José Carlos Severino Cardoso

Blockchain and Smart Contracts for the Internet of Things - an Architecture for Sensor Data Availability

Orientadores: Professor Doutor José Luís de Azevedo Quintino Rogado (ULHT)
Professor José Carlos dos Santos Guerreiro Faísca (ULHT)

Universidade Lusófona de Humanidades e Tecnologias
Escola de Comunicação, Artes e Tecnologias da Informação (ECATI)
Departamento de Engenharia Informática e Sistemas de Informação

LISBOA
2018

José Carlos Severino Cardoso

Blockchain and Smart Contracts for the Internet of Things - an Architecture for Sensor Data Availability

Dissertação apresentada para a obtenção do Grau de Mestre em Engenharia Informática e Sistemas de Informação conferido pela Universidade Lusófona de Humanidades e Tecnologias no dia 6 de junho de 2018, perante o júri, nomeado pelo Despacho Nº23/2018 de 22 de janeiro de 2018, com a seguinte composição:

- **Presidente**: Prof. Doutor Rui Pedro Nobre Ribeiro (ULHT)
- **Arguente**: Prof. Doutor Pedro Alexandre Reis Sá da Costa (ULHT)
- **Orientador**: Professor Doutor José Luís de Azevedo Quintino Rogado (ULHT)

Universidade Lusófona de Humanidades e Tecnologias
Escola de Comunicação, Artes e Tecnologias da Informação (ECATI)
Departamento de Engenharia Informática e Sistemas de Informação

LISBOA
2018

Epigraph

Everything is theoretically impossible until it is done.

Robert A. Heinlein

Acknowledgments

I would like to express heartfelt gratitude to Professor Doutor José Rogado, my advisor, for all his guidance, motivation and invaluable help throughout the master's term and especially in this thesis.

I would like to address my sincere thanks and gratitude to my co-advisor, Professor José Faísca for having challenged me to accept this topic and for having accompanied me so tightly and committedly during the development of the thesis, with his vast technical expertise in this area, and also encouraging me to continue in the most difficult moments.

I would like to thank Professor Doutor Rui Ribeiro for having believed me at the right time.

To my master's colleagues, especially Telmo Paixão and Ricardo Henriques, thank you very much for your guidance and support in the most difficult moments.

A huge thanks to my master's colleague and friend Marco Figueiredo, for all the mutual help and friendship of the last years.

Finally, an extraordinary thanks to my family, my parents and sister, because without your love I would not have gotten here. To my wife, a huge thank you for your support, strength and unconditional love, but also for the help and patience in the worst hours. Without you, everything had been harder!

Resumo

O Blockchain é uma tecnologia emergente recentemente generalizada para muitas áreas de atividade. O seu modo de operação descentralizado enquadra-se perfeitamente em vários cenários onde o principal desafio reside na melhoria da comunicação máquina-máquina e na possibilidade de realizar transferências seguras com serviços de valor acrescentado. As redes IoT são uma das áreas possíveis de aplicação da tecnologia Blockchain, uma vez que, para poder satisfazer os requisitos da indústria, a arquitetura atual das redes IoT tem limitações, que podem ser superadas através da melhoria das comunicações entre dispositivos e do acesso a formas evoluídas de agregação e consumo dos dados recolhidos.

Este trabalho procura avaliar a combinação desses dois paradigmas - IoT e Blockchain -, procurando entender como o IoT pode beneficiar das funcionalidades que o Blockchain oferece: um sistema de armazenamento mais económico, descentralização e verdadeira redundância, confiança sem autoridade central - privacidade - e segurança reforçada. Por outro lado, o Blockchain como infraestrutura financeira para o IoT é também um aspeto fundamental desse trabalho. Na prova de conceito construída, este cenário é implementado, pois os dados de um sensor podem ser transacionados com uma entidade que os solicite.

São igualmente realizadas análises estatísticas e de desempenho relativamente à arquitetura implementada, sendo também apontados alguns pontos de melhoria para alavancar o uso do sistema em situações reais.

Palavras-chave

Blockchain, Contrato Inteligente, Aplicações descentralizadas, Internet das Coisas, Dados de Sensor

Abstract

The Blockchain is a newly emerging technology recently generalized to many areas of activity. Its decentralized mode of operation fits perfectly into several scenarios where the primary challenge lies in improving machine-to-machine communication and the ability to achieve secure transfers with value-added services. IoT networks are one of the possible areas of application of Blockchain technology since the current of IoT networks architecture has limitations that do not comply with industry requirements, which can be overcome by improving communications between devices and by accessing more advanced forms of aggregation and collected data consumption.

This work assesses the combination of these two paradigms - IoT and Blockchain -, to evaluate how the IoT can benefit from the functionalities offered by the Blockchain: a more economical storage system, full decentralization and true redundancy, trust without a central authority and improved security. On the other hand, Blockchain as a financial infrastructure for IoT is also a fundamental aspect of this work. In the built-in proof of concept, this scenario is implemented, since data captured by a sensor can be transacted to a requesting entity.

Statistical and performance analysis are performed on the implemented architecture, and some areas of improvement are also pointed out that could to leverage the system's adoption in real life situations.

Keywords

Blockchain, Smart Contract, Decentralized applications, Internet of Things, Sensor Data

Abbreviations

API – Application Programming Interface

BOINC – Berkeley Open Infrastructure Network Computing Grid

BTC – Bitcoin

CPU – Central Processing Unit

DAC – Decentralized Autonomous Corporation

Dapp – Decentralized application

DHT – Distributed Hash Tables

EBCM – E-commerce Blockchain consensus mechanism

EOA – Externally Owned Accounts

ETH – Ether

EVM –Ethereum Virtual Machine

GUI – Graphical User Interface

HTTP – Hypertext Transfer Protocol

IBM – International Business Machine

IoT – Internet of Things

IP – Internet Protocol

IPC – Inter-Process Communication

IPFS – Interplanetary File System

IT – Information Technology

JSON – JavaScript Object Notation

JWE – Json Web Encryption

JWS – Json Web Signature

JWT – Json Web Tokens

MIT – Massachusetts Institute of Technology

MSC – Membership Service Provider

NFC – Near Field Communications

OBC – Open Blockchain

OSI – Open System Interconnection

PBFT – Practical Byzantine Fault Tolerance

PoW algorithm – Proof-of-work algorithm

P2P – Peer to Peer

QoS - Quality of service

REST – Representational State Transfer

RFID – Radiofrequency

RSK – Rootstock

RPC – Remote Procedure Call

RTC – Rootcoin

SDP – Smart Data Pricing

SGX – Software Guard Extensions

SQL – Structured Query Language

SSL – Secure Socket Layer

TC - Town Crier

TLS – Transport Layer Security

TLSNotary – Type of Oracle that can cryptographically attest the origin of the data

Tps – Transaction per Second

Txpool – Transaction Pool

UMTS –Universal Mobile Telecommunication System

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

WiMAX – Worldwide Interoperability for Microwave Access

WSS – Websocket Secure

General Index

Table Index

Illustration Index

1 Introduction

Blockchain technology gives the possibility of creating decentralized currencies, self-executing digital contracts - Smart Contracts - and assets that can be managed over the Internet - smart property -. This technology is also capable of promoting the development of new governance systems with more democratic decentralized – autonomous - decision-making characteristics, and of organizations that can operate over a network of computers without any human intervention (Wright & De Filippi, 2015).

In the recent years, two different technologies have been the object of increasing interest from either scientific and business communities: on the one hand, the so-called Internet of Things – IoT - has brought the vision of a fully connected network, where even the tiniest device is capable of playing a specific role on gathering and broadcasting personal and environmental data. On the other hand, the Blockchain technology creates peer-to-peer networks where non-trusting members, without a trusted intermediary, can interact verifiably and securely. This work starts by performing surveys of these two paradigms and then tries to demonstrate whether specific Blockchain characteristics can be used to add value to the IoT fundamental proposition.

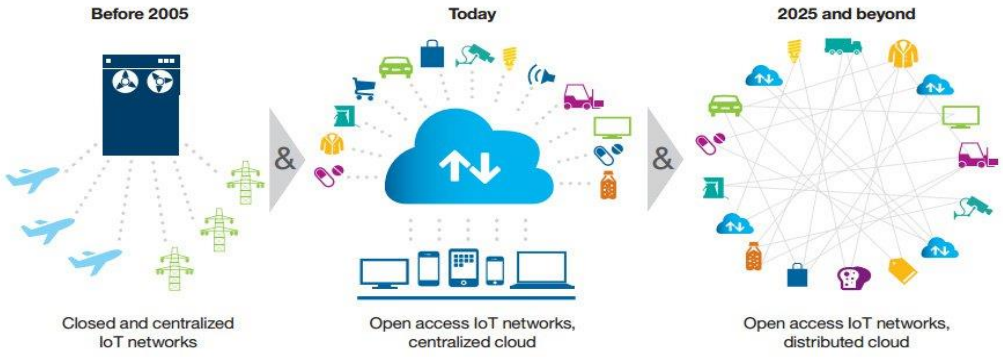

Figure 1 – Evolution of IoT networks (Brody & Pureswaran, 2014)

One of the most recent and interesting aspects of Blockchain is the possibility to allow the establishment of Smart Contracts - scripts that enable the automation of multi-step processes - which can be activated whenever a specific transaction is performed. A thorough analysis of these mechanisms is performed, to discover how a Blockchain-IoT combination may enable the sharing of services and resources, possibly leading to the creation of a marketplace between devices, by allowing the automation in a cryptographically verifiable manner, several existing time-consuming workflows (Christidis & Devetsikiotis, 2016).

By combining the IoT with Blockchain technology, it is possible to mitigate a current limitation of IoT networks: today, the massive increase of transactions from IoT devices is causing several problems like network latency, which is making the maintenance of centralized datasets an enormous cost for organizations. Adopting peer-to-peer computing - as provided by Blockchain - to process the hundreds of billions of IoT transactions, can significantly reduce costs associated with installing and maintaining large centralized data centers, since the devices can talk directly to each other, without needing a central authority (Brody & Pureswaran, 2014). Another reason to combine these two paradigms lies on the fact that current E-business models could hardly fit the IoT requirements as a convenient, safe and stable transaction system is needed to exchange information and payments.

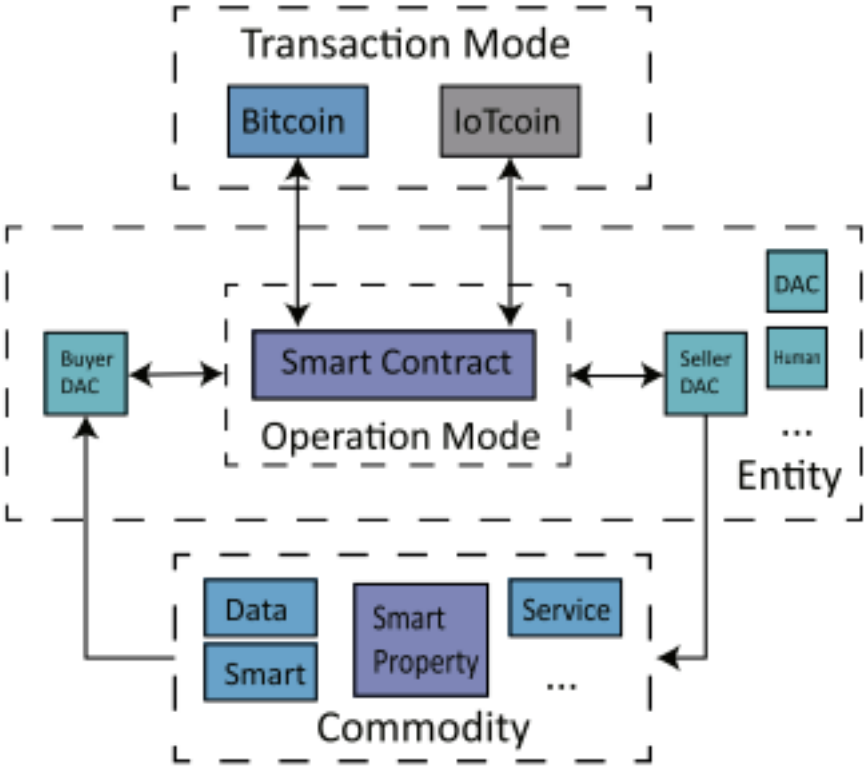

Figure 2 – IoT business model using Blockchain. DAC – Decentralized Autonomous Corporation (Y. Zhang & Wen, 2017)

For instance, consider the example of developers that need to access the sensor data of a specific device. First, they must obtain an account from a company via traditional E-commerce infrastructure. Next, the sensor data company sends the developers some data or API access authorization, and after that, they collect the payment. Due to the necessity of validating the existence of the user and the validity of the payment methods - e.g. Credit Card -, much

additional overhead is included by a trusted compensation third party. As a result, the transaction costs increase while its efficiency decreases. The emergence of the Blockchain based payment systems like the Bitcoin makes it possible to achieve P2P monetary transactions without the intervention of the third party, therefore reducing their costs and increasing their efficiency (Y. Zhang & Wen, 2017).

