentralized way without the need of a central entity that must watch over every aspect of the interactions.

4.3.1 Certificate Release Smart Service

A service that is crucial in this work consists in releasing certificates to validate user's data. Regarding the location feature, for instance, the verification results in a trusted RSU releasing a location certificate that states the veracity of a user's claim. Here trusted RSU means that it must be managed by a trusted entity (Traffic Network National Authority) and must be equipped with robust security methods. The communication consists in the AU sending a claim to the Verifier (which can be an RSU or another entity, such as a public transportation vehicle) which in turn verifies the correctness of the claim and sends back a certificate.

4.3.2 Publish Subscribe Service

A PubSub service is intended to provide a communication environment where users can share their certificates but also where they can be informed on traffic events and safety.

Basis for a Data Marketplace Data shared by users, as showed earlier, is completely accessible after an agreement with the owner. But the Data Consumer may not know what user must be contacted, hence a Marketplace is needed. The aim of this work does not focuses on the design and implementation of a Data Marketplace but all the tools needed for this case are. Multiple PubSub channels can be dedicated to publish certificates (e.g. location) in order to detect users to contact to gather their data.

4.4 Smart Services

Smart Services are meant to fully exploit smart contracts functionalities in order to provide feasible transactions into the ITS. The main operation consists in the payment of small amounts of currency, which is regulated through a series of contracts dedicated to micropayments. The user in this context becomes, from a data provider, a service consumer having the ability to access benefits through different smart contracts. Two kind of smart services exists and are differentiated by the use or not of Smart Contracts in the communication: Direct Communication Services and Smart Contracts Services.

4.4.1 Micropayments

Micropayments are payments that are not registered directly on the blockchain to avoid transaction fees and the waiting for transaction validation. After an agreement made between two parties through a smart contract they can start exchanging messages that indicates the current balance of currency among the two. Micropayments Contract events and operations, hence, are accessed by both the AU and the service provider.

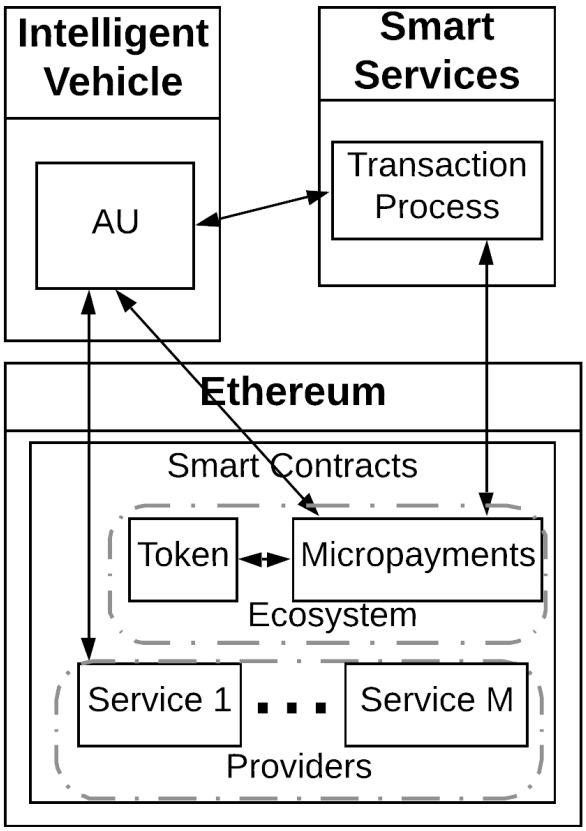


Figure 4.6: Smart Services Architecture

4.4.2 Direct Communication Services

Direct Communication Services are the one made between the user and a ITS infrastructure entity e.g. a motorway toll payment, the recharge operation of an electrical vehicle, a ticket payment. This service is characterized by the way it is accessed as there are multiple communication technologies involved: Bluetooth, WiFi, NFC... The constant is the transaction process that takes place between the two entities that consists in an exchange of messages and may also involve micropayments.

4.4.3 Smart Contract Services

Smart Contract Services involves every user definition given before. An user A may be both a Data Provider and a Service Consumer, whereas an user B provide a service by consuming data. This is achieved through the use of dedicated Smart Contracts that state the way interactions between user of kind A and user of kind B happen. Contracts are owned by user B and may include the access rights to some features produced by user A. User B then offers a service to A in exchange of a payment with a transaction directly executed by the contract.

4.5 Data Consumer Perspective

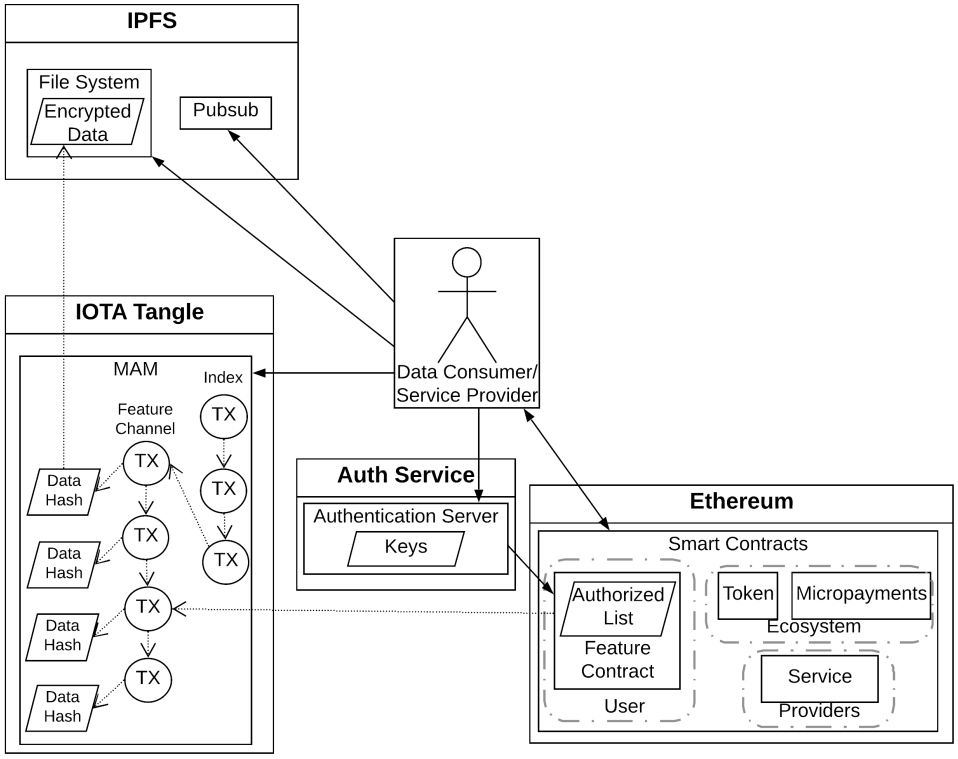


Figure 4.7: Perspective of a Data Consumer/Service Provider in the Ecosystem

A Data Consumer is an entity which is not directly involved in the ITS but may offer some Smart Services using data gathered. Here is not identified as a single application that is able to communicates in all environments showed in figure 4.7, but as an application that varies with the needs of the Data Consumer.

4.5.1 Data consumption

This kind of user is interested on accessing to data produced by other user in the ITS. He may use PubSub channels (Marketplace) to identify users that offer data that satisfies his needs. Then directly transacts with the related Smart Contracts regarding a specific feature in order to obtain the access rights. Thus, when requesting the keys to the Authentication Service, his rights ownership is stated in the contract and he will be able to access data. To do so the Data Consumer navigates through MAM channels finding the specific feature through the index channel. Each datum is referenced as hash to be accessed in IPFS as an object.
Data Consumer's application then communicates with Smart Contracts, Authentication Server, MAM Channels and IPFS at least for the first interaction, in a second time it may be possible to have a communication only with IPFS to directly gather data he can rightfully access.

4.5.2 Service provision

The Data Consumer may become a Service Provider when, using the data he obtained or not, offers services to all the other users. This may include a totally different system based on data acquisition and the communication with users in the ITS may happen in different ways (e.g. another application on the AU), but the service has to be based on smart contracts. The service benefits from the use of smart contracts for the ease of integration of payments and features authorized lists.

5 Implementation

In previous chapters has been given the idea of the ecosystem that is the main focus of this work, together with the architecture that covers the interaction with Distributed Technologies in the ITS. The ecosystem design pones the basis for the implementation described in this chapter.

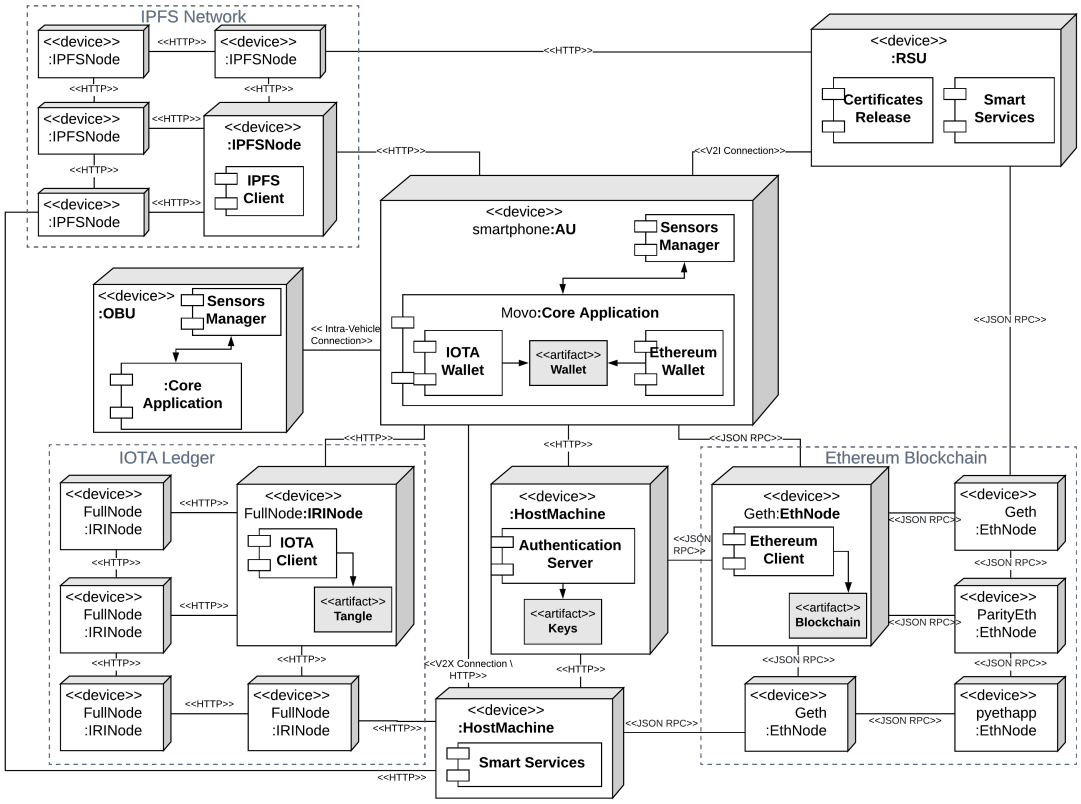


Figure 5.1: Ecosystem deployment diagram

Figure 5.1 depict the ecosystem deployment through a diagram, where the focus is given to the application that interfaces directly with users and to the nodes that allow the ecosystem perform. The core application, called Movo for an easy reference further in the text, is executed in the AU and pones the ITS user in direct contact with all the environments in the ecosystem.
This chapter will be subdivided following communication channels between the core application and the remaining components:

- **Sensors** - The acquisition of data coming from sensors placed in the vehicle or in the AU
- **IOTA Tangle** - Communication with nodes operating in IOTA to store and validate data acquired
- **IPFS** - Communication with IPFS nodes to store data

- **Ethereum Blockchain** - Communication with nodes operating in Ethereum to broadcast transactions
- **Authentication Server** - Service used to store keys to distribute to other users
- **Certificate Release** - Service that allows to validate data an user intentions
- **Smart Services** - Communication with third parties to receive services

Finally, last section is dedicated to the implementation in Android of the core application Movo, depicting all the current merits and limitations of the technologies used.