The final goal of this work is to demonstrate the benefits of this hypothesis, by developing a proof of concept representing a real use case of disclosing IoT sensor data – upon payment - through Blockchain technology, leveraging its decentralized features, which will be further described throughout in this document.

1.1 **Motivation**

The Blockchain is an emerging technology that was recently generalized to many areas. Though it started as the original the core technology behind the Bitcoin cryptocurrency, its use cases are expanding to many other areas, including the IoT (Huh, Cho, & Kim, 2017).

The motivation for this work is to understand how a Blockchain-IoT combination could be a win-win association, by facilitating the sharing of services and data, enabling micropayments and automating in a cryptographically verifiable manner several existing and time-consuming workflows (Christidis & Devetsikiotis, 2016). This is achieved using Blockchain and Smart Contracts to build a decentralized application that can sell IoT sensor data to any entity that wishes to buy it.

1.2 **Research questions**

Is it possible to apply the Blockchain technology and Smart Contracts to read sensor data, building open access Internet of Things – IoT - networks?

1.3 **Research Goals**

- Develop an architectural model of Blockchain technology application for transacting IoT sensor data;

- Implement a real scenario to achieve the mentioned model, validating the potential of the Blockchain as an infrastructure for supporting IoT operation.

1.4 **Main contributions**

As mentioned above, the primary goal of this study is to evaluate how the unique Blockchain characteristics can leverage the IoT operation, resulting in a broad understanding

of how these two paradigms can fit together. This result leads to the most significant contributions this work brings to the scientific community: firstly, the conceptual architecture proposed, and secondly, the proof of concept implemented, both conceived using a highly modular approach. This allows an easy possible replacement of components by some other modules available within the community/market, thus allowing the existence of several alternatives of the solution, on which the same underlying philosophy and mechanisms are maintained.

1.5 **Structure of the report**

This work starts with an 'Introduction' chapter, giving a general overview of the topics under analysis – Blockchain and IoT -, and then develops the research question.

The 'Background, Concepts and Technologies' chapter, describes the concepts of Blockchain and one of its current implementations – the Ethereum. Then, the Internet of Things concept is explained. At the end of this chapter, the technologies used for implementing the proof of concept are detailed.

The 'Related Work' chapter explains how the Blockchain can facilitate the IoT to overcome its most prominent challenges, analyzing several use cases where these two paradigms are combined. Some architectural alternatives for each system component are also described.

The 'Development of a Solution Model' chapter describes the conceptualized solution from a high-level perspective, also providing information about its abstract components and the interactions between them.

The 'Implementation of the Solution Concept' chapter details the implementation of the high-level design introduced in the previous chapter, explaining how the technologies were used, and the interaction flow within the system, along with some user interface description schemas.

The 'Results' chapter presents the experiments performed to measure system's performance, as well as proposes a statistical model to explain the possible connection between confirmation time and other factors thought to be highly related.

'Conclusions and Perspectives' is the last chapter of this work. It contains the conclusion obtained from the results interpretation. At the end of this chapter, some directions for future research are also presented, where some possible improvements for this work are pointed out.

2 Background, Concepts and Technologies

This chapter prepares the stage for the more technical parts that are covered in subsequent steps.

2.1 Concepts

2.1.1 Blockchain

A Blockchain is a type of a distributed ledger in which all transactions are verified, recorded and immutable, being shared by the participants of the network (Watanabe et al., 2015). Everybody can join the network, and download a copy of the Blockchain from other nodes (De Jong, 2015). The structure of a Blockchain is a block that consists of multiple transactions and is related to a previous block in chain-like form. To ensure reliability, when a new block is added to the previous block, a consensus algorithm is executed. (Watanabe et al., 2015). This distributed consensus mechanism is responsible for managing the addition of new items, consisting of a set of rules for validating and broadcasting transactions and blocks, resolving conflicts and also controlling the incentive scheme for blocks creation. The consensus guarantees all stored transactions are valid, and that each valid transaction is added only once (X. Xu et al., 2017).

From an architectural point of view, Blockchain enables new forms of distributed software designs, where agreement on a shared state for decentralized and transactional data can be established across a vast network of untrusted participants. This fact bypasses the need to rely on a central, trusted integration entity, which has the power to control and handle the system, being a single point of failure (X. Xu et al., 2017). In this way, it is common to say that Blockchain technology has the potential to reduce the role of one of the most critical economic and regulatory actors in our society - the middleman -, which is perhaps its most disruptive feature (Wright & De Filippi, 2015).

Immutability, non-repudiation, integrity, transparency, and equal rights are the principal properties supported in existing Blockchains, and applications built on top of it can benefit from them. Data contained in a Blockchain committed transaction will eventually become in practice immutable, producing nonrepudiation of the stored data and also providing data integrity. The public access enables data transparency, and equal rights concede every participant the same ability to access and manipulate the Blockchain. These rights can be weighted by the compute power or stake owned by the miner (X. Xu et al., 2017).

Regarding history, many time before the advent of the Blockchain digital cash had been conceptualized with a central server trusted to prevent double-spending. However, failure to ensure compatibility between centralization, anonymity and double-spending prevention put the viability of this new form of money into question, despite all significant cryptographic enhancements. Three decades later, Bitcoin has acquired notoriety in the global marketplace by replacing the central server's signature with a consensus mechanism based on proof of work algorithm. The novelty and improvement are the decentralized nature of the payment system allowed by Blockchain technology, thereby ushering in a new era that extends beyond global payments, to corporate governance, social institutions, democratic participation and the functioning of capital markets. (Pilkington, 2015)

2.1.1.1 **The bitcoin example**

The most known example of the Blockchain technology use is Bitcoin, which is a p2p version of electronic cash, enabling payments to be sent directly from one party to another without being needed action from a central authority - a bank -. Digital signatures are part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. To solve this, bitcoin uses proof-of-work algorithm to record a public history of transactions that becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power (Nakamoto, 2008).

2.1.1.2 **Generalization of Blockchain concept**

Nowadays there are millions of software applications currently in use, and the vast majority follows a centralized server-client model. Some are distributed, and a select few novel ones are decentralized. Figure 3 shows a visual representation of these three models for software.

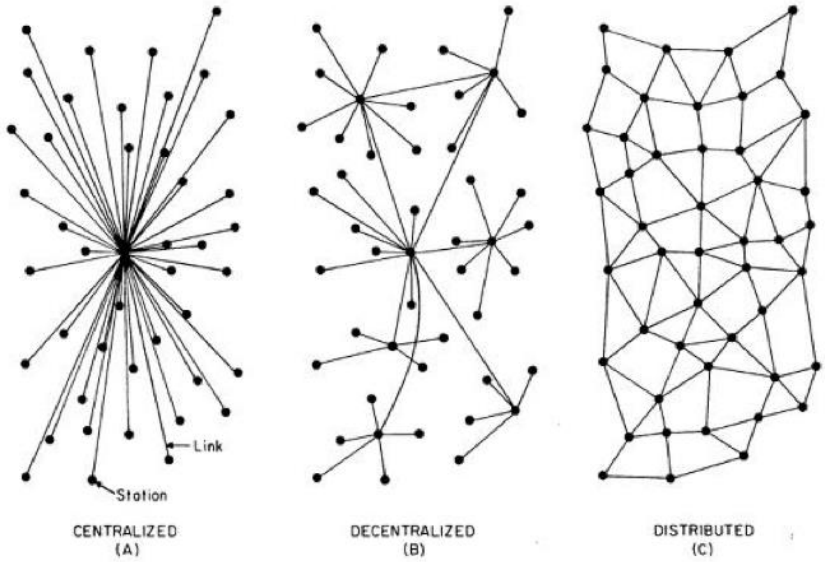

Figure 3 – Comparison between centralized, decentralized and distributed models (Raval, 2016).

Centralized systems are currently the most common model for software applications. They control the operation of the individual units, and flow of information from a single point and everyone in this kind of system is directly dependent on the central power to send and receive information. Facebook, Amazon, Google and every other mainstream services used on the Internet uses this model. Distributed means computation is spread over multiple nodes instead of just one. In another hand, decentralized allows that if one node fails, the network is still able to operate, which means that any application that uses a Blockchain alongside other peer-to-peer tools can be distributed and decentralized (Raval, 2016).

The dapp – decentralized application - concept appears with the natural extension of Blockchain scope, being currently an emerging field with a considerable development, where new models appear quite often. The following principles are generally accepted as the core features for a dapp:

A. **Open Source**

Closed-source applications require users to trust that the app is as decentralized as the authors say it is, thus raising a red flag to users since they cannot verify the source code. This is particularly pronounced when the application is designed to receive, hold, or transfer funds. Open sourcing a dapp makes the Internet to be the standard denominator instead of a chain of closed silos (Johnston et al., 2014).

B. **Internal Currency**

How to monetize a dapp is an essential question that consistently comes up in dapp circles. In the context of a daap, the traditional monetization strategies for centralized applications like transaction fees, advertising revenues, referral commissions, access rights to user data and subscription services are not possible, as some user might fork the app and take out the commissions that came from these strategies, which is not ideal. A possible approach is to allocate resources in the network using a token: an 'appcoin'. Users need this appcoin to use the network and owners of the resources get paid in appcoins. Because the network is designed to grow and include more users and there was a fixed amount of coins, the values of the coins grow as well. This model can be applied to any dapp, and their resources could be trades, images, texts, storage, videos, ads, and so on. However, this does not mean that users would need to pay to use a dapp, but pay for trading its underlying assets (Raval, 2016).

C. **Decentralized Consensus**

Before Bitcoin, consensus on transaction validity always required some degree of centralization. Any payment/transaction had to go through a central authority that monitored all transactions. Distributed Hash Tables – DHTs - like BitTorrent were invented before the Blockchain. They are useful for storing and streaming decentralized data, however but if an agreement is needed in a decentralized way, a decentralized solution such as Blockchain is also mandatory. The Blockchain's unique strength is that does not force nodes to trust each other on the validity of data, but instead uses transactions with robust security mechanisms and consensus algorithms such as the proof of work.

The Blockchain's main innovation is the decentralized consensus. In the past, a lot of decentralized protocols were created, but they all required nodes to trust one another. The Blockchain is an immutable record that every node has a copy of, so there is no need of trust between nodes. (Raval, 2016).

D. **No Central Point of Failure**

Due to its decentralized nature, dapps cannot be shut down, since there is no server to turn off. Data in a dapp is decentralized across all of its nodes. Each node is independent; if one fails, the others are still able to run on the network. Some decentralized database systems build

dapps that allow for this feature, such as Interplanetary File System, BitTorrent, and independent DHTs (Raval, 2016).