5.1 Sensors

Sensors are the entities that stay at the bottom of the implementation structure and of the ecosystem itself. Sensors are the main components that constitute Smart Cities and, more general, IoT systems and, when appropriately placed and correctly chosen, they produce data that can be interpreted in information regarding the context. In this work two kind of sensors are used to produce data related to an user: vehicle sensors and AU sensors. Movo directly communicates with the two components that collect sensors data for both devices (Figure 5.2).

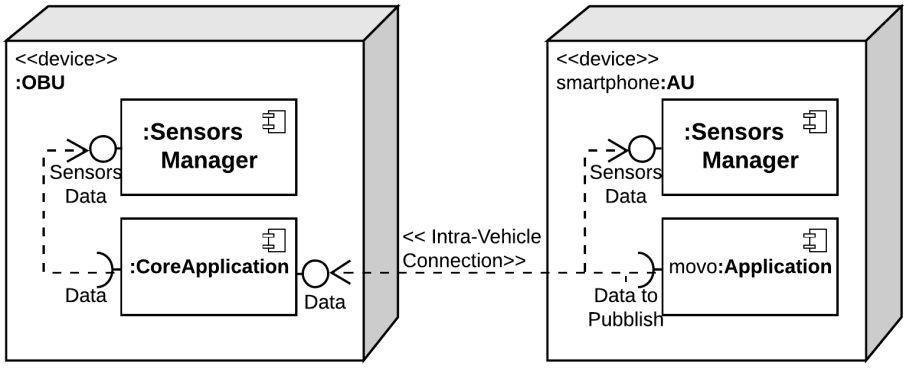


Figure 5.2: Communication between AU and OBU components to gather sensors data

5.1.1 AU sensors

The device where Movo is implemented, e.g. smartphone, must comprehend a component that manages all the data coming from sensors. The AU operating system usually handles this task, making feasible the process of inter-communication between Movo and Sensor Manger. Obviously different AUs probably have different sensors, but in the case of smartphones there is usually a set of sensors useful in a transportation environment: GPS, camera, accelerometer... The user is able to choose which sensor enable to gather his data directly through Movo and then, in the application itself, data will be subdivided in Features.

5.1.2 Vehicle sensors

In the case in which the AU is a part of the vehicle itself there is no difference between OBU and AU, but, when these two are placed in two distinct devices, they must be connected through an Intra-Vehicle communication. Bluetooth, USB or Wi-Fi Direct technologies could be used to pair AU and OBU. Through this communication channel Movo is able to request data to the core application of the OBU and the latter is able to respond with data gathered from vehicle sensors. Both core applications must support standard modules for Bluetooth, USB and Wi-Fi Direct communications in which the two devices pair and directly exchange packets containing messages. These messages must be opportunely formatted in JSON to ensure that data is joined by metadata.

5.2 IOTA Tangle

To publish data on the IOTA Tangle, Movo process must create a transaction and broadcast it to IOTA Nodes. Movo does not require to be a full node because it is not feasible to store all the Tangle in the device, hence it depends on other full nodes. The full process of data publishing is divided in two parts: Transaction Creation and Transaction Propagation. The former is entirely executed in Movo whilst the latter starts on Movo but finally takes place between all the IOTA nodes that receive the transaction.

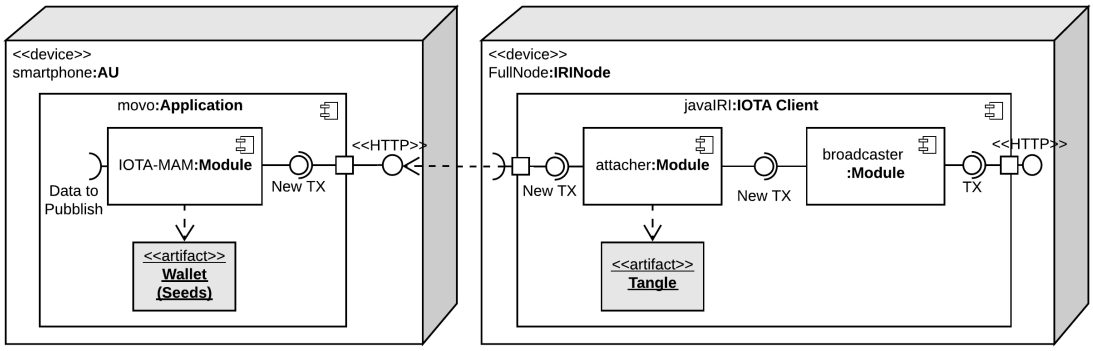


Figure 5.3: Communication between AU and a IOTA Node to publish data in MAM channels

5.2.1 Transaction Creation and Propagation

In order to update all the full nodes Tangles, new transaction must be correctly created and then broadcasted to all peers in the network.

5.2.1.1 Creation

The correct creation of a transaction that must be published in the Tangle is executed by a module that relies on IOTA and MAM client libraries. Data obtained from sensors, already subdivided in features, starts the process of transaction creation.

A Feature Data Packet represents the payload that must be attached to a MAM channel, it contains sensor measurements and a timestamp and it is represented as a JSON object. For each feature, the application maintains the corresponding channel references (seeds) and keys in the Wallet.
The creation process is showed in Figure 5.6 and consists in a sequence of operations:

1. The feature channel is represented by the root of the last transaction attached to this one. The root together with the key needed to encrypt the transaction are taken from the Wallet
2. If the packet is too big to be stored in a transaction the hash of this will become the new packet (used to reference the packet in IPFS)
3. The packet is converted to trytes (see 2.3.2.1)
4. A new payload is created following the MAM rules for Restricted channels (see 2.3.3.1) using the key obtained from the Wallet to encrypt the data.
5. The new payload, containing the MAM structure, compose the transaction to attach to the Tangle. Two tips must be referenced in this transaction, then these are requested to a IOTA full node.
6. Finally the PoW is executed to finish the transaction creation process.

5.2.1.2 Propagation

Once a transaction has been created, Movo forwards it to nodes in the IOTA network. The process of finding neighbors nodes in the network can be automatized using standard protocols or can depend on a service that lists a series of addresses that represent full nodes. When a full node receives a new transaction, this is firstly validated and then given in input to the process of attachment to the Tangle (see 2.3.1). Then the transaction is broadcasted to other IOTA nodes in order to let them update their Tangle.

5.2.2 MAM Channels Structure

As stated earlier MAM channels are organized in Feature channels and Index channel. Index channel in turn can be hierarchically decomposed in Session channels.
Links between transactions are given by roots: a transaction A has a link to another transaction B if A contains B's root (Recall that roots are the addresses used in IOTA to find transactions in the Tangle). Thus, the Index channel is updated only when a Feature channel or a Session channel is dropped in favor of a new one, but obviously the old channel reference remains in the old transactions of the index channel (nothing is ever deleted). The Session channel is updated at each session started by the user, indexing channels used at that time. With this hierarchical structure the only information to maintain is the Index channel first transaction root (stored in Smart Contracts).

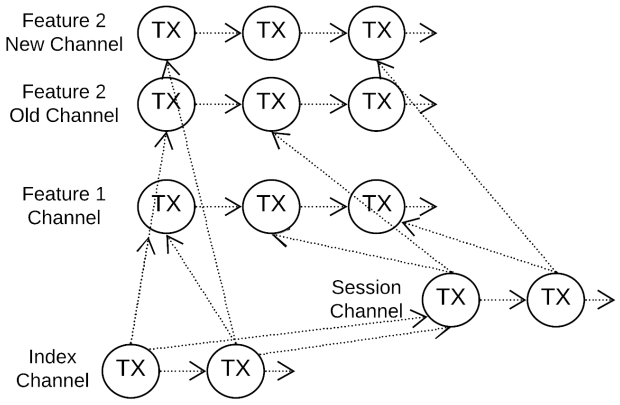


Figure 5.4: MAM channels structure

In Movo, Wallet must maintain only current roots and the seeds, in order to attach new transactions to the last one with the correct keys that allows to modify channels. After a wallet reset, roots can be restored knowing the index root, but seeds must be maintained and secured. Finally using MAM libraries makes it trivial to invoke methods that publish transactions in "open" channels, where open means that the application knows the last root and maintain the seeds.

5.3 IPFS

As stated earlier IPFS file system maintains all the data shared by the user and can be directly accessed without the need to wait for PoW. The AU could potentially represent an IPFS node itself but current implementations are not feasible to achieve this goal.

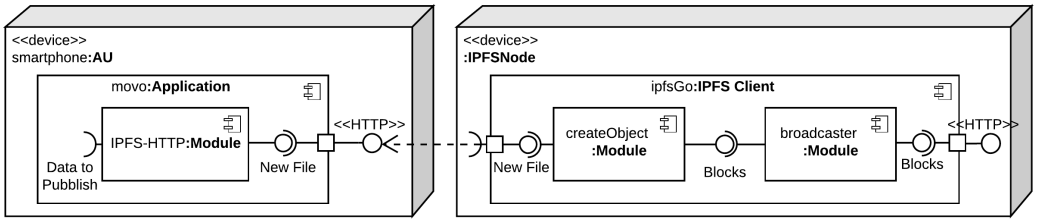


Figure 5.5: Communication between the AU and an IPFS node to store and distribute files

Once a file is stored by an IPFS node, it will be available to everyone until the node itself is online. To ensure persistence, blocks of this file are shared between other nodes. To access a file is only required its hash value. Thus the file initially represents a data packet and is directly sent from Movo to an IPFS node that takes charge of the object creation and broadcasting.

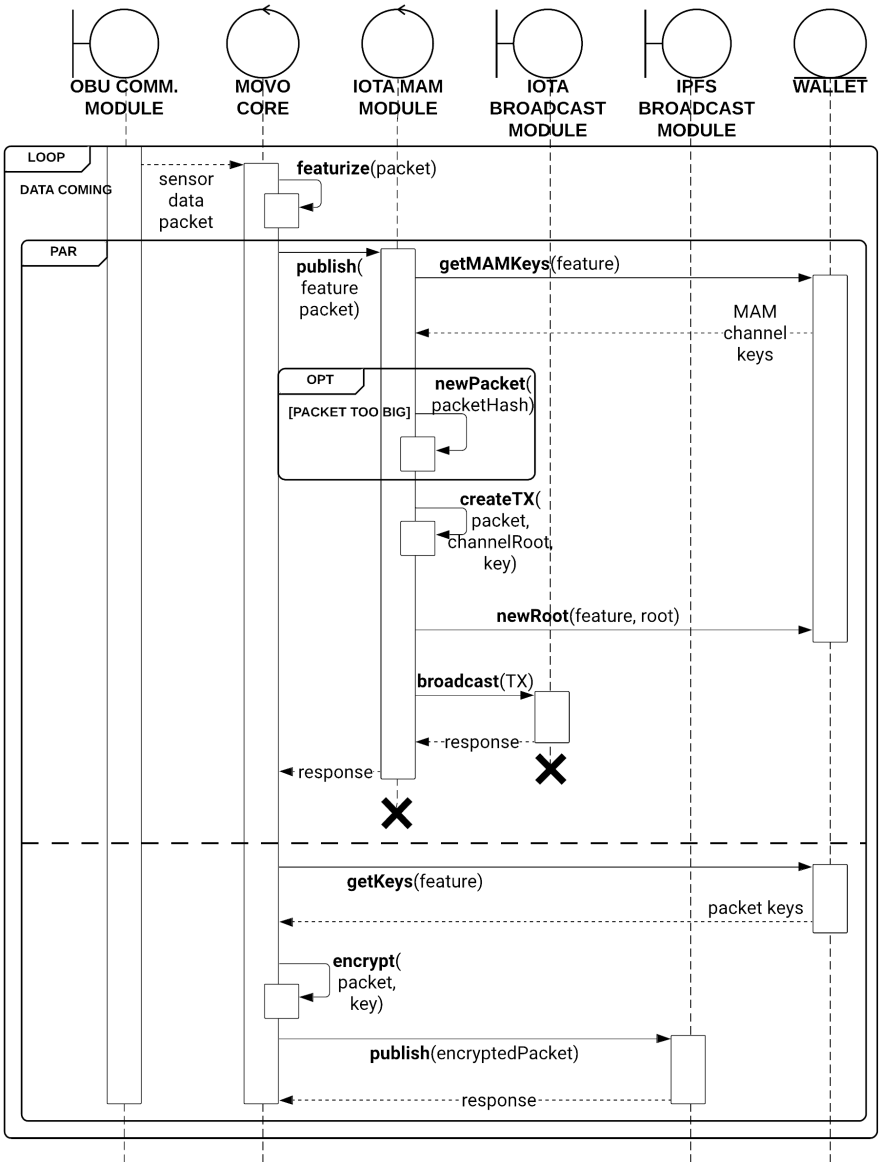


Figure 5.6: Sensors data publication Sequence Diagram

5.3.1 IPFS Object preparation and persistence

IPFS Object is not directly created in Movo but the raw (encrypted) packet file is sent to an IPFS node to process object creation and propagation. In this case the mobile node, i.e. Movo, does not have to be forced to follow all the protocols needed for IPFS, but the entire work is moved to the IPFS node itself.