Blockchain can be used for multiple purposes, rather than financial transactions. Despite Bitcoin's popularity, the Blockchain remains the central innovation due to its core features. That's why there have been a number of subsequent developments from this elements , usually referred as the second wave of Blockchain innovation - or Bitcoin 2.0 technologies - ,that are trying to generalize Blockchain functionality by linking it with Smart Contracts, enabling the appearance of the dapps, where all these concepts are applied (Garrod, 2016). Following, we present some daaps that take advantage of the mentioned features:

OpenBazaar

OpenBazaar proposes a decentralized eBay version. No broker can tell sellers what they can and can't sell or settle on the fees for using the service. However, the problem is that the sellers must host their private stores. Ideally, they could upload their store data to the network spending a small fee, which requires a decentralized design of incentivized storage miners. Open Bazaar uses Bitcoin as currency for transactions between buyers and sellers, and BitTorrent's protocol for data transfer (Raval, 2016).

FireChat

FireChat began with a famous use case - the Hong Kong protests for democracy in 2014 -. China's infamous 'Great Firewall' is known for blocking IP addresses for content that it considers pro-democracy or not in its interest. The protesters feared that the authorities could try to shut down access to various social networks to stop collaboration, which is possible to do with the HTTP protocol. Alternatively, they adopted FireChat, an app that used a new feature called multi-peer connectivity, allowing phones to connect to each other in a straight way. Thus, the administration would be forced to manually power off each node, which is not possible, leading protestors to be able to communicate with one another securely (Raval, 2016).

Gridcoin

Gridcoin is an open-source project based on the Bitcoin protocol that provides a middleware system for volunteer and grid computing - Berkeley Open Infrastructure Network Computing Grid or BOINC -. Gridcoin is an unexpected Blockchain application that implements a p2p internet-based cryptocurrency which purposes are to provide real benefits to

humanity by compensating miners for participating in BOINC projects. By redirecting the computational power towards BOINC research, participants contribute to advances in medicine, biology, climatology, and astrophysics.

Blockchain-based Digital Identity Providers

Digital identity can benefit from Blockchain core features. An innovative company in this field is OneName, a 'passcard identity company building access control on the Blockchain.' With OneName, digital identities cannot be controlled by a central institution or company, as it grants a trustless and decentralized service (Pilkington, 2015).

Blockchain-based Voting Systems

Voting systems are using the cryptographic capabilities of Blockchain technology to, upon the inherent immutability, transparency, and consensus record every vote in a secure way. A Danish political party first implemented this for internal elections purposes**.** (Millet, 2014).

2.1.2 **Ethereum**

The Blockchain paradigm has demonstrated its utility through many projects when linked with cryptographically secured transactions. This model is called a transactional singleton machine with shared-state. Ethereum implements this standard in a generalized way. (Wood, 2014). The purpose of Ethereum is to empower developers to create applications with scalability, standardization, feature-completeness, and ease of development, by improving the concepts of scripting, altcoins, and on-chain meta-protocols. Ethereum does this by building a Blockchain with a built-in Turing-complete programming language, enabling anyone to write Smart Contracts and decentralized applications with their particular arbitrary rules for transaction formats, ownership, and state transition functions (Buterin, 2014).

Ethereum, in a global perspective, can be seen as a transaction-based state machine: it begins with a genesis state and incrementally executes transactions to morph it into some terminal state. This state can incorporate such data as account balances, reputations, trust arrangements, data concerning to information of the physical world, that is, anything that can be described by a computer is allowed. The execution model defines how the system state is modified given a series of bytecode instructions and a small tuple of environmental data. This is specified within a formal model of a virtual state machine, known as the Ethereum Virtual Machine - EVM. Transactions are arranged into blocks; blocks are chained together applying a

cryptographic hash as a means of reference. Blocks perform as a journal, recording a series of transactions together with the preceding block and an identifier for the final state. Blocks also punctuate the transaction series with incentives for nodes to mine, acting like a state-transition function, adding value to a nominated account. To incentivize computation within the network, it is necessary to have an agreed method for transmitting value. This led to the creation of an intrinsic currency, named Ether, also known as ETH. The smallest subdivision of Ether, and thus the one in which all integer values of the currency are counted, is the Wei. One Ether is defined as being 10^{18} Wei (Wood, 2014).

2.1.2.1 **Key characteristics**

Ethereum platform can be briefly described by the following features/capabilities, which allow leveraging conceptual Blockchain potential:

Accounts

Ethereum accounts are the simplest way to store Ether - Ethereum currency -. They are public/private key pairs which are used use to sign transactions. With any Ethereum client, it is possible to generate one and send some ether to it. Ethereum has two types of accounts - Externally Owned Accounts, EOAs - and Contract Accounts. EOAs are the accounts held and managed by the users, having an Ether balance connected with it. These accounts can send transactions to other EOAs or contract accounts. The contract accounts are linked with a contract and its code, having also an ether balance. Messages and transactions from other contracts trigger contract code execution, which can perform operations like manipulating its own persistent storage. (Bahga & Madisetti, 2016).

Transactions

A transaction is a signed data package that contains a message to be sent from an EOA, being formed by the amount of ether and the data to send, the recipient of the message and a signature identifying the sender. Also, for limiting the exponential blow-up and infinite loops in code, all programmable computation in Ethereum is subject to fees. To achieve this, every transaction has three values called ‘gas limit’ - a limit to how many computational steps of code execution it can produce, including both the initial message and any additional messages that get produced during execution -, ‘gas price’ - the fee per computational step to pay to the miner -, and ‘transaction fee’ – the total gas spent on a transaction, calculated as the multiplication between gas used and gas price. In the case of a transaction ‘runs out of gas’, every state changes

revert - except for the payment of the fees, and if transaction execution halts with some remaining gas, then the unused portion of the fees is refunded to the sender. To create a Smart Contract, there is a different transaction type - and corresponding message type -, which address is computed based on the hash of the account nonce and transaction data (Buterin, 2014).

Entities performing transactions are open to establishing any gas price they want. However, miners are also free to disregard transactions as they choose. A higher gas price on a transaction will consequently cost the sender more regarding Ether and present more significant value to the miner, being more likely to be selected for inclusion by more miners. Miners, in overall, will choose to display the smallest gas price for which they will perform transactions and transactors will be free to consider these prices to determine what gas price should be proposed. Due to a weighted allocation of minimum satisfactory gas prices, transactors will inevitably have a trade-off between dropping the gas price and maximizing the possibility that their transaction will be mined on time. It is also important to describe the confirmation time concept – the time that Ethereum takes to include the transaction in at least one block (Wood, 2014).

Mining

Mining is the method of applying effort - working - to bolster one series of transactions - a block - over any other possible competitor block (Wood, 2014). In Ethereum, miners refer to a vast global network of computers, operated mostly by enthusiasts in their homes and offices, running Ethereum nodes that are paid in ether tokens for the work of executing Smart Contracts and validating the canonical order of transactions around the world. Each node may undertake the process of mining, but the term also refers to the collective effort of the network: individual nodes mine and the network itself can be said to be secured by mining. Thus, mining can adequately be defined as dedicating computational effort to the bolstering of a given version of history as the correct one. The mining process is computationally demanding for nodes because it involves executing a memory-intensive hashing algorithm known as a proof-of-work algorithm. The proof-of-work algorithm - PoW - for the Ethereum protocol is the 'Ethash' function, also referred as Ethereum's consensus algorithm or consensus mechanism, created by the core developers to address the problem of mining centralization evident in Bitcoin, as top miners are increasing its hardware capabilities to retain the mining power and therefore contradicting Blockchain decentralized nature (Dannen, 2017).

Smart Contracts

Created by Nick Szabo, Smart Contracts are essentially digital contracts that are enforced automatically by a set of computer protocols (Garrod, 2016). Bitcoin is considered as a particular type of Smart Contract, in which a payment is agreed between two parties; but Smart Contracts can govern any agreement that can be digitized (Aron, 2015).

A Smart Contract is a unit of business logic uploaded to the EVM. In the context of a typical dapp architecture, Smart Contracts, along with the EVM, form the backend layer that will be connected to a frontend accessible to the end user.(Dannen, 2017).

Events

Ethereum Events are inheritable members of Smart Contracts, which can be used to invoke JavaScript callbacks in the user interface of a dapp that is listening for these events. After being called, the arguments are stored in the transaction's log, which relates with the address of the contract and will be incorporated into the Blockchain being available as long as the respective block is accessible (Ethereum, 2017).

2.1.3 Internet of Things - IoT

The IoT technology can be explained as a connection between humans, computers, and things. All the equipment's we use in our day to day life can be controlled and monitored using the IoT (Suresh, Daniel, Parthasarathy, & Aswathy, 2014). The term Internet of Things - IoT - has been quite known for last few years, and due to the advancement of wireless technology, is getting more attention in recent time. The essential idea is that the variety of objects such as RFID, NFC, Sensors, actuators and mobile phones can relate to each other by having a distinct address and may share and synchronize information. When IoT started, Radiofrequency - RFID - seemed to be necessary for it. However, today there are multiple technologies similar to RFID like Near Field communications - NFC -, which can also be used to implement the modern concept of IoT. (Shah & Yaqoob, 2016).

2.1.3.1 IoT architecture

The typical IoT architecture can be represented into five layers as shown in Figure 4. Each layer is briefly described below:

- Perception Layer: this layer is related to the physical layer in OSI model,

consisting of the different sensor types - i.e., RFID, Zigbee, QR code, Infrared. -, devices and environmental elements. This layer copes with the overall device management, identification, and collection of specific information by each type of sensor devices. The gathered information can be location, wind speed, vibration, pH level, humidity and amount of dust in the air and will be transmitted through the Network layer for its secure communication toward central information processing system.

- Network Layer: this layer plays a vital role in secure transfers, keeping the sensitive information confidential from sensor devices to central servers through 3G, 4G, UMTS, WiFi, WiMAX, RFID, Infrared and Satellite. Hence, this layer is mainly responsible for transfer the information from Perception layer to upper layer - Middleware Layer -. The devices in the IoT system may generate various type of services when they are connected and communicated with others.
- Middleware layer: it has two essential roles, including service management and store the lower layer information into the database. Moreover, this layer can retrieve, process and compute information, and then automatically decide based on the results.
- Application Layer: this layer is responsible for applications management based on the processed information in the Middleware layer. These applications can be a smart car, smart postal, smart health and smart home.
- Business Layer: this layer responsibilities cover the whole IoT applications and services management, as it consists of presenting the accurate data received from the lower layers to the end user, by creating practically graphs, flowchart and executive report. This data will allow managers or executives to make more accurate decisions about the business strategies and roadmaps (Tuwanut & Kraijak, 2015).

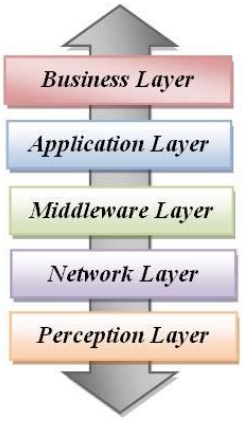

Figure 4 – The IoT generic architecture (Tuwanut & Kraijak, 2015).

Since its birth in 1989, the Internet had the intended objective of connecting 'things'. The first Internet device was created in 1990 by John Romkey, consisting of a toaster that could be turned on and off remotely. The WearCam was invented in 1994 by Steve Mann. It had a near-real-time performance using a 64-bit processor system. Paul Saffo's gave the first brief description about sensors and their future course of action in 1997. In 1999, The Internet of Things term was coined by Kevin Ashton, executive director of the Auto-ID Centre, MIT. As a significant leap in commercializing IoT, in 2000 electronics giant LG announced its plans of revealing a smart refrigerator that would determine itself whether the food items stored in it are replenished. In 2003 RFID was deployed at a massive level in US army in their Savi program. In the same year, retail giant Walmart deployed RFID in all its shops across the globe. In 2005 mainstream publications like The Guardian, Scientific American and Boston Globe cited many articles about IoT and its future course. In 2008 a group of companies launched the IPSO Alliance to promote the use of Internet Protocol - IP - in networks of 'smart objects' and to enable the Internet of Things. Finally, the launch of IPv6 in 2011 triggered massive growth and interests in this field. Later IT giants like Cisco, IBM, Ericson took a lot of educational and commercial initiatives with IoT (Suresh et al., 2014).

The Internet of Things is growing, and sensors are being installed at ever-increasing rates. In not too long a fully deployed IoT will become a reality, with it comes vast promises for the future of Big Data, Analytics, and research in general (Hashemi, Faghri, Rausch, & Campbell, 2016). Projects in this are having a wide range of applications, since intelligent traffic system sensing, to frost depth monitoring, smart devices for healthcare and greenhouse and water quality monitoring. In all cases, standard threads include a sensor network, data collection, and usually a control or feedback component (McLeod, Ferens, & Friesen, 2016).

Innovation in IoT will undoubtedly be one of the drivers the future of technology. However, some challenge problems need to be addressed:

- Low-power sensing unit: A low-power sensing unit can work over a lifetime without the need for battery replacement. However, high-performance processing units or intelligent sensor modules may consume a massive amount of power and become a significant design problem that needs to be considered. One of the standards and appropriate techniques to create power-efficient high-performance units is by switching the analog transmitters to digital transmitters in wireless communications.

- High efficiency in connectivity: Through rapid growth and popularity in Internet technologies, many wired/wireless standards and protocols are defined. However, currently used standards and protocols may not be able to handle a significant amount of traffic from intelligent devices which connect to the Internet at the same time. Furthermore, the number of sensor devices is growing at a pace that the increasing rate of the available wireless spectrum cannot follow. To solve this problem, clustering techniques can be considered, as it will reduce network traffic by sending a massive amount of gathered information to a cluster system, which is nearest to the particular node, and then transmit to the base station, increasing the rate of network reusability.
- Reliable communication: Most wireless networks often create insecure connections that allow an intruder to access to the particular objects and sensitive information. Hence, the IoT essential needs at least an appropriate mechanism to prevent unauthorized access. One of the most commonly and widely used techniques to address these problems is encryption, to ensure data integrity. Another efficiency approach is defining security policies and regulations during transmission or reception data across the wireless network (Tuwanut & Kraijak, 2015).

2.2 **Technologies**

The purpose of this section is to describe the technologies used in this work, which relationships between them are described in chapters 4 and 5 of this report. It is divided between client and server-side technologies.

2.2.1 **The Client Side**

Browser

A web browser is a software that can query a remote web server, retrieving and displaying the resulting content, using a stateless and anonymous protocol called HyperText Transfer Protocol – HTTP -. The results are displayed using web pages, built using a language called Hypertext Markup Language and also an additional layer with Javascript, for client-side computation (Grosskurth & Godfrey, 2005).

Web3.js

Web3.js is a group of JavaScript libraries which enables - using HTTP or IPC - the interaction between an application’s front end - a JavaScript application using Web3.js - and its

backend - the Ethereum Blockchain -, using a local or remote Ethereum node. This Ethereum compatible API implements the Generic JSON RPC specifications ("JSON RPC API," 2018).

In Web3.js, there are two relevant modules, which contain specific functionality:

- web3.eth is explicitly used for Blockchain interactions and Smart Contracts;
- web3.shh is expressly used for whisper interactions.

JSON-RPC objects can be seen as passing back and forth continually between the front end- the HTTP Web - and the backend, the Ethereum Web (Dannen, 2017).

Websockets

A WebSocket is a bidirectional, full-duplex and single-socket connection. With this technology, the HTTP request is transformed in a single request to open a WebSocket connection - eventually using TLS, Transport Layer Security, formerly known as SSL -, and reuses the mentioned connection from the client to the server, and the server to the client. The server can send messages as they become available, which reduces latency. Also, the server does not need to wait for a request from the client, and similarly the client can send messages to the server at any time, simplifying the architecture (Wang, Salim, & Moskovits, 2013).

2.2.2 The Server Side

Rest Services

A service is a self-contained, self-describing and modular piece of software that performs a business function such as validating a credit card or generating an invoice. By being self-contained, it means that services include all that is required to get them working. Modular means that services can be aggregated to form more complex applications. Self-describing means they have interfaces that represent their business functionalities. Services are set to be standards-based, and platform-and-protocol-independent. A single service presents a collection of capabilities, often arranged together within a functional context as placed by business conditions.

Most importantly, REST is an architectural style of networked systems - not a specification, not a protocol - whose aim is to expose resources on a networked system, especially on the Web. It is based on the following core web standards:

- Resource identification – all resources are uniquely identified using a URI - Uniform Resource Identifier -. That is, all resources in a REST-based system are identified by a Web standard naming scheme.
- Unified resource interface – all resources are available to their potential client

applications via a set of HTTP operations. The HTTP operations will allow the resources to be retrieved, created, deleted and updated.

- Links and hypermedia – the resources in a REST-based system can be connected via various relation link types. These links are used by potential client applications to drive between diverse states of the same resource or different resources (Paik, Lemos, Barukh, Benatallah, & Natarajan, 2017).

Security - JWT

JSON Web Token, or JWT, is a standard for safely transferring claims in space-constrained environments. Usability, simplicity, and compactness are essential characteristics of its structure.

Claims are definitions or assertions made concerning a particular party or object and are defined as a member of the JWT spec, or user-defined. The purpose of JWTs is that they are capable of standardizing specific claims that are useful in the context of some traditional operations, like the establishment of the identity of a specific party.

Another core feature of JWTs is the possibility of signing them, using JSON Web Signatures and/or encrypting them, using JSON Web Encryption. Along with JWS and JWE, JWTs provide a robust, secure solution to many different problems (Peyrott, 2016).

Web3j

Web3j is a reactive, lightweight and type-safe library for integrating Java code with clients – nodes - on the Ethereum network, allowing the interaction with Ethereum Blockchain, without having to write integration code for the platform. It also supplies Smart Contract wrappers, to call its functions directly from Java classes. (Svensson, 2017).

Oraclize API

Oraclize is an Oracle service for Smart Contracts and Blockchain applications based on Ethereum, Bitcoin, and Rootstock.

In the Blockchain context, an Oracle is a party which provides data from external sources. This feature is needed because Blockchain applications cannot reach and retrieve directly external - outside the Blockchain context - data they require - for example, weather-related information for peer-to-peer insurance or price feeds for assets and financial applications-.

The solution developed by Oraclize prove the genuineness of the data fetched from the original data-source, which is also untampered. This evidence is performed by appending the returned data together with a document called authenticity proof, which can be built upon different technologies such as Trusted Execution Environments and auditable virtual machines.

This solution elegantly solves the Oracle Problem:

• Blockchain Application's developers and the users of such applications do not have to trust Oraclize, as the security model is maintained;

• Data providers do not have to modify their services to be compatible with Blockchain protocols, and Smart Contracts can directly access data from Web sites or APIs. (Oraclize, 2016).

InfluxDB

InfluxDB is a time series database to handle high write and query loads, for any use case including large amounts of timestamped data, including IoT sensor data, DevOps monitoring, application metrics and real-time analytics. It allows:

- Custom high-performance datastore explicitly wrote for time series data, allowing for high ingest speed and data compression;
- Simple, high performing write and query HTTP(S) APIs;
- Expressive SQL-like query language tailored to query aggregated data easily;
- Continuous queries automatically that compute aggregate data to make frequent queries more efficient;
- Tagging of concede series to be indexed for efficient and fast queries;
- Efficient auto-expiring of stale data due to retention policies (Dix, 2017).

Storj

Storj is a protocol that builds a distributed network for the formation and execution of storage contracts between peers, enabling them to negotiate contracts, transfer data, verify the integrity and availability of remote data, retrieve data, and pay other nodes, without meaningful human interaction. The removal of central controls would significantly increase privacy, security and data control as well as mitigate most traditional data failures and outages. Since data availability in P2P networks is a function of popularity rather than utility, it is unfeasible for production storage systems. In this regard, it Storj presents a challenge-response

confirmation system linked with direct payments, where periodic data integrity inspections are made, offering rewards to peers maintaining data (Wilkinson et al., 2016).

3 Related Work

3.1 Blockchain & IoT

3.1.1 Why does make sense to join these two concepts?

From a theoretical point of view, it is quite apparent the potential of joining IoT and Blockchain. In the manufacturer's point of view, the current centralized model of an IoT platform has a high maintenance cost, related to the distribution of software updates to millions of discontinued devices. From the consumer's side, it presents a justified lack of trust in devices in the background, being needed a 'security through transparency' approach. These issues can be solved with a scalable, trustless P2P model that can operate transparently and distribute data securely, which is in line with the core functionalities that Blockchain can provide.

Blockchain enables trustless networks as the parties can exchange data even if they do not trust each other, meaning a faster agreement between transaction actors. The massive use of cryptography, an essential characteristic of Blockchain networks, brings authoritativeness behind all the interactions in the network. Also, Smart Contracts integrate these concepts and allow for distributed and heavily automated workflows (Christidis & Devetsikiotis, 2016).

Figure 5 presents a set of current IoT challenges and possible solutions that Blockchain can provide.

Challenge	Explanation	Potential blockchain solution
Costs and capacity constraints	It is a challenge to handle exponential growth in IoT devices: by 2020, a network capacity at least 1,000 times the level of 2016 will be needed.	No need for a centralized entity: devices can communicate securely, exchange value with each other, and execute actions automatically through smart contracts.
Deficient architecture	Each block of IoT architecture acts as a bottleneck or point of failure and disrupts the entire network; vulnerability to distributed denial-of-service attacks, hacking, data theft, and remote hijacking also exists.	Secure messaging between devices: the validity of a device's identity is verified, and transactions are signed and verified cryptographically to ensure that only a message's originator could have sent it.
Cloud server downtime and unavailability of services	Cloud servers are sometimes down due to cyberattacks, software bugs, power, cooling, or other problems.	No single point of failure: records are on many computers and devices that hold identical information.
Susceptibility to manipulation	Information is likely to be manipulated and put to inappropriate uses.	Decentralized access and immutability: malicious actions can be detected and prevented. Devices are interlocked: if one device's blockchain updates are breached, the system rejects it.

Figure 5 – IoT challenges and possible solutions by Blockchain (Kshetri, 2017).

In short, the main advantages of combining these two paradigms are the following:

Cost effective storage system

Adopting peer-to-peer computing to process massive amounts of IoT transactions can dramatically reduce costs associated with managing large centralized data centers. Due to its efficiency and the small costs incurred by each host, compared to Amazon S3's $ 25 per terabyte per month, Blockchain storage could cost about $2 per terabyte per month (Sharma, Chen, & Park, 2017).

Decentralization and true redundancy

The scalable and efficient management of resources may well be one of the most critical objectives for the realization of the future IoT network. With the use of Blockchain technology, it is possible to create a distributed cloud data storage where data is stored in dozens of discrete nodes intelligently disbursed across the globe, making it extremely difficult to cause significant disruptions (Sharma et al., 2017).

Also, P2P storage technologies are characterized by robustness. Since some redundancy is added, even in the scenario that a peer shuts down and specific pieces of data get lost, it is still feasible to recover the original data. Examples of this capability are IPFS or Tahoe-LAFS (Conoscenti, Vetro, & De Martin, 2017).

Trust without a central authority

Successful decentralization of the IoT will depend not only on being peer-to-peer but also in being trustless, meaning that there is no need for members to be trusted and no centralized, single point of failure (Brody & Pureswaran, 2014).

By creating a trust without the need for a trusted third party, it is widely believed that Blockchains will overhaul antiquated cloud computing systems. At present, companies like Amazon offer a decentralized storage infrastructure cloud services. Nowadays, most organizations that have hosted their servers on their premises have moved to the cloud to reduce the number of servers and maintenance cost drivers; and by replicating data across multiple data centers, a cloud service like Amazon S3 can offer reliable uptime and redundancy and charge approximately $ 25 per terabyte per month. This convenience has shown that nowadays it is placed too much trust in third parties. Due to a lack of proper points of reference and low costs, we are obliged to trust these third parties to secure our most private and sensitive data, which are mostly unencrypted. Due to the parallelism between the financial and cloud

infrastructures, we can replace the existing systems with Blockchains and eliminate our reliance on trusted third parties (Sharma et al., 2017).