5.3.1.1 File preparation

The data packet will be treated as a simple file in a file system. To be shared the file must be obscured, hence, as showed in Figure 5.6, the correct key for the encryption of the IPFS file must be obtained from the Wallet. Then the file is encrypted and sent through an HTTP POST to an IPFS Node.

5.3.1.2 Persistence

IPFS nodes receive files already encrypted, hence their only task is to propagate them on the network. An IPFS object is created from the file and will be referenced through its hash. Then all the blocks that compose the object will be broadcasted to other IPFS nodes: this allows the persistence of the object. Optionally objects can be stored in an OrbitDB, in order to have a data indexed also in IPFS.

5.4 Ethereum Blockchain

The Ethereum blockchain and a variety of Smart Contracts support the business logic of the entire ecosystem. Thanks to the property of distributed computing, Ethereum allows to have an ecosystem where no central authority manage transactions between users or between user and infrastructure. Smart Contracts are, indeed, central components where data access and smart services encounters, as can be seen in the layered architecture in Figure 4.2.
As well as the IOTA case, there is no need to fully store the ledger in the AU, hence Movo communicates with other Ethereum full nodes to propagate a transaction. This transaction can include a currency transfer but, most importantly, it can be a direct execution of an action in a Smart Contract.

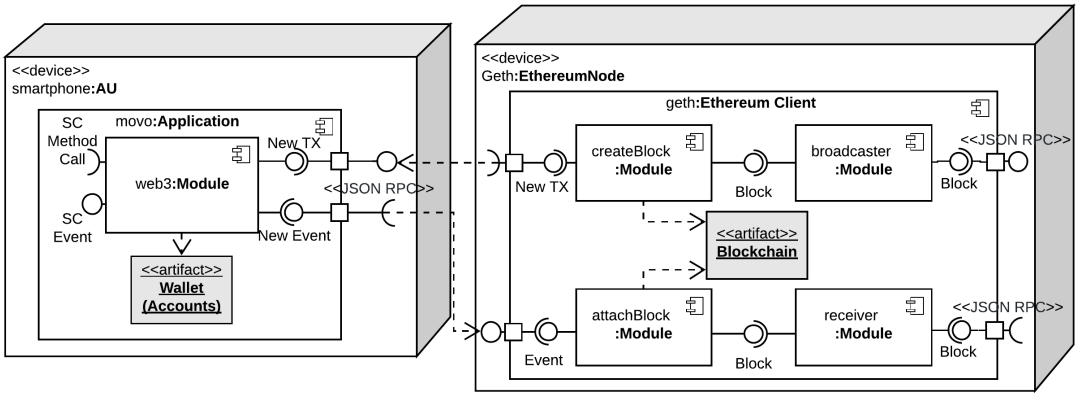


Figure 5.7: Communication between the AU and an Ethereum full node to execute transactions in the blockchain

5.4.1 Transaction Creation and Propagation

Similarly to the other two Distributed Technologies, to distribute an information between all the peers in the network, a transaction object must be firstly created and, then, validated and broadcasted.

5.4.1.1 Creation

A transaction object is created in Movo to execute an operation in the blockchain. Through a series of libraries usually defined as "web3" it is possible to make payments or execute Smart Contracts functions. Any user operates in the Ethereum blockchain

using an address and in Movo it is stored in the Wallet. To execute a Smart Contract method, a new transaction is created by the web3 module containing:

- The user Ethereum address
- The cost of gas fees to pay
- The smart contract method parameters
- A signature made using the private key

5.4.1.2 Propagation

Once a transaction has been created it must be propagated in the Ethereum Network. The communication with neighbors take instance through the use of JSON Remote Procedure Calls using HTTP Requests or Web Sockets. When a transaction arrives to a full node, this one starts the procedure of validation. A valid transaction is then listed in a block to start the PoW. When the node manages to fully complete a block, it updates its blockchain and then broadcasts it to his neighbors.
Another property of full nodes consists in receiving a new block where an event occurred in a particular smart contract related to the ITS user. This event is then notified to all subscribers, i.e. Movo, that can start a process in order to inform the user or take actions.

5.4.2 User Smart Contracts

In the Ethereum Blockchain each user will be related to a set of Smart Contracts that regulates the access to the data he shares. Each Data Feature will have a dedicated Smart Contract where the main component is a list that associate data packets to addresses, representing the data access ownership (Figure 5.8). These feature contracts are instantiated during the user "registration" thanks to the Factory design pattern. A Factory contract is used to create and deploy “child” contracts. The factory is used for storing the child contract addresses so that they can be extracted whenever necessary and new contracts can be created in the blockchain. Once a Feature Contract is created through a Factory it completely controls the ownership of access rights. This is possible through a the use of Smart Contract attributes and methods.

5.4.2.1 Feature Contract Attributes

Attributes are visible by all the users, but they ca be modified only by contract owners:

- **contractOwners** - set of addresses that can make changes on the contract (set other attributes)
- **packetCost** - cost of a single data packet expressed in wei. Together with the other costs and indexMAM it represents the "Feature Menu".
- **sessionCost** - cost of a single session expressed in wei

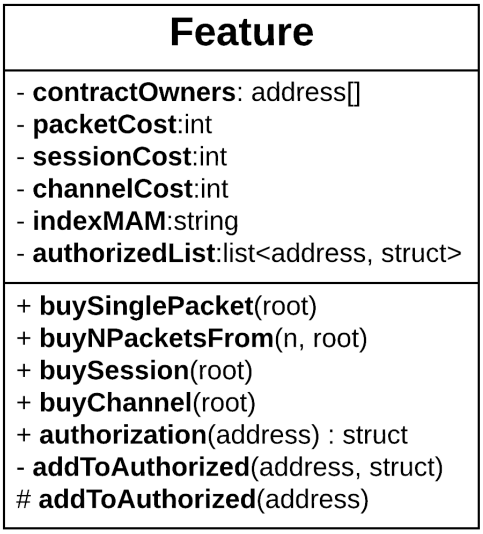


Figure 5.8: Feature Contract represented as a class. The symbol # represents that a method can be called only by owners addresses. Constructor and get and set methods are omitted.

- **channelCost** - cost of a single channel expressed in wei
- **indexMAM** - root of the index channel
- **authorizedList** - list that contains a mapping between an Ethereum address and a structure where are stored all the packets, sessions and channels owned bu the address

5.4.2.2 Feature Contract Methods

Only methods that allows to buy data are accessible by all the users. Remaining methods are accessible only by contract owners:

- **buySinglePacket** - method that registers the ownership relation between an address and a data packet (referenced by its root) after a payment
- **buyNPacketsFrom** - registers the ownership relation between an address and the N data packets linked in succession from *root*, after a payment
- **buySession** - registers the ownership relation between an address and the data packets linkedby a session referenced by *root*, after a payment
- **buyChannel** - registers the ownership relation between an address and the entire set of data packets in a channel referenced by *root*, after a payment
- **authorization** - returns all the packets an *address* has the rights to access
- **private addToAuthorized** - internal method used to modify the authorized list

- **protected addToAuthorized** - method used by owners to add addresses to the authorized list that have the rights to access every packet of the feature

Decentralized executed business logic conveyed through these properties and operations are the fundamental junction of all environments in the ecosystem. No third parties are between data provider and data consumer, but only these contracts that once deployed are immutable.

5.4.3 Micropayments Channels

Micropayments Channels (or State Channels) is a design pattern for instant blockchain transactions made off-chain. As transactions do not touch the blockchain, fees and waiting times are avoided in a secure way. In this work, an implementation based on μRaiden framework is used to create Micropayments channels. μRaiden is an open source framework used to implement ERC20-based free pay-per-use payment channels.
The implementation can be found in *https://github.com/miker83z/Mycropayments*.

5.4.3.1 Opening Channel

The procedure (Figure 5.9) starts when an user that wants to use a service creates a payment channel in favor of the service provider. The former is called Sender and the latter is called Receiver. In order to create a channel the sender must invoke a method in the Micropayments Smart Contract indicating the Reicever's Ethereum address and the tokens amount he wants to deposit in the channel. Both parties receive an opening channel confirmation event and, from now on, they can transact without the use of a Smart Contract.

5.4.3.2 Updating Balance

Once a channel is created, both sender and receiver can open a communication channel in which they exchange transaction messages. The sender makes a transaction by digitally signing a message that indicates the new balance (using the private key associated to the address used to open the contract) and then sends it to the receiver. The receiver responds providing the service requested and sending back his signed balance message (with a balance equal to the sender's one). This exchange takes place until the sender needs no more services or when the balance becomes equal to the payment channel deposit.

5.4.3.3 Closing Channel

Both sender and receiver are able to close the payment channel. One of them must invoke the closeChannel method in the Micropayments Smart Contract indicating the last balance agreed and the other party signature of this. The Smart Contract will internally check if the signature is valid and then will distribute the balance amount of tokens to the receiver and the remaining part to the sender.