Improved Security

Blockchain-based identity/access management systems can be used to increase IoT security, being nowadays used to securely deposit information about goods' provenance, identity, credentials, and digital rights, as Blockchain's immutability can be accomplished as long as the original data entered is accurate. In this context, a significant challenge that appears in IoT applications is that it is tough to guarantee that the properties of physical assets, individuals – credentials -, resource use - energy and bandwidth through IoT devices -, and other relevant events are saved securely and reliably. This issue can be managed relatively easily for most IoT devices. As an example, a private Blockchain can be applied to save cryptographic hashes of device firmware. Such a system generates a permanent record of device state and configuration, being this record a possibility to verify that a specific device is genuine and that its software/settings have not been tampered with or breached, and only after this is the device allowed to connect to other services or devices. Because it is not possible to change approved Blockchains transactions, devices cannot join a network by altering themselves by injecting fraudulent signatures into the record (Kshetri, 2017).

Blockchain as a financial infrastructure to IoT

An essential future IoT functionality is to allow 'things' to automatically and ubiquitously make payments to other 'things' without any human interaction. This would, for example, allow the possibility for an IoT device to, on its own, decide to directly pay other devices for internet access or rent extra computational power when needed. The payments would typically be small, in a large number and should be autonomous. Due to limited capacity and high transaction charges, current traditional payment solutions are not well suited to handle these massive amounts of micro-transactions. It is not possible to payments in the IoT era using the traditional centralized payment approach.

Previous research suggests that Blockchain technology could be used to handle thing-to-thing payments, as this technology has numerous encouraging properties that make it a suitable candidate for handling thing-to-thing payments.

First, they are based on a decentralized P2P network for doing relaying and bookkeeping of transactions, and the decentralized nature of Blockchain cryptocurrencies allows it to sustain autonomous and numerous transactions.

Second, the account creation process is very lightweight and cheap – any device can merely have its account, and a new one can be created in instants without internet access. Therefore, there is no central authority in control of the accounts, which are not directly linked to anyone, only to the IoT device itself (Lundqvist, de Blanche, & Andersson, 2017).

Third, fees for foreign exchange transactions, remittances, credit card transactions and other products can be decreased considerably - $16 billion can be saved annually, equally one-third of transaction fees -. Recently, Blockchain intermediaries have been providing excellent Bitcoin transaction service in countries like Philippines and Kenya.

Last but not least, Blockchain helps boost the transaction speed and efficiency of execution, optimizing the time to be completed, which is currently up to 3 working days. (Nguyen, 2016).

3.1.2 **Disadvantages**

However, despite the demonstrated potential of joining IoT and Blockchain, some drawbacks need to be addressed to take full advantage of this combination:

Scalability

One of the principal obstacles to enable a decentralized IoT design supported by the Blockchain is the limited scalability of the most common Blockchain frameworks. For example, in the Bitcoin code, the maximum block size limits to 7 the number of transactions per second that can be written in the Blockchain, an idea which aims to limit the effort of running a full node. In fact, with an increase in this limit the throughput would be higher, but on the other hand, the Blockchain required storage would increase at a high pace, requiring more disk space. Lowering the number of full nodes - nodes that store all Blockchain data - could be a solution to an improvement in transaction throughput, but it will imply a more centralized system, as the number of nodes decreases and with it also the consensus's scope. Recent research is concentrating on how to scale Blockchain networks without changing the block size. One promising approach is the light protocols such as Bitcoin Light Network or Raiden Network – Ethereum - (Conoscenti, Vetro, & De Martin, 2016).

Throughput and Latency

Blockchain systems may have performance issues, concerning throughout and latency. In fact, the transaction processing rate per second – tps - for bitcoin network has a theoretical current maximum of 7 tps. This could be alleviated if each block were bigger, but a rise in blocksize would create other issues by promoting the existence of a Blockchain bloat. Example metrics in other transaction processing networks are VISA - 2,000 tps typical; 10,000 tps peak -, Twitter - 5,000 tps typical; 15,000 tps peak -, and advertising networks - >100,000 tps typical.

Also, each Bitcoin transaction block demands 10 minutes to process, indicating that it may take at least 10 minutes for the transaction to be verified, which may inhibit real-time applications using Blockchain technologies.

However, newer Blockchain applications than bitcoin are already trying to mitigate this issues, in order to provide instantaneous verified transactions (Swan, 2015).

Lack of privacy in Smart Contracts

Despite the expressiveness and power of Smart Contracts, they lack transactional privacy. The whole sequence of steps taken in a Smart Contract is propagated across the network and recorded on the transaction logs, being therefore publicly visible. Even though public keys are used to enhance anonymity, transaction values and balances are still publicly visible. Also, the Smart Contract source could be seen with a simple block explorer tool, as a working Smart Contract is deployed on the network with an associated address, and eventual vulnerabilities in the code could be explored.

This lack of privacy is a major obstacle towards the broad adoption of decentralized Smart Contracts since financial transactions must be secret (Kosba, Miller, Shi, Wen, & Papamanthou, 2016)

Lack of regulation in financial transactions

Incompleteness concerning legal and regulation on Bitcoin and cryptocurrencies limits Blockchain technology from being widely applied, making it very challenging for Blockchain to make a breakthrough in the payment industry.

A legal framework for digital currencies should be drafted considering both domestic and international measurements so that risks are minimized while a desire for freedom and innovation is still not stiffened. Policies should be designed so that the community benefits from Blockchain while preventing illegal use of Blockchain for such activities as money laundering, terrorism sponsorship, and even capital control (Nguyen, 2016).

3.1.3 **Architectural alternatives**

The architecture presented in this work – chapter 4, 'Development of a Solution Model' -, is highly modular, which means that every single component can be easily replaced. Alternatives for each component of the proposed design are presented below.

User Agent / Web Browser

A new generation of native applications - mobile or desktop -, like the ones based on Electron framework could be easily an alternative to the browser, as this technology can use JavaScript libraries (Electron, 2017). In our context, the mentioned feature is required since Web3.js JavaScript library must be used to generate the payment and the following subsequent steps for fetching sensor data.

Distributed Ledger / Ethereum + Smart Contract

There are many Blockchain platforms with different characteristics and features. In this section, we select and compare other platforms considered to be similar to Ethereum.

- In 2016, IBM open sourced Open Blockchain - OBC - project as part of the Linux Foundation's new Blockchain project, forming now a core part of Hyperledger Fabric project. One of IBM's goals with OBC is to propose business processes automation by deploying business rules as Smart Contracts on the Blockchain that can be validated by all stakeholders in a trusted way, serving as a platform to issue, trade, manage and service assets. The traditional Blockchain model, as it is, does not satisfy the requirements that make it appropriate for use in enterprise-critical applications. The lack of support for private transactions is an excellent example of this, but it can be addressed by OBC, as in this platform, nodes require permission to join the network. This is achieved by having authority on the network that can issue identities for entities to transact on the network, with these identities having distinct levels of permissions associated with them, called Membership Service Provider – MSC -. Additionally, Hyperledger Fabric offers the feature of creating channels, enabling a group of participants to build a separate ledger of transactions. This fact is relevant for networks where some members might be rivals and not want every transaction they make disclosed to every participant. If two participants build a channel, then only those participants have copies of the ledger for that channel.
- Sidechain Elements is a significant Blockchain innovation, as it proposes

'sidechains' - standalone Blockchains which can be combined into other Blockchains. It allows for the transfer of assets between two Blockchain networks and also introduces a degree of modularity using 'elements': confidential transactions - only those involved in the transaction can see the amount transferred - and Signature Covers Value - invalidates a transaction's signature when its inputs are spent, allowing for faster transaction validation -. Its native currency is pegged to the value of another Blockchain currency automatically by using proofs of payment. Some possible uses for sidechains include the production of Blockchains with advanced Smart Contracts or advanced privacy features. Rootstock – RSK - platform is an example of a sidechain, where the Rootcoin – RTC - is two-way pegged to the BTC - more specifically, a Rootoshi, the minimum unit of account in RSK, is pegged to a Satoshi, the minimum unit of account in Bitcoin -. In reality, when BTC is exchanged to RTC, no currency is 'transferred' between Blockchains in a single transaction because Bitcoin cannot verify the authenticity of balances on another Blockchain. When a transfer happens, some BTC is locked in Bitcoin, and the same amount of RTC is unlocked in RSK. When RTC needs to be converted back into BTC, the RTC get locked again in RSK, and the equal volume of BTC are unlocked in Bitcoin.

- Eris is a Blockchain platform for enterprise applications. Their nuclear capabilities are used to manage permissions, enabling the definition of who can create Smart Contracts, transact on the network and validate transactions. This platform is formed by Ethereum VM, the Tendermint Socket Protocol for consensus, the mint-client for talking to Terndermint, Eris's key signing daemon and a Solidity compiler. Because Eris uses the Ethereum VM, it also requires knowledge of Solidity or Serpent, which are programming languages specific to the Ethereum VM. They are based on JavaScript and Python, two reasonably well-known languages. Eris also currently requires developers to find their proper key signing daemon, which requires knowledge in this domain.

Regarding a general overview comparing Ethereum and the mentioned alternatives, we concluded that Ethereum was the best solution in terms of documentation, support, development, and scalability. It also has no identified security issues, with some security improvements planned for the near future. OBC is also a valid Blockchain platform although it has some issues, as some lack of maturity due to being quite recent and not massively adopted, unlike Ethereum. Eris and Sidechain Elements have potential as general-purpose Blockchain

platforms, particularly for enterprise applications that require permissioned consensus mechanisms. However, like OBC, they are both too underdeveloped at this time to be used for developing Blockchain-based applications. In the end, although Blockchain platforms have significantly progressed since the concept was first introduced as part of Bitcoin, it will be interesting to see how they continue to develop in the future (Macdonald, Liu-Thorrold, & Julien, 2017).

Blockchain Oracle - Oraclize

Oraclize is a part of what is called a software Oracle. This type of Oracle can cryptographically attest the origin of the data – TLSNotary - and push the information to a Smart Contract. It performs a public claim about the contents of secure web pages and provide an actionable gateway for a decentralized application. There are some technical pros and cons of such a design, but the principal issue is that we must trust the source of information, which can be not so linear. (Larchevêque, 2016). However, we can have other Oracle types, such as those described below.

- Consensus-based Oracles

ChainLink is a decentralized Oracle network. This decentralized strategy restricts the trust in any single party, enabling the tamperproof quality valued in Smart Contracts to be extended to the interaction between Smart Contracts and the APIs they rely on. (Ellis, Juels, & Nazarov, 2017).

Augur is a trustless, decentralized platform for prediction markets. It is an extension of Bitcoin Core's source code, built from Bitcoin's input/output-style transactions. It incorporates the features – betting and consensus mechanisms – required by prediction markets. Tradeable Reputation is a crucial feature of Augur, which the total amount is a fixed quantity, defined upon the launch of Augur. Keeping Reputation empowers its owner to report on the outcomes of events after the events occur. These tokens are similar in other respects to Bitcoins: they are divisible to eight decimal places, they are accounted for by summing over unspent transaction outputs, and they can be sent between users (Peterson & Krug, 2014).

- Hardware Oracles

The Town Crier – TC - system addresses the problem of ensuring trust in Smart Contract external data fetch by using trusted hardware, specifically the Intel SGX instruction

set. TC collects data from destination URLs defined in queries from application contracts. TC uses SGX to achieve what is designated by authenticity property. Considering that SGX is trustworthy, data read by TC from a website to an application with Smart Contracts is proved to be free from tampering. This authenticity property means that to trust TC data, it is only needed to trust Intel's implementation of SGX and the target website (F. Zhang, Cecchetti, Croman, Juels, & Shi, 2016).

Hardware Pythias is another hardware Oracle alternative, more targeted for reading data from sensors. The following characteristics define it: a cryptographic proof of the sensor reading, authenticating the measure, as each device has a private key signing outgoing payloads - with a nonce to avoid replays -; an anti-tampering establishment of the reader device, presenting it inoperable - by wiping the private key - in case of manipulation attempt - connect to another object and inject false stimuli. - (Larchevêque, 2016).

IoT Device / Raspberry Pi + Sensors

The Arduino platform, connected with a DHT11 sensor can gather temperature and humidity values, similarly to the raspberry pi. The DHT11 sensor has a full range temperature compensation, low power consumption, long-term stability and calibrated digital signal. They are made of two parts, a capacitive humidity sensor, and a thermistor. It also has a basic chip that does some analog to digital conversion and spits out a digital signal with the temperature and humidity. This set is much cheaper than raspberry pi, although the programming model/interfaces are more complex (Krishnamurthi, Thapa, Kothari, & Prakash, 2015).

DB - Time Series Database / InfluxDB

Riak TS is a distributed NoSQL database, tuned for fast reads and writes of time series data. It presents resiliency and massive scalability - horizontally with commodity hardware -, making it easy for administrators to append capacity without creating complex sharding. It has the same basic features than InfluxDB, and it should also be mentioned the presence of the eventual consistency, a characteristic of distributed systems such that the value for a certain data item will, given adequate time without updates, be uniform across all nodes (McCrory, 2015).

Decentralized Storage / Storj

Filecoin implements a decentralized data storage and retrieval service via a network of independent storage providers that does not rely on a single supervisor, where clients pay to save and get data, retrieval miners earn tokens by serving data and storage miners collect tokens by offering storage. Filecoin works as an incentive layer on top of IPFS, which can give storage infrastructure for any data, especially for implementing Smart Contracts and building/running distributed applications (Protocol Labs, 2017).

Other consensus algorithms

With the Blockchain's evolution, other consensus algorithms started to be used rather than Proof of Work.

- The Proof-Of-Stake protocol varies from the Proof-Of-Work protocol as it makes mining new blocks more straightforward for who hold the most significant amounts of the cryptocurrency. This characteristic of the Proof-of-Stake protocol generates a reason for miners to spend their mined currency as opposed to converting it into fiat currency instantly upon mining, which produces downside stress on the cryptocurrency. Furthermore, Proof-Of-Stake protocols require less energy consumption than Proof-Of-Work-based Blockchains as computational power is not the driver of mining rewards, mitigating the risk of hardware centralization. Early Proof of Stake algorithms just included rewards for producing blocks and no punishments. Without this, incentives become misaligned if there are competing chains. Consequently, the Blockchain may never reach consensus, even in the absence of attackers. MintCoin and Nxt are examples of cryptocurrencies that are mining using the Proof-Of-Stake mechanism.
- In the Byzantine Fault Tolerance – PBFT - consensus creation, each node in the network publishes a public key. Later, when messages come into a node, it is signed by it to verify the message as having the correct format. Once enough same responses to the message are given, a consensus that the message is a valid transaction is achieved. The energy consumption is lower than in PoW and PoS-based Blockchain networks because PBFT consensus mechanism does not require any hashing power to validate transactions, also mitigating the risk of centralization.
- Regarding Hybrid Proof-Of-Work and Proof-Of-Stake, Peercoin is an example that uses both Proof-Of-Work and Proof-Of-Stake to verify its transactions, mitigating the

existing issues of a pure PoW consensus-based Blockchain. By joining the two consensus models, much lower volumes of energy is needed, and the risk of a 51 percent attack is also decreased considerably. (Seibold & Samman, 2016).

3.1.4 Real-world examples of Blockchain & IoT together

As already presented, combining Blockchain with IoT has significant potential, and some companies already started adopting this paradigm. Here, we present some actual examples of this combination:

Slock.it

Slock.it works on smart electronic locks – 'Slocks' - that can be unlocked with a device that provides the appropriate token, bought on the Ethereum. The owner of a Slock that wishes to rent their good establishes a price for timed access to that electronic door lock. An interested party can use a mobile app to identify the slock, pay the demanded amount in Ethers, and then communicate with the lock via an adequately signed message to unlock it (Christidis & Devetsikiotis, 2016).

Managing Supply Chains

A supply chain is one of the most used cases to demonstrate the value of a Blockchain. A typical flow is the following: a container that leaves the manufacturer's site, gets transported via railway to the nearby port, then gets shipped to the destination port, gets transported again to the distributor's facilities, until it finally reaches the retailer's site. This process involves several stakeholders and validations along the way. Each stakeholder usually holds their database to track the asset, which they update based on inputs from the other parties along the chain. With Blockchain technology, it is possible to have one shared database to keep track of, where updates come with cryptographic verifiability, get propagated along the network automatically, and an auditable trail of information is also created. These updates could be originated from IoT devices, like smart sensors - ex., automatic payment to the supplier after a specific asset arrives at the warehouse - (Christidis & Devetsikiotis, 2016).

On-Demand Manufacturing and Machine to Machine Transactions

IoT and Blockchain systems could facilitate a marketplace of manufacturing services where the devices will have their Blockchain accounts, and the users will be able to provision and transact with the machines directly to avail the mentioned services.

For example, a customer can send a request for manufacturing an item by sending a transaction to a manufacturer's Smart Contract along with the payment made in a cryptocurrency. The manufacturer's Smart Contract can then send transactions to Smart Contracts associated with individual devices. If the services of several devices are required for manufacturing a product, the devices can send micro-transactions to other devices - machine-to-machine transactions - (Bahga & Madisetti, 2016).

Smart Diagnostics & Machine Maintenance

IoT, along with Blockchain can be used for producing smart diagnostics and self-service applications for machines where they will be able to monitor their state, check for problems, and autonomously place service, consumables replenishment, or part replacement requests to the machine maintenance merchants. Smart Contracts between manufacturers and vendors for procurement of supplies and service of machines can help in automating the machine maintenance tasks (Bahga & Madisetti, 2016)

Product Certification

In this context, the manufacturing information for a product - facility and machine details, manufacturing date and parts information - can be stored on a Blockchain network. This consolidated and shared data can be used to ensure the authenticity of the products, eliminating the need for physical certificates which can be suitable to tampering and forge (Bahga & Madisetti, 2016).

Smart Cities

In recent years, the world has encountered unprecedented urban growth due to population increase, climate change, and scarcity of resources. To deal with these pressures, cities are focusing on modern technologies to create a longer livable urban environment. Meaningful advancements in IoT and wireless communications have made it simple to interconnect a variety of devices and enable them to send data ubiquitously. However, it has been shown that smart terminals such as bicycle rental terminals, self-service machines, and information kiosks have some security flaws. The cybercriminals can abuse these devices, and they may obtain access to critical data.

Therefore, new solutions must be developed to provide privacy, integrity, and data

confidentiality, features that can be found in Blockchain technology, allowing entities in a smart city to communicate securely. The core benefit of using Blockchain in this context is that it provides many unique characteristics such as better fault tolerance capability improved reliability, faster and efficient operation, and scalability, creating a collaborative platform where all devices would be able to communicate securely in a distributed environment (Biswas & Technology, 2016).

IBM Adept - decentralized IoT

The main goal of the Adept was to establish a foundation on which to show several capabilities that are crucial to building a decentralized IoT.

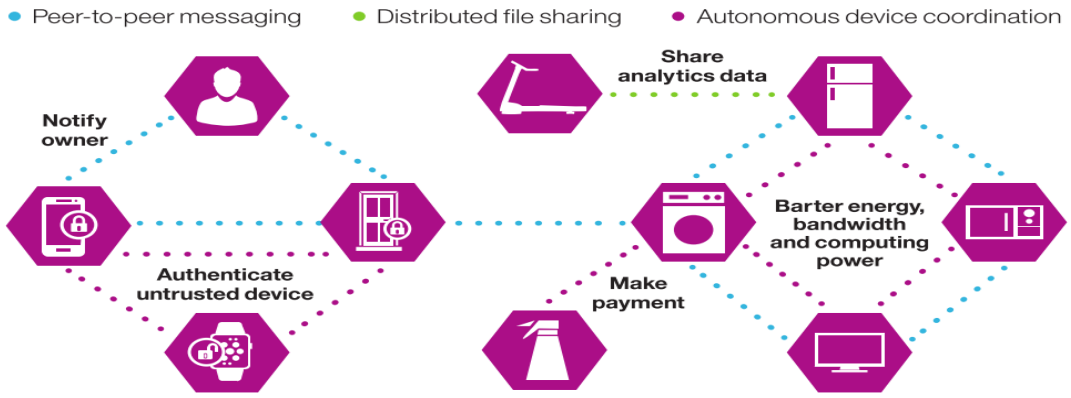

Figure 6 – IBM adept architecture (Pureswaran, Panikkar, Nair, & Brody, 2015).

This project has validated the practicability of implementing the foundational functions of a decentralized IoT. Adept opens the door for the electronics industry to investigate further the difficulties and possibilities of potential hybrid models that can approach the complexity and variety of demands posed by an Internet that continues to scale.

To achieve the functions of traditional IoT solutions without a centralized broker, any decentralized approach must support three foundational functions (cf. Figure 6):

- P2P messaging;
- Distributed file sharing;
- Autonomous device coordination.

The Adept implemented these functions using three open source protocols: Telehash for messaging, BitTorrent for file sharing and Ethereum for autonomous device coordination

functions such as device registration, authentication, proximity-based and consensus-based rules of engagement, contracts and checklists (Pureswaran et al., 2015)

4 Development of a Solution Model

4.1 Architecture overview and steps

To answer the research question, a system was designed that allows buying data from a specific sensor - temperature and humidity -, generating a query with specific criteria and returning the results to the client that requested it. Blockchain technology is used as a backend to process the service payment and the transactions for obtaining the sensor records. The following figure represents a conceptual overview of the architecture adopted for the proof of concept used in this work:

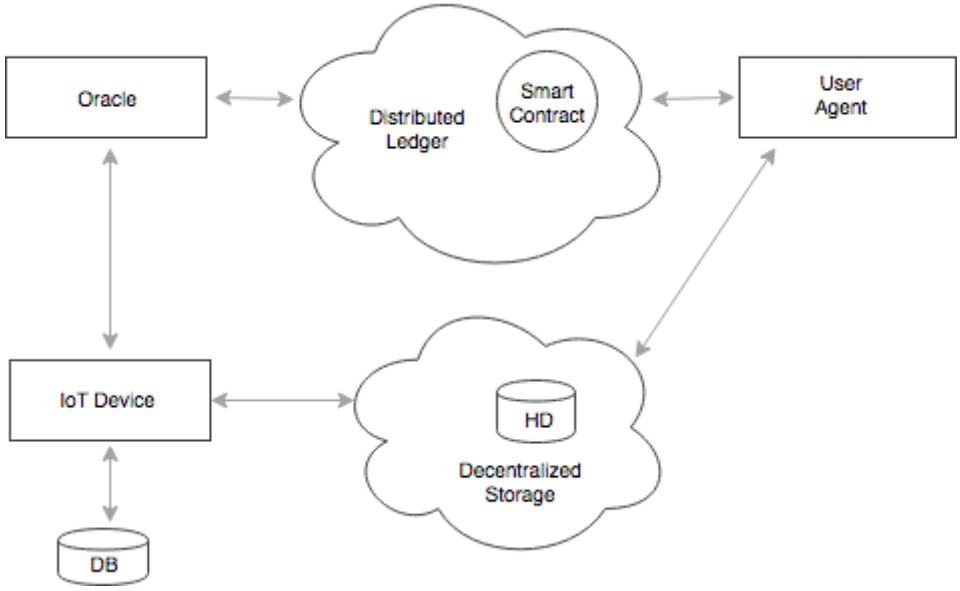

Figure 7 – Conceptual design of the proposed architecture.

4.2 Components of the architecture

4.2.1 User Agent

An interface that allows a particular entity - ex: a person - to specify which type of data – and the criteria -, to be queried and returned, after being paid. It interacts with the distributed ledger via Smart Contract – to manage all the associated transactions -, and with the decentralized storage, for downloading the result of the specified query (Wang et al., 2013).

4.2.2 Distributed Ledger

A component consisting in a consensus of replicated, shared and synchronized data spread across multiple nodes. It can also be a database held and updated independently by each node in an extensive network. Participants in the network govern and agree by consensus on the updates to the records in the ledger. No central, third-party mediator is involved. This every

single node on the network processes every transaction, coming to its conclusions and then voting on those conclusions to make sure the majority agree with the conclusions. Once there is this consensus, the distributed ledger has been updated, and all nodes maintain their identical copy of the ledger ("Blockchain basics: Introduction to distributed ledgers," 2017; Coindesk, 2017). It interacts with the user agent, receiving the query and returning the result, and with the Oracle component, for fetching the data from the sensor.