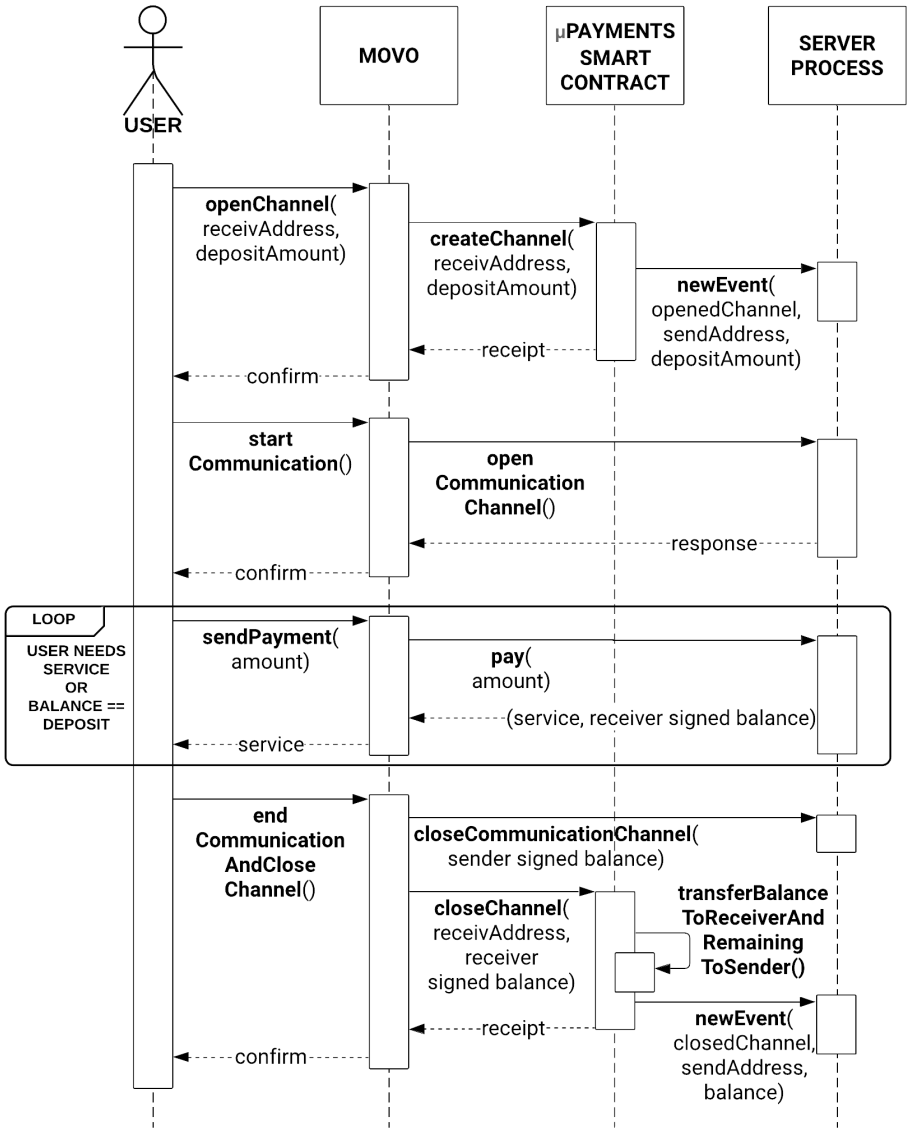


Figure 5.9: Mycropayments Sequence Diagram

5.4.4 Movo Token

The MOVO is a standard ERC20 token used to make payments inside the ecosystem. This token in involved in every payment explained until now, from data access rights to micropayments and designed also for Smart Services. Having a unique token for all operations in different environments allows the continuity in using services in the ITS, for instance an user may receive MOVOs for the data he shared and use them to pay a ticket bus.

5.5 Authentication Server

The authentication server is the service that provides access keys to eligible users. This service is completely based on the Feature Contract authorized list that represents the access rights ownership, i.e. bonding between an account and access keys. This service is the only one that includes measures of centrality in the ecosystem, but its role is needed because of the impossibility of Smart Contracts to autonomously distribute secret messages: being executed in every Ethereum node, everyone can look at the process of a message encryption.

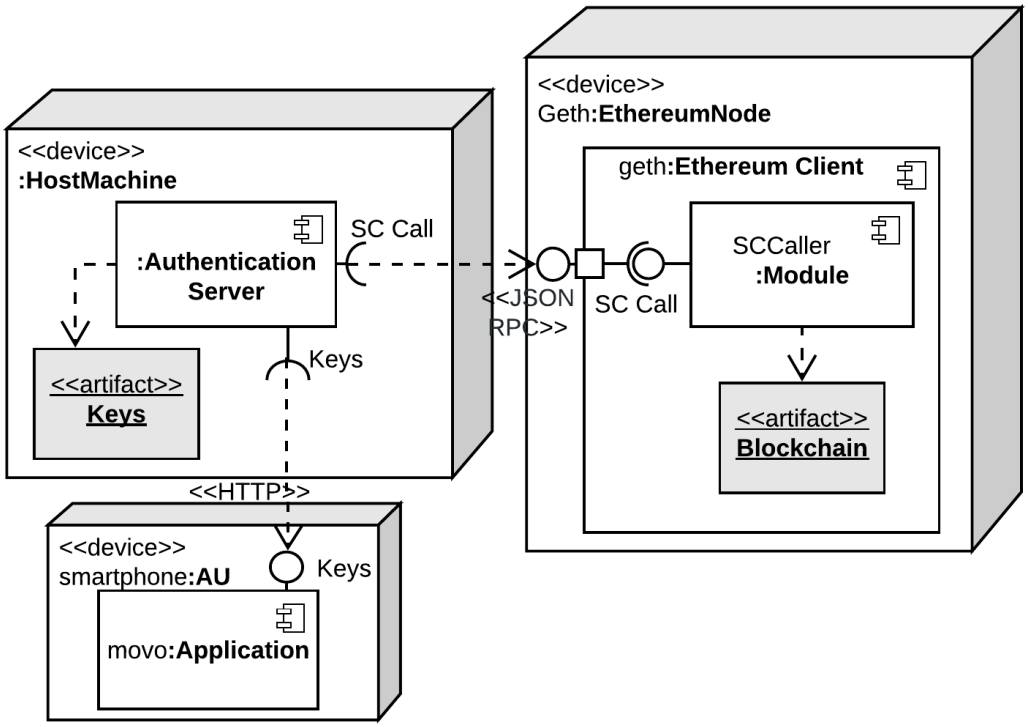


Figure 5.10: Communication between the Authentication Server and an Ethereum full node and Movo in order to provide data access services

5.5.1 Keys

For each data packet exists a key used to encrypt the corresponding IOTA transaction and the IPFS object. This key is produced by the Movo app and handed to the authentication server. The key used to encrypt a MAM transaction and an IPFS object is given by the hash of the concatenation between a master key and the root of the transaction in which the data packet is stored: SHA256(master-key‖root). The master key is a key generated randomly from a seed of 512 bits and it is sent to the authentication server through an HTTP POST: optimistically this operation is executed only once but can be repeated multiple times if it is possible to leave traces of the fact. The transaction root is always known a priori (before attaching the transaction to the tangle) during the transaction creation. This mechanisms follows the standards of Hierarchical Deterministic Wallets (BIP-32 [33]).

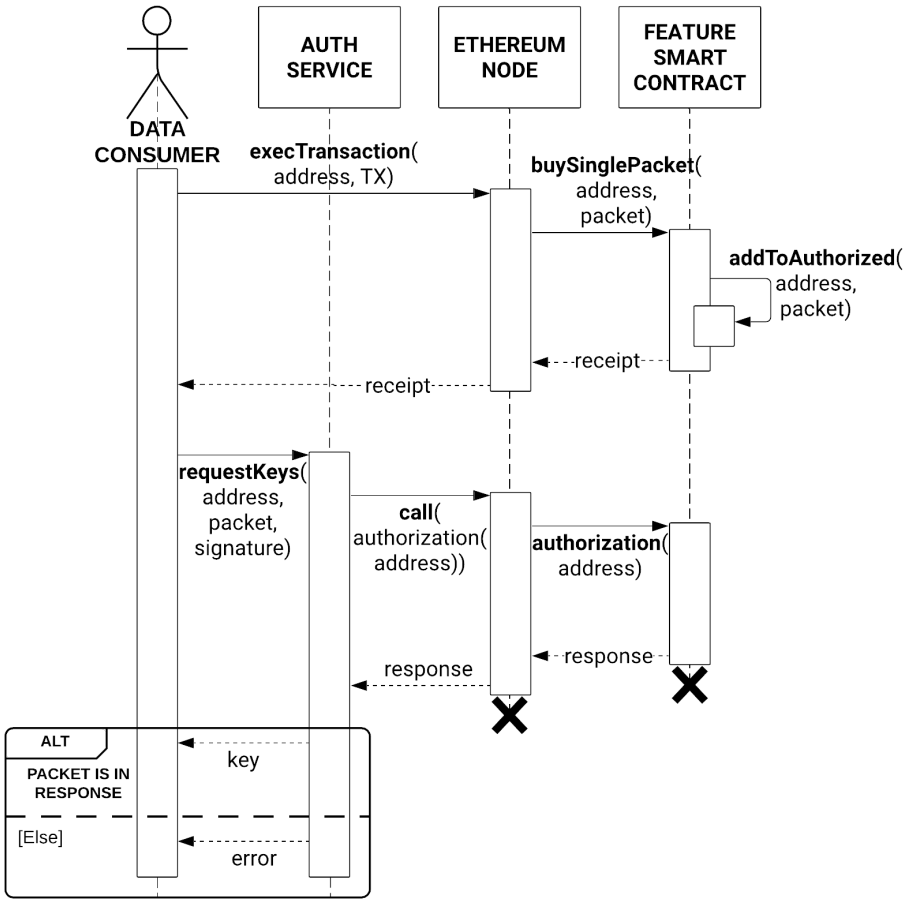


Figure 5.11: Data packet purchase and access request Sequence Diagram

5.5.2 Smart Contract access rights check

The Authentication Server communicates with Ethereum full nodes to check whether the Data Consumer that requested a particular access is eligible to get the key. The process is described in Figure 5.11 and based on components of Figures 5.10 and 5.13. Data Consumer sends a transaction to an Ethereum full node containing a method invocation for a Feature Contract. This method takes as input a data packet root and the address that is buying it, then it collects the token amount from the address account and, finally, it adds to the authorized list the address for that particular packet. Once the transaction has been confirmed, Data consumer directly contacts the Authentication Server to request the key for that packet he just bought, providing his address and a validation that the address belongs to him (a signature). The server checks the user's access rights in the smart contract by calling the authorization method that returns packets accessible by a specific address. If the packet the user requested is in the response, the server releases the key to the requester.

5.6 Certificate Release

Movo communicates directly with RSUs or other devices that act as trusted authorities to validate user's data. These devices validate a user's property through certificates release. This mechanism is used in order to avoid false data propagation in the ecosystem and to limit the purchase of incorrect data. A certificate may be released, for instance, by an RSU in order to validate the location of a user or by a vehicle that validates the user presence inside.

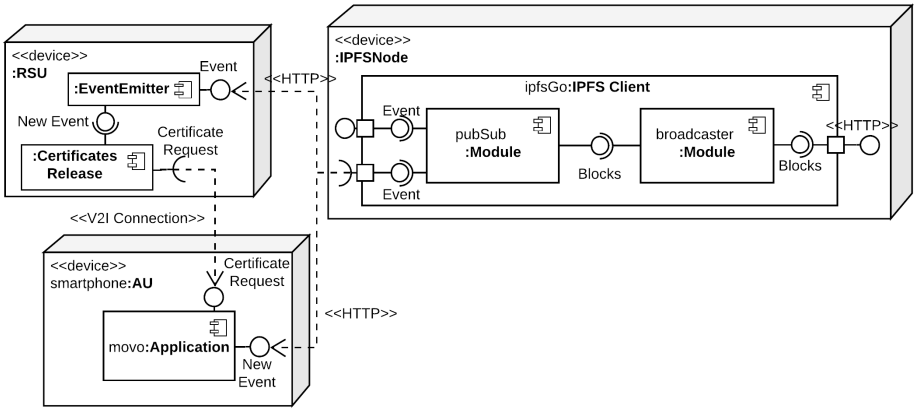


Figure 5.12: Communication between the AU and a trusted device (RSU). Both of them can communicate with a IPFS node to publish or subscribe to a channel

5.6.1 Certificate

The certificate must contain the property that is being validated and a signature from a trusted authority. To provide trust a Public Key Infrastructure is needed. Hence a certificate can be placed inside the data packets MAM transactions and publicly shared through IPFS pubSub channels.
Being used mostly for location validation, Zero Knowledge Proof of Location (section 2.5.2) is used when a certificate is published without encryption. A certificate, then, is a simple object containing an exact or summarized information about an user property that is validated through a signature.

5.6.2 Release

A certificate is requested directly by Movo to the trusted device through a V2X communication (Wi-FI, Bluetooth). The trusted device verifies that the requirements are true (e.g. a Bluetooth communication implies a strict area in which the requester must be located) and releases a signed certificate to the requester. Now both of them can publicly share the privacy protected certificate version (e.g. using zk-PoL) through IPFS pubSub channels, to which anyone can subscribe. A release of a certificate can be done also for traffic events witness (e.g. incidents) where RSU can validate messages broadcasted by users in the same way.

5.7 Smart Services

Smart Services are the second major branch of this thesis work and they are based on Data Sharing. These are intended as multi-purpose services built on top of the Smart Contracts presented above and custom ones. Smart services tap into different ecosystems to provide utilities to the ITS users, creating new infrastructures or offering simple utility processes.

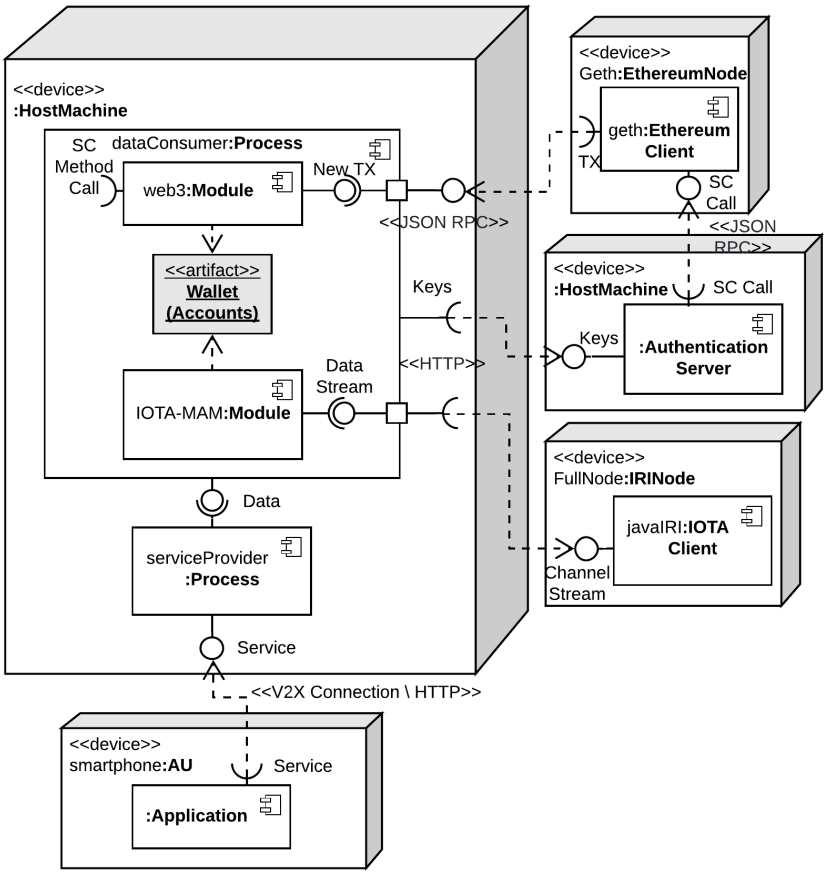


Figure 5.13: Communication between a host machine where a Data consumer/Service Provide process is executed

5.7.1 From Data Consumer to Service Provider

Smart services may consume users data to provide a service to other users. This process encapsulates the same components of IOTA, Ethereum and IPFS, working from another prospective. Data access purchase is described in Figure 5.11, in which the interaction with the Smart Contract is made possible thanks to the transactions sent to an Ethereum full node. Once a key or a set of key is obtained from the Authentication Server, the data consumer may look into the MAM (or IPFS) data

stream using MAM libraries. Through the keys he is able to decrypt transactions obscured from their owner. At this time data needed is consumed and data consumer is able to become a service provider offering services through processes that varies from case to case. This service process is accessed from an user's AU through a direct V2X connection (Wi-Fi, Bluetooth) or through HTTP requests (dedicated application).

5.7.2 On Road Services

On road services are direct communication services operated by devices/vehicles directly present in the ITS. These use V2X connections, where X indicate a device that is capable of offering Smart Services and the connection consists of short range communication technologies. Through these communication channels, messages exchanged contains different payload and may enact other operations regarding the service. To pay these services, Micropayments are used as showed in Figure 5.9. A set of plug-in modules may be used in Movo to accommodate different cases of services, for instance:

- **Parking and Bus tickets** - Messages are exchanged through NFC to make a micropayment
- **Motorway tolls payments** - Messages are exchanged through Wi-Fi Direct and while waiting in line
- **Limited Traffic Area** - Access to these areas is payed associating the license plate number to an Ethereum account in a Smart Contract
- **Pay-per-Drive** - Rented cars can be payed during the use through the communication between the AU and the OBU considering, for example, only kilometers traveled

These examples are all based on micropayments and in compliance with the technologies used in this work. Hence the implementation of these "plug-in" consist in processes of choosing the communication technology and transferring signed messaged through the channel. Here also a PKI is needed for signatures.

5.7.3 Smart Contract Based Services

Smart Contract Based Services are the ones where the real change from Data Consumer to Service Provider happen. Users that provide these services can transact with other user through ad-hoc Smart Contracts, implying that the service payment is done directly on-chain when contract methods are called. Data can be used to acquire knowledge of a particular area of the ITS and then a service can be built around this knowledge using Smart Contracts as Business logic.
The other important feature is the service application, that has to give an interface to access the service to the user. Everything else is integrated in the ecosystem, providing the Service Provider with a set of off-the-shelf tools for data, transactions and communication with users. The provider's app, indeed, uses the same components as the Movo app to communicate with Ethereum for Smart Contracts and eventually

with IOTA and IPFS for data. The service logic may run in a dedicated server maintained by the provider user. Some instances are explained in the Use Cases (section 6.2).

5.8 Android Movo App Implementation

Movo's implementation consists in an Android application developed considering the ecosystem presented in this work. The Android minimum API level used is the 21st, hence Movo is compatible with android devices mounting Lollipop 5.0 or superior. A Components Diagram is shown in Figure 5.15 and it is used for further comprehension of the implementation in this section. IPFS and Authentication Service communications are not implemented because of their trivial and unnecessary use.

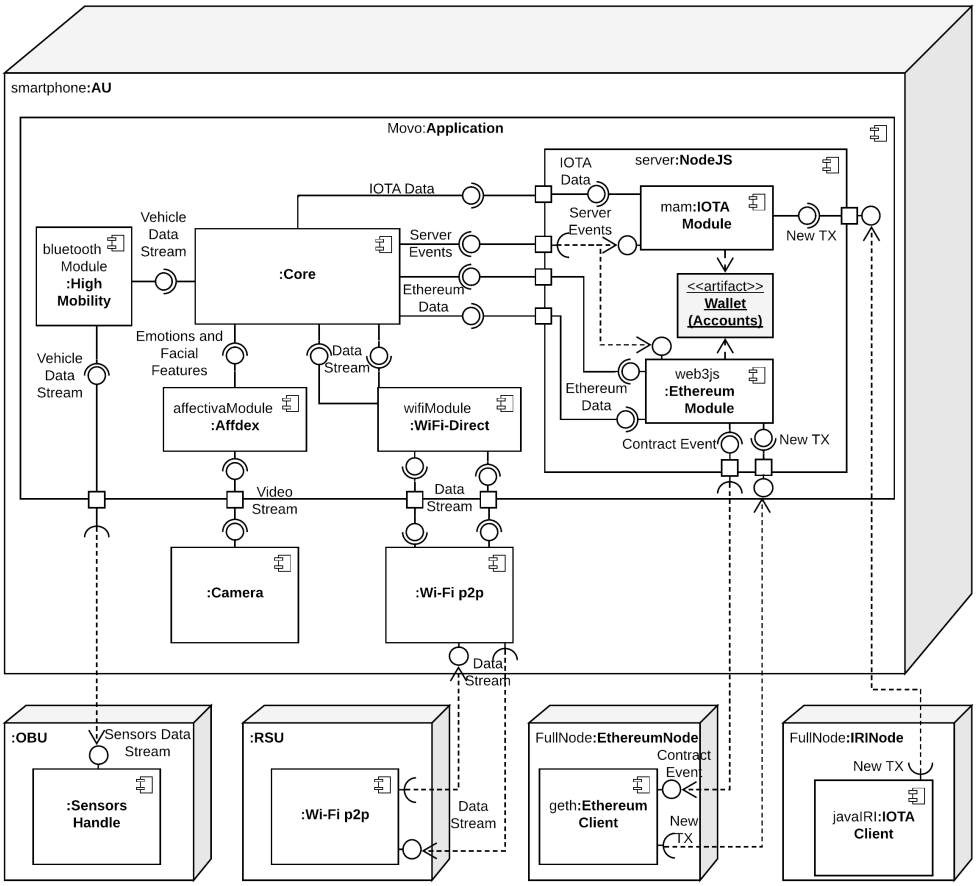


Figure 5.14: Movo Android App Components Diagram

5.8.1 Core module

Movo's core module controls every other module and the information on screen, i.e. the User Interface. This module is implemented using React Native, an open-source mobile application framework created by Facebook, used to develop applications for Android and iOS. As the name suggest React Native is like React, but it uses native components instead of web components as building blocks, hence the programming language used is Javascript. The core module executes two main functions, it acts as:

Gateway Modules engages dialogues between each others through the core. Information coming from the smartphone sensors are directed to the Core and then routed to the appropriate module for communication with external services.

User Interface Thanks to the React Framework the UI is easy to implement and it is composed by a combination of Components. Each component is a class and can be exposed on the screen.

5.8.2 High Mobility module

In the Inter-vehicle communication Movo communicates with the OBU. This dialogue is implemented through the High-Mobility module and the use of Bluetooth. High-Mobility [34] is a platform for apps working with personalized car data where tests are done in a simulation environment and access to data is verified from multiple carmakers using a standardized connected car API. Through these API it is possible to access different vehicle sensors data after a Bluetooth pairing, but also send command to control the vehicle. The simulation environment consists in a Web Browser that simulates a car actions, managed through the user interface, and then the device connected, e.g. Movo, receives or sends commands via Bluetooth. This module is completely written as a Java class.

Communication with core module Bluetooth communication channel is mainly used one way, allowing only incoming messages and never sending commands to execute vehicle actions. From this module, at regular intervals, the core receives JSON objects that contains data registered from vehicle sensors and timestamps. From these objects Features are created in order to be published.

5.8.3 Affdex module

For security purposes not only vehicle status data can be shared but also driver's. Affdex is an emotion measurement technology developed by Affectiva [35]. This software is able to recognize human emotions based on facial cues or physiological responses using the Facial Action Coding System (FACS), the most comprehensive and widely used objective taxonomy for coding facial behavior. A dataset of millions of faces has been used to train facial action and emotion classifiers, resulting in a automated facial coding based on four main components: face and facial landmark detection, performed using the Viola-Jones face detection algorithm; face texture

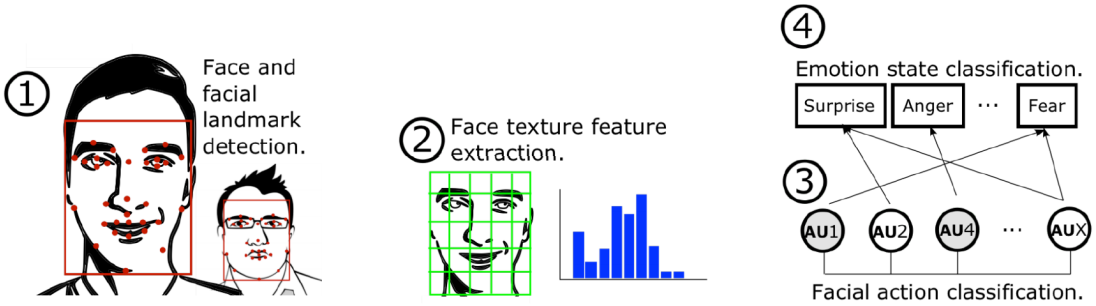


Figure 5.15: Affdex pipeline. 1) Detection of faces and localization of the key facial landmarks. 2) Extraction of texture features using HOG. 3) Classification of facial actions. 4) Modeling of prototypic emotions using FACS.

feature extraction; facial action classification, where Histogram of Oriented Gradient (HOG) features are extracted from the image region of interest to detect actions; emotion expression modelling, based on combinations of facial actions. Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are used provide accurate, real time estimates of emotions on mobile devices. Affdex SDK allows to register videos at 10 frames per second on mobile devices to detect emotions and facial measurements.
Affdex library is used in Movo to check driver's emotions and facial expression, being useful to detect behavior anomalies and making transparent traffic accidents. Affectiva module directly connects to the smartphone camera and starts analyzing 10 frames per seconds of raw footage returning various emotions and expressions indices. Eyes closure, attentions, anger, surprise and others are decoded into values between 0 and 1 from all the faces framed by the camera, together with face appearance and measurements to identify users. This module is completely written as a Java class too.