4.2.3 **Smart Contract**

A computer protocol, with business logic embedded that allows the establishment of an agreement between two parties. In the context of this work, it serves as an intermediary between the distributed ledger and the Oracle, passing the query request and returning the file response (Garrod, 2016).

4.2.4 **Oracle**

Because Smart Contracts only live in the Blockchain context, an Oracle is a component used for searching and verify Blockchain-external data, submitting it to be used by the Smart Contracts (Oraclize, 2016). It interacts with the sensor, querying the data that matches the criteria passed and returning the result to the Smart Contract.

4.2.5 **IoT Device**

An IoT device is a computing device that connects to a network and can capture and transmit data, a 'thing' on the internet of things (Bahga & Madisetti, 2016). It interacts with the database, fetching the records that match the criteria and with the distributed storage to save the file containing the referred records and with the Oracle, returning the hash of the stored file.

4.2.6 **Database**

An organized collection of data that can be accessed managed and updated (Bellevue Linux Users Group, 2006). It has the device data, interacting with it by returning the records that match the specified criteria.

4.2.7 **Decentralized Storage**

A computer network where information is stored on more than one node, often in a replicated way, which behaves as one storage system although data is distributed between the nodes (Wilkinson et al., 2016). It differs from typical storage because it is more flexible, fast and cheap. It interacts with the IoT device receiving the file with the sensor data, storing it and

returning a file hash. That hash will return to the user-agent, which will use it to download the file.

4.3 **Summary**

This design provides a system that enables data exchanging between a requester and a producer - sensor data owner -, allowing the monetization of the corresponding transaction and highlighting the advantages of the joint use of IoT/Distributed Ledger instead of a typical client/server application with a central database. Its modular design allows a simple component replacement if a better alternative is found, which is likely to happen due to the fast growth of the technologies associated with the IoT/Distributed Ledger set. This architecture is explained regarding technical details in the next section.

5 Implementation of the Solution Concept

The proposed design was implemented with a set of technologies that represent each step/component of the system. Open-source platforms were preferred due to their large developer communities, which allowed having quick and insightful hints for the development process. All the associated code in the used platforms is adjacent to this document (cf. Appendix III).

A vital aspect of this architecture is the capability of the client paying instantly for the data and the data owner of receiving it, which is not easy to do if standard banking/payment systems are used. Apart from security and immutability, this was the main reason for using the Blockchain, and in particular, the Ethereum platform, as it is possible to transfer ether - Ethereum token - within the transactions.

Also, web technologies - HTML, JavaScript, WebSockets, and RESTful Web Services - were preferred to connect with the core part of this system – Ethereum and Smart Contracts – mainly because they are supported on a set of standards that allow robust and straightforward access to the data being exchanged.

Here we describe in detail each component used and already referred in the conceptual architecture.

5.1 Components of the Proof of Concept

The following components were considering regarding the implementation of the proposed architecture, for answering the research question. References to ‘steps’ in the below sub-sections refer to the various steps described in section 5.1.7 - Proof of Concept Interplay.

5.1.1 Web Browser and Websockets

In this work, the requesting entity is a web browser, through which it can trigger the entire process of buying data (cf. Appendix III – sensor-client).

Transactions, Smart Contracts, and Oraclize calls could take a varied amount of time to execute, being uncertain the exact duration spent. Since the browser wants to be immediately notified when the result file is available, WebSocket’s full duplex communication capabilities are suitable for this case, since it means that data can be sent either way on the connection at any time (Wang et al., 2013). In this work, after service payment is made, a WebSocket connection is established between the browser (cf. Appendix III – sensor-services, src/main/resources/js/ws-server.js) and a remote host - step 6 - from where the ‘Oraclize’ REST

service will be called. After the file has returned to the browser, the WebSocket connection is automatically closed - step 22 -. The WebSocket server was built using Node.js platform.

To secure the WebSocket communication, an SSL certificate was generated - self-signed - and used within the connection. This allows the use of WebSockets over the HTTPS protected protocol – WSS (Websocket Secure) -. WSS is encrypted, thus protecting against man-in-the-middle attacks (Panagiotakis, Kapetanakis, & Malamos, 2013; Wang et al., 2013).

This component is mapped to 'User Agent' - section 4.2.1 of architecture's design.

5.1.2 **Ethereum and Smart Contract**

Blockchain technology is a type of distributed ledger, and it was chosen for this work, more precisely by using Ethereum platform and some associated libraries – Web3js and Web3j.

Web3js library is used to generate a signed transaction for transferring the service payment - step 2 -. Due to security reasons, it is generated on the client side – browser -, as we do not want to transfer the user account file and its corresponding private key to the server. First, the browser establishes a connection with a configured Ethereum node on a remote server, uses the uploaded Json account file – unlocked via password typing - and creates a wallet object where the account private key is contained. Then, a raw transaction is created – with an associated value, the payment -, and signed using the referred private key. Next, the transaction is sent and processed on the network. If the result is successful, the client will be notified and will start the WebSocket connection - step 6 -. By using this kind of system, data owner immediately receives the service fee, unlike other payment systems (cf. Appendix III – sensor-client, index.html and js/main.js).

Web3j library is used to interact with our Smart Contract deployed on Ethereum network when receiving a call from the 'Oraclize' REST service - step 7 -. First, it unlocks the data owner account present on the Ethereum node. Then, a transaction is created to call the 'update' function of the developed Smart Contract, passing the query criteria as an argument - step 8 -. Because the Oraclize API call is paid, the corresponding value is transferred within the transaction. Next, a filter is created that waits for the occurrence of the event 'newQuerySensor', which will contain the file hash filled on the Smart Contract 'update' function. In the end, this filter is added to an observable object, which will start the referred wait until the result is returned (cf. Appendix III – sensor-services, src/main/java/org/meisi/rest/Oraclize.java).

The integration between the service invoked from the WebSocket connection, and the

sensor data is made using a Smart Contract. In this scenario, a Smart Contract called 'QuerySensor' was developed and deployed on the Ethereum network, using solidity as the language and truffle framework to manage the deployments. It has an 'update' function, which is triggered after the service fee payment and connects with Oraclize API, which will return the sensor data corresponding to the criteria - steps 10 and 11 -. The result is inserted on a Smart Contract event, stored on transaction log, likely to be caught by Web3j from 'Oraclize' REST service (cf. Appendix III – sensor-contract, /contracts/QuerySensor.sol).

This component is mapped to 'Distributed Ledger' and 'Smart Contract' - sections 4.2.2 and 4.2.3 of architecture's design.

5.1.3 **Oraclize**

Because Smart Contracts only live in the Ethereum network context, they are not able to fetch external data, which is a requirement in this architecture. Due to that, we used Oraclize API to query the sensor database by invoking the 'sensor-data' REST service (cf. Appendix III – sensor-services, src/main/java/org/meisi/rest/SensorData.java). This is done by using 'oraclize_query' instruction inside 'update' function of the 'QuerySensor' contract - step 11 -. It will generate an HTTP get call to 'sensor-data' REST service - step 13 -, and the result will return via the '__calback' - steps 17, 18 and 19 - a function of the contract, creating the 'newQuerySensor' event (cf. Appendix III – sensor-contract, contracts/QuerySensor.sol).

This component is mapped with 'Oracle' - section 4.2.4 of architecture's design.

5.1.4 **Raspberry Pi and Sensors**

The sensor device used was a raspberry pi 3, and its data was simulated using a python script, which was run at periodic intervals, inserting data on a time series database - InfluxDB. The measures simulated were temperature and humidity.

This component is mapped to 'IoT Device' - section 4.2.5 of architecture's design.

5.1.5 **InfluxDB**

To store sensor data correctly, we decided to use InfluxDB. It disposes the data in a time series format, which matches the typical IoT sensor data structure. The data is queried from the 'sensor-data' REST service invocation, using the InfluxDB java driver for connecting the database and in this way select the records that match the specified criteria - step 13 -.

This component is mapped to 'Database - DB' - section 4.2.6 of architecture's design.

5.1.6 **Storj**

After querying the InfluxDB database, the returned records are stored in a temporary file and must be stored safely. For safe storage, we decided to use Storj because is also based on Blockchain technology and has high encryption standards, availability, performance and a robust and straightforward API. Using the referred API, the temporary file is uploaded to Storj storage - step 14 -, which returns a file hash - step 15 - that will return to Smart Contract context via the oraclize API, being stored on transaction log - Ethereum event - as mentioned above (cf. Appendix III – sensor-services, src/main/resources/js/storj-files/storj-service.js).

This component is mapped to ‘Decentralized Storage’ - section 4.2.7 of architecture’s design.

5.1.7 **Proof of Concept Interplay**

The following figure represents the real implementation of the conceptual architecture described in the previous section.

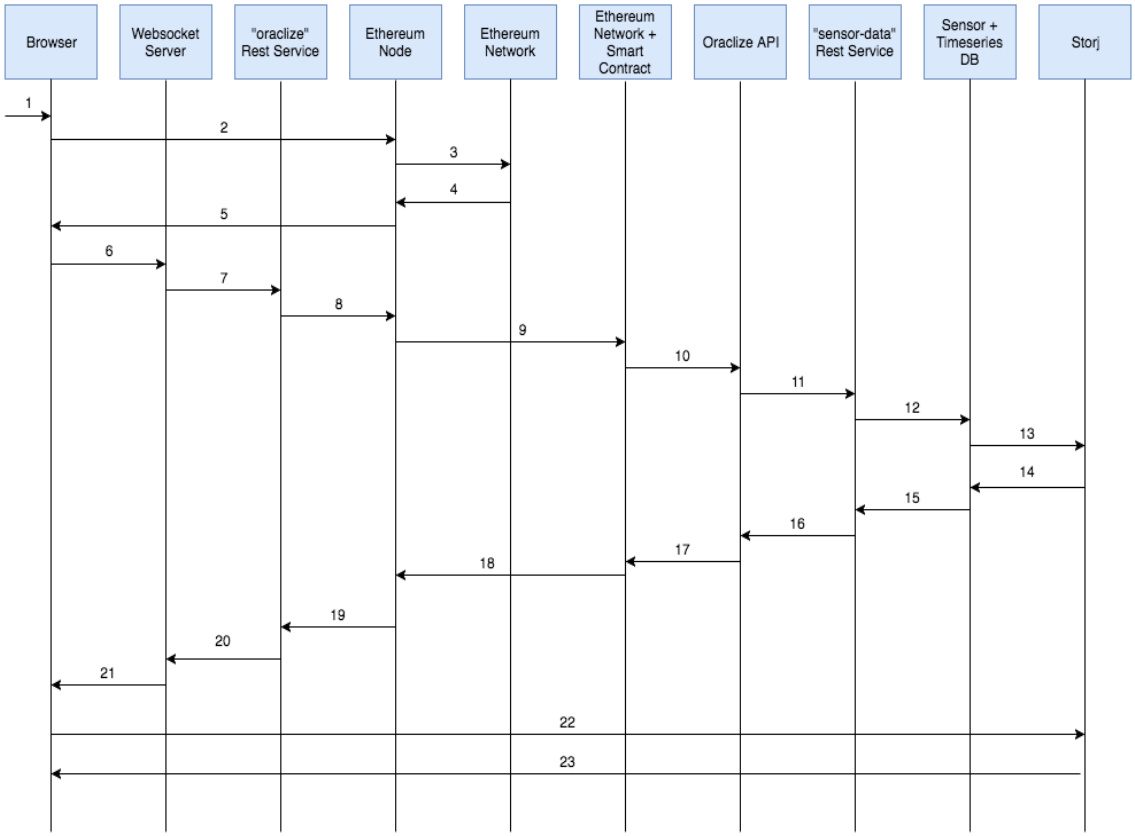

Figure 8 – Flow diagram and implementation architecture for the proof of concept

Steps

1. A user goes to the browser, inserts its Json Ethereum account file with the password, and specifies the criteria query, pressing on submit button.
2. The ‘submit button’ event triggers the creation of a signed transaction ‘tx1' - generated on the browser –, which intends to pay the service - ETH transfer to service owner account;
3. tx1 submitted on Ethereum network;
4. tx1 execution;
5. tx1 result - ok/not ok - returned to the browser;
6. If tx1 result ok, then establish a secure – WSS - connection with WebSocket server;