Communication with core module Camera raw footage is directly processed through this module using algorithm described above. Once a set of pre-determined values are registered for facial expressions and emotions these are encoded in a JSON object and sent to the core module. The core module groups all these object coming in sets each 10 minutes and then publishes an unique transaction containing a single JSON object with multiple detections and timestamps.

5.8.4 Wi-Fi Direct module

Wi-Fi Direct module exploits Android's Wi-Fi Direct libraries to manage p2p connections and communications. Through this module Movo discovers peers, connects to them and communicates through sockets. This module is used for the interaction with devices such as RSUs, hence different kind of messages, i.e.

location certificates requests or micropayments receipt, pass though this channel. This module, hence, is fundamental for Smart services that uses Wi-Fi Direct.

Communication with core module Messages may vary in contents but are always encoded in JSON. Coming messages are directly sent to the core that creates a response to send to the other device. The response may depend on web3 methods, thus an interaction with the NodeJS module from the core may be enabled in this case.

5.8.5 NodeJS module

Communication between NodeJS module and Core module can be seen as a client-server architecture. This module completely runs NodeJS programs, especially used to interact with Ethereum and IOTA nodes using javascript libraries. Ethereum's Smart Contracts requests are done through the web3js library as well as operations with accounts and instances of contracts events listeners. IOTA MAM js library is used to directly publish data to the Tangle. In DLTs accounts private keys are extremely important, then both Ethereum and IOTA modules maintain them in an internally secured Wallet.

6 Validation

The ecosystem has been analyzed entirely in its design and its implementation in previous chapters. This chapter is focused on the ecosystem validation, presenting an assessment on performances and applications in the real world. In particular, the first section will be dedicated to the use of IOTA in a mobility scenario such as the ITS, whilst the second section will show some use cases based on the ecosystem.

6.1 Performances Evaluation

Data sharing covers the entire set of environments presented in the ecosystem, hence the ecosystem performance evaluation must assess whether or not mechanisms that control sharing present bottlenecks. Assuming that is feasible to deploy enough IPFS nodes in order to let anyone store his data (section 7.2), the operation of data storing in IOTA remain the one that can cause performance issues. Publishing transaction in MAM channels is, indeed, the primary action that leads to all the benefits in term of transactions and services into the ecosystem.
In this section latency measurements are evaluated in different contexts and compared to provide a prospective on actual IOTA network capabilities. The results will be showed in plots and described together with the cases they bring up.

Testing equipment In order to provide a well rounded perspective on transaction creation and publishing in IOTA, various devices and network nodes have been used for testing. Since one of the major influence in performance is given by the PoW, varying the computational power is necessary to check differences. But in IOTA case, it is also possible to test multiple PoW difficulties switching networks.
To obtain data two kind of devices have been used:

- **PC** : Intel Core i7-6700HQ CPU, 8 GB RAM
- **AU**: Xiaomi Mi A1, Qualcomm Snapdragon 625 MSM8953, 4 GB RAM

Furthermore, IOTA nodes used differ in the kind of network they are in:

- **Mainnet**
  - Provider1: *http://node.deviceproof.org*
  - Provider2: *https://nodes.thetangle.org*
- **Devnet** - Provider: *https://nodes.devnet.iota.org*

In particular, the Mainnet is the IOTA network used generally as the main one where real transaction happen and the token has monetary value, whilst the Devnet is a network used for development purpose where the token has no monetary value. One of the differences between the two network is the PoW difficulty: Minimum Weight Magnitude is the number of zero trits at the end of a transaction required for it to be valid and is set to 14 in the Mainnet and 9 in the Devnet.

6.1.1 Simple Transaction and MAM channels

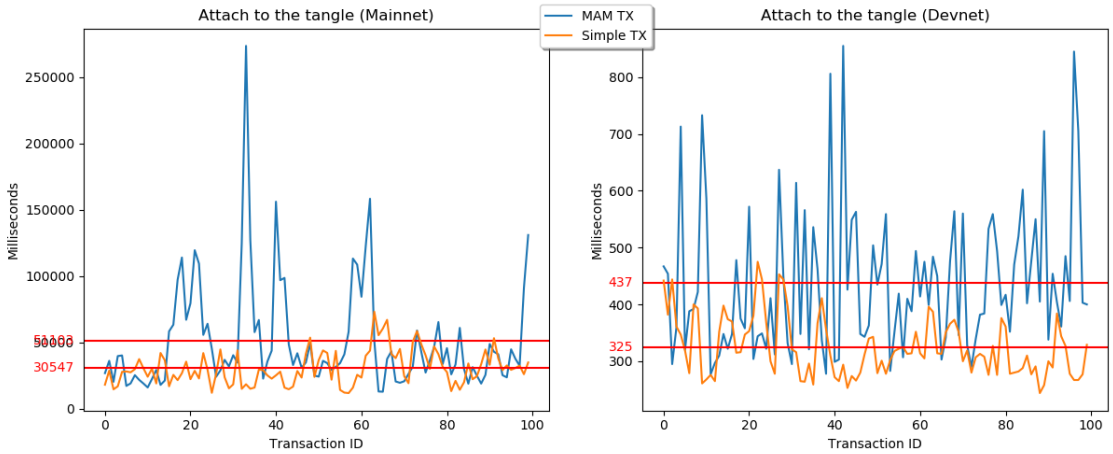


Figure 6.1: Latency in the process of attachment to the tangle (find tips + PoW) for both Mainnet and Devnet in PC

Given the main role of MAM channels, the first test compares the latency in publishing on the Tangle the same payload as a simple transaction and as a MAM transaction. The difference between these transactions is that for the MAM one a data structure is needed (section 2.3.3.1), adding an overhead in dimension. The payload dimension is 2187 trytes (will be the same for all tests) because it is the maximum that a single transaction allows. The result consists in a single Simple TX and four MAM TX, because of the structure overhead that require almost three entire transactions. Figure 6.1 shows the latency in the process of attachment to the tangle. This process comprehend two sub-processes:

- **Find tips** - To attach a TX to the Tangle it is mandatory to reference two other transactions called tips (section 2.3.1). These tips are provided by the IOTA full node that stores the Tangle.
- **Proof-of-Work** - Last step consists in perform the PoW for the TX in order to be validated. This is done in the device itself.

IThe plot on the left shows the time required to fully complete the process of attachment in the Mainnet, with the device used being the PC. Curves are obtained considering the latency for each of the 100 transaction attached in succession: orange curve represents simple transactions and blue curve represents transactions published in the same MAM channel. Average latency for Simple TXs is 30.5 seconds, while for MAM TXs is 51 seconds in the Mainnet. This clearly states how the same payload is published faster when it is not in a MAM channel. This is because of the fact that the payload generates four transactions in the case of MAM, hence requires more tips and more PoW. The high variability in the MAM TXs case (maximum value is more than 250 seconds) may be attributed to the latency of the IOTA provider in releasing tips.

The plot on the right shows the same data, but in the case of using the Devnet. Here latency is reduced by an order of magnitude of 2, in fact for Simple TXs is 325 milliseconds and for MAM TXs is 437 milliseconds. This is the resulting effect of a decreased difficulty in PoW.

6.1.2 IOTA full nodes performances

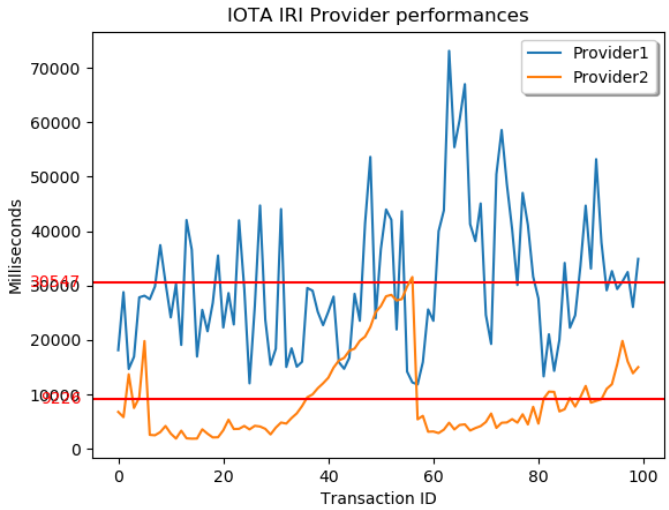


Figure 6.2: Latency comparison between two IOTA providers

Another important factor to consider in this evaluation is the performance difference between IOTA full nodes. As shown in Figure 6.2 the same task may be performed differently by two different providers. This may depend by the difference in resources and by the fact that a node with an higher degree on the network may receive more transactions from others, implying that it may provide tips faster to clients. For this test, the transaction to attach is the same Simple TX as before. In the plot, both providers work in the Mainnet, but Provider1 is relatively "unknown", while Provider2 is one of the most used node to operate in IOTA. This means that Provider1 has more resources to provide to clients but less tips because it receives less transactions, while Provider2 provides tips faster, but has limited resources for each client. The result is that Provider2 allows to attach transaction in 9 seconds on average, while Provider1 in more that 30 seconds on average. The downside of Provider2 is that it allows a limited number of request due to its limited resources, meaning that using it has not always been possible during testing (Provider2 is the one used regarding Mainnet measurements).

6.1.3 Comparison between PC and AU

The most important evaluation consists in comparing the latency registered for attaching a MAM transaction using the PC with the one registered using the AU. These measurements mark the difference in using IOTA from mobile devices or PCs. The difference here shows the impact of computational power in this context. In

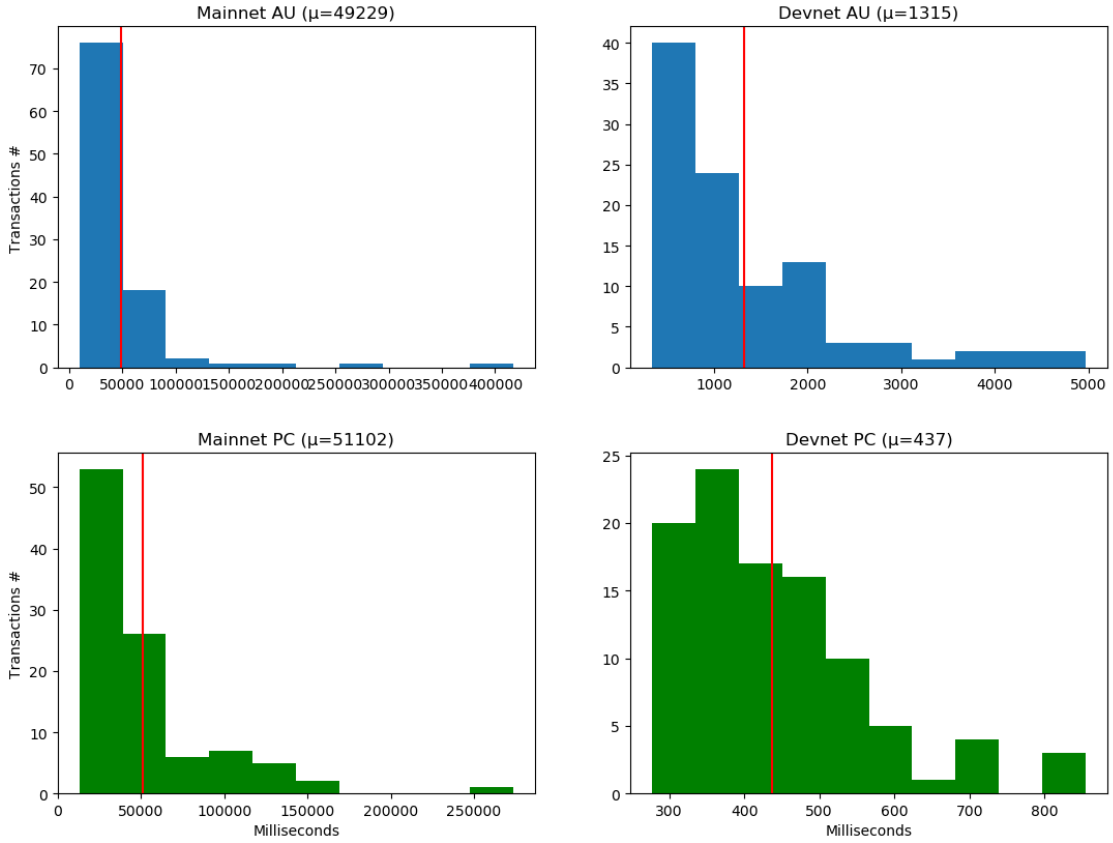


Figure 6.3: Histograms of latency in the process of MAM TX attachment to the tangle for PC and AU in Mainnet and Devnet

Figure 6.3 histograms on the left shows latency when working in the Mainnet, while the ones on the right shows latency in the Devnet. In the case of the Mainnet there is almost no difference in latency between AU and PC, in average it is around 50 seconds. This implies that PoW does not play a big role in this case, but the problem is given by the time spent waiting for tips (Mainnet provider corresponds to Provider1 of previous section, hence it is slower than usual).
Publishing a transaction in a MAM channel in the Devnet takes advantage of the decreased PoW difficulty spending only 437 milliseconds on average using the PC and 1315 milliseconds on average using the AU. The difference here is by a factor of 3 because more weight is given to the computational power. Devnet provider is, indeed, one of the most used, such as Provider2 in Mainnet, but its resources are higher because of less requests typical of a Devnet.

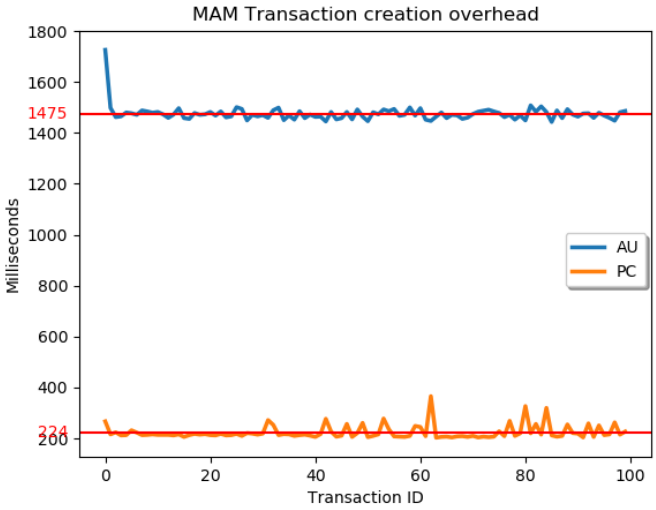


Figure 6.4: Comparison of latency in the creation of a MAM transaction between PC and AU

6.1.4 MAM overhead

Finally, the last factor to consider in this evaluation is the time required to create a MAM TX, that is much greater than the time required for a Simple TX (less than 1 millisecond). This is given by the fact that to create a MAM transaction it is necessary to maintain secure the channel (Merkle Tree based signature scheme section 2.3.3.1). In Figure 6.4 is shown the difference between creating a MAM TX with the AU and with PC. The difference here is due to the computational power, but it is increased by the fact that MAM libraries for the AU are not native. Hence performances are affected by the fact that libraries are not optimized to be used in Android or iOS (only available in JavaScript and Rust). This overhead has a big impact in the process of publishing data in MAM channels, because it doubles the latency using the AU (Figure 6.5). The total time required to create a MAM TX and to attach it to the Devnet Tangle during testing results in 2788 milliseconds on average using the AU and 642 milliseconds using the PC.

6.1.5 Conslusion

From the tests discussed in this section differences between using simple transactions in IOTA and MAM channels have been pointed out, together with, most importantly, the latency in attaching a transaction to the Tangle. Results show that the choice of the right IOTA provider is more important than the computational power of the device making the requests. Another consideration is that optimization, together with a decrease in PoW difficulty (section 7.1.1) , may be the key factors in the use of MAM channels as showed in this work. Simple transactions will always be faster, but MAM channels are needed to provide security and access rights.

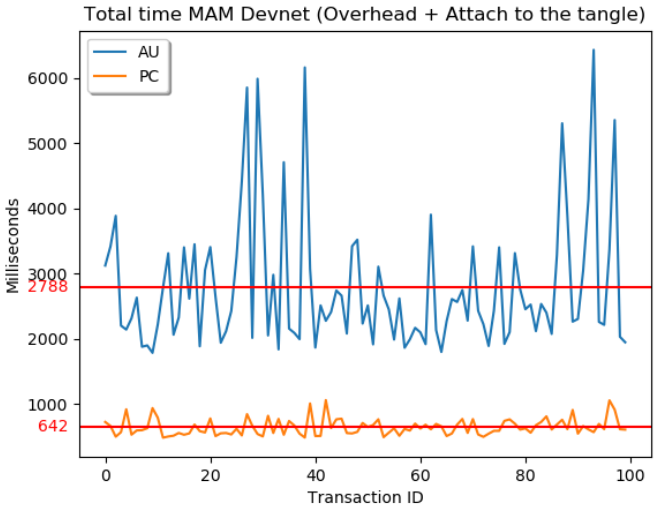


Figure 6.5: Total time required to create a MAM TX and attach it to the Devnet Tangle

6.2 Real world use cases

Real world use cases can be used to assess the ecosystem feasibility. These use cases are intended to show the potential use of some parts of the ecosystem, in order to validate, but also to specify, what has been showed. These are meant to be built by users, that are able to exploit the entire set of features provided by the ecosystem.

6.2.1 Peer-to-peer Ridesharing

Peer-to-peer Ridesharing services, such as Uber, can be still "more" peer-to-peer thanks to the ecosystem. Payments and trip organization is completely handled by Smart Contracts and the intermediary (the role Uber holds) between user A that provides the vehicle and user B that requests the service is limited to only show users like A to users like B.

Intermediary's application The service provided by the intermediary, in this case, is of the type of a Smart Contract based service showed in the previous chapter. This service may be implemented using a client server architecture where the server is subscribed to different pubSub channels regarding location certificates of users and the client is a smartphone app that shows available drivers for a ridesharing in a zone. This implies that drivers that intend to provide the ridesharing service must release the certificate of location (using zk-PoL) in a pubSub channel dedicated to ridesharing (pubSub channels may be subdivided in macro and micro zones).

Ridesharing request The user that requests a ridesharing initiate the process through the intermediary's app, in which he maintains an Ethereum wallet. The app stores current drivers available in the zone and sends a method invocation to a specific driver's Smart Contract. This contract has been uploaded in Ethereum by

the driver during the registration process to the service. The method invoked make a specific request (indicating ridesharing start and end locations) leaving a Movo token deposit purchased through the user's Ethereum account , together with an automatic fee payment to the Intermediary account.

Final transaction The driver is able to receive the request through the Smart Contract event raised by the method. If he refuses to provide the service the transaction will be reverted. If he decides to provide the service, he only needs a signed (through the user's Ethereum address) message, exchanged, for instance, through QR code or NFC, to redeem the deposit.

6.2.2 Data based services

A variety of real world use cases are deployable in the ecosystem regarding data based services. These includes services provided to the user that is sharing the data and services provided to multiple users using multiple user's data.

Vehicle Data For a driver may be useful to provide data access to specific features to some entities that provide maintenance or security. It is easy to include to a feature Smart Contract an address owned by a virtual car mechanic that checks anomalies on the driver's vehicle. Full access to data may be also given to Assurance companies or to Police to assess dynamics of particular traffic events and incidents.

Environmental Data Data gathered from the environment may be acquired by third parties interested in collecting informations from different points in the space to provide different services. A weather app may be based on the aggregation of data coming from different independent sources. This is trivial purchasing MAM channels access through Smart Contracts.

Public transportation services Public transportation services may be enhanced using data gathered by consumers. It is easy to prove that an user has consumed a public transportation service using certificates released by the service itself. Then data validated by these certificates can be purchased by third parties that offer services, e.g. showing the actual condition of a transportation service.

7 Discussion

This chapter is dedicated to some concerns, clarifications and enhancements regarding the ecosystem presented in this thesis. For each environment there may be something to point out, with a particular attention to some non-functional requirements. But technologies are the main focus point, because of all the aspects they offer and that are not used in this work. The following is subdivided in sections where some of the technologies used are analyzed and discussed and sections regarding other issues and final work.

7.1 IOTA

The IOTA Tangle has been central in the development of this thesis work, assuming a key role in data sharing. There are some concerns regarding the actual implementation of the protocol, but also some enhancements that may take place in the future. IOTA is still a ledger "in development", lot of projects are promised by the foundation and some of them are just taking off. Potentially, for the aim of this thesis work, the Tangle and all the mechanisms built around it will be sufficient to replace entirely the use of Ethereum blockchain. In the following sections there are some IOTA use cases not showed in previous chapters.

7.1.1 Scalability

One of the main reason that brought to the use of IOTA in the ecosystem is the Tangle scalability. Ethereum blockchain could be used instead of the Tangle for data sharing, but its use is impractical compared to IOTA because of transactions fees and throughput. In a traditional blockchain (Ethereum, Bitcoin), scalability is not optimal as more and more accounts perform transactions within that specific blockchain. In IOTA, since every transaction verifies two other pending transactions in the Tangle, scalability rises with adoption: as more and more users perform transactions on the network, the time of attaching transactions become much quicker.

Economic Clustering The throughtput issue needs to be solved in order to serve the massive IoT data transactional needs. With Economic Clustering it is possible to take a subset of nodes in a geographic area to validate this operations only in that area. There are subtangles (clusters) in different areas with their own nodes taking care of that zone validations. For instance, with this separation it is possible to avoid loading nodes in China with validation of a transaction in Italy. This allows the user to only need confirmation checks for transactions on local cluster nodes, reducing latency and mitigating systemic transaction volume to prevent bottlenecks. The Economic Clustering is still a work in progress and may be used in the future in the IOTA protocol.

7.1.2 Centrality

IOTA still presents mechanisms of centrality which is a contradiction in terms of decentralization. The Coordinator controlled by the IOTA foundation, introduced in section 2.3.1.4, issues transactions called milestone in order to protect the Tangle against hostile attacks and double spending. Hence the Mainnet cannot be still considered fully decentralized. But the process of removing this Coordinator has been issued by IOTA developers in a process named "Coordicide" [36].

7.1.3 Qubic

Ethereum Virtual Machine remains fundamental for the use of Smart Contracts, especially in the ecosystem described in this work. IOTA does not actually include mechanisms of distributed computing such as Ethereum, but in the future this could change with the introduction of Qubic, making IOTA the only needed DLT in the ecosystem. Qubic is a protocol that specifies IOTA's solution for quorum-based computations, including such constructs as oracle machines, outsourced computations, and smart contracts. Qubic provides general-purpose, permissionless, multiprocessing capabilities on the Tangle. In the long term, Qubic will allow users to leverage world-wide unused computing capacity for computational needs, all while helping to secure the IOTA Tangle: IOTA will become, like the Ethereum Virtual Machine, a world supercomputer.

7.2 IPFS

In this work IPFS has been shown as a solution to store data in a decentralized way, without the need of PoW. Its role is fundamental in the ecosystem and its general purpose may bring to the origin of various decentralized services and application at the core of a new web. Its current limit, as other decentralized technologies, is the user adoption. In particular, to be used in the context of the ecosystem, numerous nodes must be installed by some authorities that are interested in data sharing. The use of IOTA and Ethereum nodes depicted in the implementation chapter is currently possible due to the broad adoption of these technologies (not considering IOTA latency in the Mainnet at the moment). But the implementation specification for IPFS nodes is based on the fact that these nodes must be set to receive files and redistribute blocks generated from these files. Hence a deployment in large quantity is needed by the specification and must be managed.

7.3 Data Access

How data is accessed in the ecosystem may be the biggest concern regarding this work. The advantages of a decentralized structure are reduced by the centrality required in this task. An authorization server is needed to access data in ledgers with current protocols, as showed also in [37]. When data is obfuscated using encryption, keys are needed to access that and a trusted and secure communication channel must be used to release that. Ethereum and IOTA does not allow that.

Solutions Some solutions may be studied, for example in [38] authors modify IPFS software to allow the use of access control lists for files. Another solution, at the expense of performance, may be the use of key escrow mechanisms to distribute keys between different authorization services. In this way someone can access data only knowing all the keys (or a part of them) and a single authorization service cannot directly access data.

7.4 Micropayments

Micropayments are used in the ecosystem to scale transaction operations and to avoid fees payment. These micropayments are useful when a user has to access a service, but a common issue rise up. With current implementation, in order to be able to transact off chain, the sender must deposit a certain amount of currency in favor of a specific Ethereum address. To close the payment channel the sender must present a valid signature coming from that address. But in the case of a service in which multiple devices able to release these signature, it is easy to think that each device has to store its own private key (to distribute the same private key in different devices is not a secure method). In this case a mechanism to accept multiple signature must be used. For example the service may store an internal record of payments of a particular users for all the devices owned.

7.4.1 Flash Channels

A similar approach to Ethereum micropayments are Flash Channels, developed by IOTA foundation as an ad-hoc solution. Flash is a bi-directional off-Tangle payment channel to enable instantaneous, high-throughput transactions. Only two transactions will occur on the Mainnet: opening and closing transactions of the Flash channel. When a channel is created each party deposits an equal amount of IOTA into a multi-signature address controlled by all parties. Once the initial deposits are confirmed the channel does not need to interact with the network until it is closed. Once the parties have finished transacting, the final balances are published to the network.

7.5 Privacy

The solutions adopted in this work always take in consideration user's privacy. User data is protected through cryptography and methods to access it are mostly focused on security (a well defined Smart Contract cannot be hacked due to decentralized computation). Mechanisms that add complexity, such as zk-PoL, are implemented in order to ensure that any data must be accessible only when the user authorize it. The real concern about privacy in this work is regarding the bonding between an user identity and the Ethereum account. Once an Ethereum account is associated to an user by a third party, this one will be able to know the users transactions in the blockchain. This is a general limit of Ethereum that other blockchain proposals are trying to resolve (e.g. Zcash).

7.6 Future work

The idea and the architecture behind this work are heavily biased towards comfort applications, leaving few space (but making them possible) to security applications. This is due to the initial statement regarding the fact that this work has been carried on focusing on current DLT solutions and ready to work.
An enhancement for the infrastructure proposed, consists in the introduction of vehicle-to-vehicle messaging towards safety. Currently pub-sub channels allows the exchange of messaging based on certificates released by trusted authorities. But messaging in regards of traffic accidents or other safety communications can be achieved using Smart Contracts, in order to validate the reputation of messages senders. Through a system based on rewards, incentives and anonymization, users can trust messages received, making messaging become crucial for ITS.
Future work will be focused on the analysis of new paradigms of distributed ledgers in order to provide faster transactions and new consensus algorithms. Blockchain sharding allows a much higher transactions throughput, hence making it possible to enable real-time transactions between moving vehicles. This means that consensus can be reached in traffic situations regarding also decisions to take during driving. It may lead to a situation in which entities in the transportation system are able to communicate and transact in a trusted and secured way, reaching consensus in order to regulate the traffic flow. Content-centric networking allows to access data faster compared to current IP networking standards. Regarding this work, MAM channels and IPFS could be replaced thanks to this type of networking, in which a datum can be directly accessed and routable. The aggregation of blockchain sharding and content-centric networking will be exploited for the ecosystem and it may lead to an increasing in performances and utilities, useful for the creation of a Smart City.

8 Conclusion

In this work an infrastructure for Intelligent Transportation Systems based on the use on Distributed Ledger Technologies has been presented. ITS are rising as a medium to increase the efficiency of transportation services for drivers, passengers and pedestrians, while DLTs pones a new way of trust removing the need of third parties, enhancing transactions between users. Here, a decentralized infrastructure has been presented in terms of an ecosystem where various environments interact and exchange informations in order to obtain two major services for users: Data Sharing and Smart Services.
The vision of the infrastructure has been showed focusing on these two concepts where, Data Sharing allows any user to distribute produced data in a decentralized way, whilst Smart Services are services in the ITS based on a distributed computation. The main idea is supported by some functional and non functional requirements and some scenarios in order to analyze the problem to resolve.
The ecosystem design is composed by four environments that includes different components regarding DLTs and other architectures. An environment is dedicated to data acquisition and storage, consisting in the communication between the AU, user's personal device, and vehicles that allows to obtain data to store in the IOTA Tangle or IPFS. Access to data is provided, in another environment, by an authentication server to anyone that has access rights, obtained through the use Smart Contracts. A third environment embodies a Second Layer Trust needed to validate used data veracity, in which certificates are produced and released. Finally Smart Services represent an entire environment that covers multiple ecosystem components.
The ecosystem implementation is proposed focusing on each environment and on the analysis of the core components and their operations. Sensors are the initial components that act in the infrastructure, as they are placed in vehicles and AUs and register data that users can share. The IOTA Tangle is used to store and validate user's data through the use of Masked Authenticated Messaging Channels. In this process the AU application acts as a wallet that communicates with IOTA full nodes to attach transactions to the Tangle. When data is too large, it is stored in IPFS and referenced in MAM channels. The Ethereum blockchain realizes the infrastructure business logic through Smart Contracts. Some contracts are used by users to manage data access and communication with smart services, while some others are used in the ecosystem to create a unique token and to allow payments off-chain. The authentication server provides access to data to eligible users releasing keys, whereas certificates, released by trusted RSUs or devices, assure the data correctness. Smart Services are implemented through Smart Contracts managed directly by the service provider or they are standard transportation services that exploit the blockchain for trustless transactions.

Finally, a series of tests have been used to validate the infrastructure regarding IOTA performances. A comparison between standard transactions and MAM transactions has been carried out considering different IOTA providers and devices. Results show how finding tips to reference in the attachment to the Tangle process has more influence on latency rather than the Proof of Work operation. This implies that choosing the right provider is fundamental. Considering the promises of IOTA scalability, has been showed that using MAM channels for the ITS is a feasible task.

References

[1] Y. Yuan and F.-Y. Wang, "Towards blockchain-based intelligent transportation systems," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2663–2668, IEEE, 2016.

[2] R. Baldessari, B. Bödekker, M. Deegener, A. Festag, W. Franz, C. Christopher Kellum, T. Kosch, A. Kovacs, M. Lenardi, C. Menig, T. Peichl, M. Röckl, D. Seeberger, M. Straßberger, H. Stratil, H.-J. Vögel, B. Weyl, and W. Zhang, "Car-2-car communication consortium - manifesto," 01 2007.

[3] M. Faezipour, M. Nourani, A. Saeed, and S. Addepalli, "Progress and challenges in intelligent vehicle area networks," *Communications of the ACM*, vol. 55, no. 2, pp. 90–100, 2012.

[4] V. Buterin *et al.*, "Ethereum white paper," *GitHub repository*, pp. 22–23, 2013.

[5] S. Popov, "The tangle," *cit. on*, p. 131, 2016.

[6] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[7] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, "Survey on blockchain for internet of things," *Computer Communications*, 2019.

[8] K. Biswas and V. Muthukkumarasamy, "Securing smart cities using blockchain technology," in *2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd international conference on data science and systems (HPCC/SmartCity/DSS)*, pp. 1392–1393, IEEE, 2016.

[9] S. Ibba, A. Pinna, M. Seu, and F. E. Pani, "Citysense: blockchain-oriented smart cities," in *Proceedings of the XP2017 Scientific Workshops*, p. 12, ACM, 2017.

[10] DOVU, "Whitepaper." https://dovu.io/whitepaper.pdf, 2018.

[11] B. Leiding and W. V. Vorobev, "Enabling the vehicle economy using a blockchain-based value transaction layer protocol for vehicular ad-hoc networks," *URL: https://uploads-ssl. webflow. com/5a4ea18a81f55a00010bdf45/5b69e53263e2a6076124ecbe_ Chorus-Mobility-WP–v1. 0.1. pdf*, 2018.

[12] MOBI. https://dlt.mobi/, 2019.

[13] IOTA. https://bitcointalk.org/index.php?topic=1216479.0, 2015.

[14] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[15] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *Journal of network and computer applications*, vol. 37, pp. 380–392, 2014.

[16] C. E. Palazzi, M. Roccetti, and S. Ferretti, "An intervehicular communication architecture for safety and entertainment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 90–99, 2009.

[17] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger,"

[18] S. Popov, O. Saa, and P. Finardi, "Equilibria in the tangle," *arXiv preprint arXiv:1712.05385*, 2017.

[19] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pp. 326–349, ACM, 2012.

[20] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," *Tech. rep. 2016–1.10. Zerocoin Electric Coin Company, Tech. Rep.*, 2016.

[21] L. Wolberger and V. Fedyukovych, "Zero knowledge proof of location," 2018.

[22] B. Leiding, P. Memarmoshrefi, and D. Hogrefe, "Self-managed and blockchain-based vehicular ad-hoc networks," pp. 137–140, 09 2016.

[23] P. K. Sharma, S. Y. Moon, and J. H. Park, "Block-vn: A distributed blockchain based vehicular network architecture in smart city.," *JIPS*, vol. 13, no. 1, pp. 184–195, 2017.

[24] M. Singh and S. Kim, "Intelligent vehicle-trust point: Reward based intelligent vehicle communication using blockchain," *arXiv preprint arXiv:1707.07442*, 2017.

[25] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[26] R. Shrestha, R. Bajracharya, and S. Y. Nam, "Blockchain-based message dissemination in vanet," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pp. 161–166, IEEE, 2018.

[27] V. Ortega, F. Bouchmal, and J. F. Monserrat, "Trusted 5g vehicular networks: Blockchains and content-centric networking," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 121–127, 2018.

[28] H. Khelifi, S. Luo, B. Nour, H. Moungla, and S. H. Ahmed, "Reputation-based blockchain for secure ndn caching in vehicular networks," in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–6, IEEE, 2018.

[29] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on Cloud Computing Security Workshop*, pp. 45–50, ACM, 2017.

[30] K. R. Özyilmaz, M. Doğan, and A. Yurdakul, "Idmob: Iot data marketplace on blockchain," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 11–19, IEEE, 2018.

[31] H. Mousannif, I. Khalil, and H. Al Moatassime, "Cooperation as a service in vanets.," *J. UCS*, vol. 17, no. 8, pp. 1202–1218, 2011.

[32] P. C. Bartolomeu, E. Vieira, and J. Ferreira, "Iota feasibility and perspectives for enabling vehicular applications," in *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–7, IEEE, 2018.

[33] BIP-32. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, 2012.

[34] High-Mobility. https://high-mobility.com/, 2019.

[35] D. McDuff, A. Mahmoud, M. Mavadati, M. Amr, J. Turcot, and R. e. Kaliouby, "Affdex sdk: a cross-platform real-time multi-face expression recognition toolkit," in *Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems*, pp. 3723–3726, ACM, 2016.

[36] I. Foundation, "Coordicide." https://files.iota.org/papers/Coordicide_WP.pdf, 2019.

[37] V. A. Siris, D. Dimopoulos, N. Fotiou, S. Voulgaris, and G. C. Polyzos, "Interledger smart contracts for decentralized authorization to constrained things," *arXiv preprint arXiv:1905.01671*, 2019.

[38] M. Steichen, B. Fiz Pontiveros, R. Norvill, W. Shbair, *et al.*, "Blockchain-based, decentralized access control for ipfs," in *The 2018 IEEE International Conference on Blockchain (Blockchain-2018)*, pp. 1499–1506, IEEE, 2018.