7. HTTP get call to ‘Oraclize’ REST service, with the query criteria from the browser - step1 -;

8. ‘Oraclize’, using the Web3j library, creates and sends a transaction ‘tx2’ to call the ‘QuerySensor’ Smart Contract ‘update’ function. With tx2 it is sent an amount of ETH, to pay the use of Oraclize API;

9. tx2 submitted on Ethereum network;

10. Smart Contract execution;

11. During Smart Contract execution, the Oraclize API is called, to invoke ‘sensor-data’ service that will query the time series database on the sensor machine;

12. tx2 finishes here its execution;

13. HTTP get call to query the sensor database using the criteria sent as an argument from the client - step1 -;

14. Results of the query are stored in a file in Storj storage, using its API (HTTP post request);

15. Storj returns a hash of the stored file;

16. The hash is returned through the ‘sensor-data’ REST service result;

17. The Oraclize API creates a transaction ‘tx3’ to handle the returned hash, corresponding to the invocation of ‘_calback’ function of the ‘QuerySensor’ Smart Contract;

18. tx3 submitted and tx3 executed. The hash is stored on tx3 data, and an event is fired;

19. The event is caught on ‘Oraclize’ service side and the event data - file hash - is fetched;

20. The ‘Oraclize’ service returns the file hash to previously opened WebSocket connection;

21. The WebSocket server sends a message to the browser with the file hash;

22. The browser receives the file hash, building an HTML link; the link is clicked, which corresponds to an HTTP get call to the Storj API for downloading the file;

23. The file is downloaded from the browser.

Standardization was also a priority in this work, which led to the use of well-known architectural design principles like REST services, to manage the connection and information exchange between some components of this system. Two services were developed: one to

interact with Ethereum network from the user agent, other to make the integration between Oraclize API and the sensor database (cf. Appendix III – sensor-services, src/main/java/org/meisi/rest/ SensorData.java and Oraclize.java). Javax.ws packages from java language and Spring framework were used to build the services.

When dealing with data exchange, privacy and security issues could be a concern. To mitigate it, JWT standard was used to authorize the use of 'Oraclize' REST service - step 8 -. This was done by implementing a JWT authorization server associated with the REST services package. From the WebSocket connection, an HTTP post request is made to the JWT authorization URL – using a valid username/password on the server -, requesting a valid JWT token (cf. Appendix III – sensor-services, src/main/java/org/meisi/security and sensor-services, src/main/resources/js/ws-server.js). If the supplied credentials are valid, the requested JWT token is generated and sent to the WebSocket, which will use it to call the 'Oraclize' service in an authenticated and authorized way. ("JSON Web Tokens," 2015; Peyrott, 2016). Privacy and security were also a concern when establishing the WebSocket connections, by using WSS. The WSS - WebSocket Secure - URI scheme describes a WebSocket connection above Transport Layer Security - TLS or SSL -, and works with the same security mechanism that HTTPS uses to secure HTTP connections, protecting the integrity, confidentiality, and availability of the network communications (Wang et al., 2013).

5.1.8 **Entity Interaction**

The following figures illustrate the interaction between the entity who wants to buy the data and the implementation model.

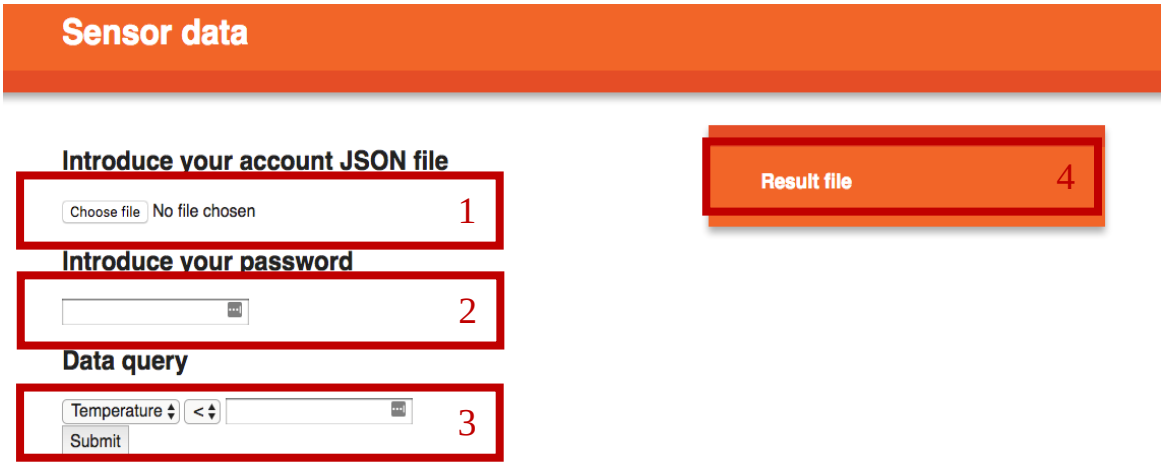

Figure 9 – Web interface for querying and buying the data.

'1' is the component to the upload the Json account file, '2' the account password input field, '3' the component to specify the criteria and submitting the query, '4' the component where file hash will be displayed as soon as it is available.

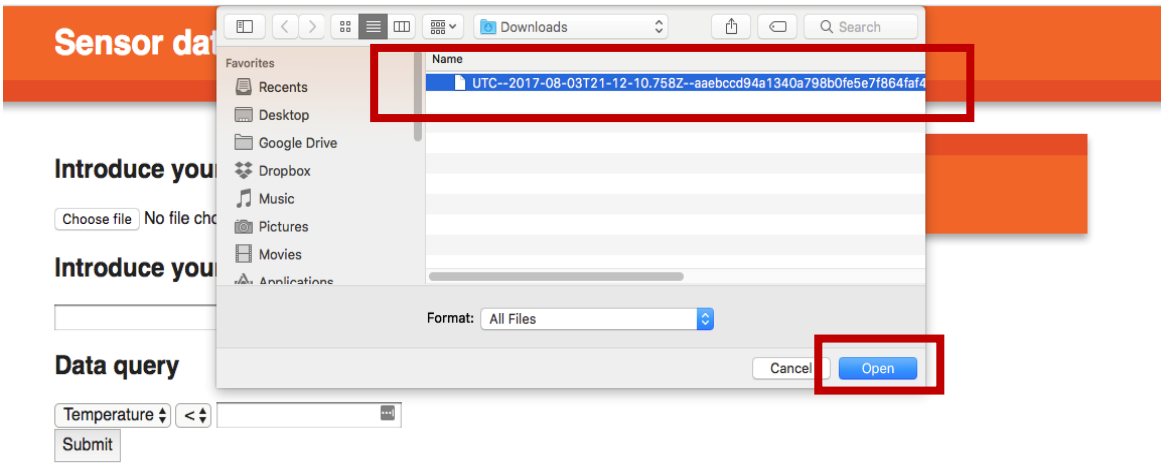

Figure 10 – Uploading Json account file.

It contains the account private key, needed to generate a signed transaction for transferring the service fee.

Figure 11 – Setting account password and data query.

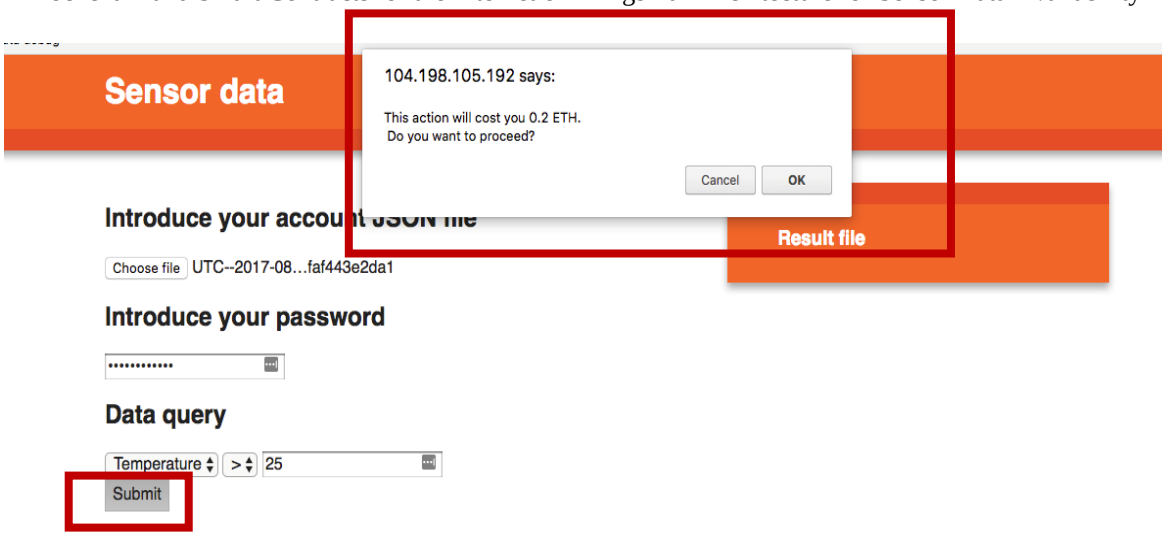

Figure 12 – Submitting query and cost warning.

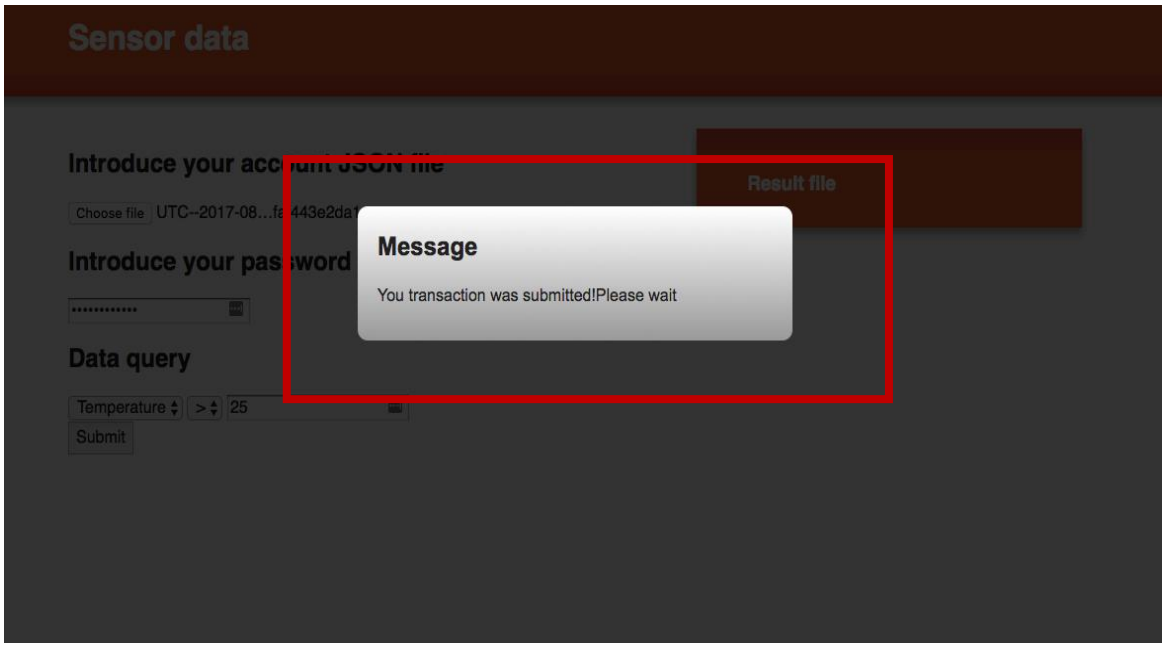

Figure 13 – Transaction submission and waiting for the result.

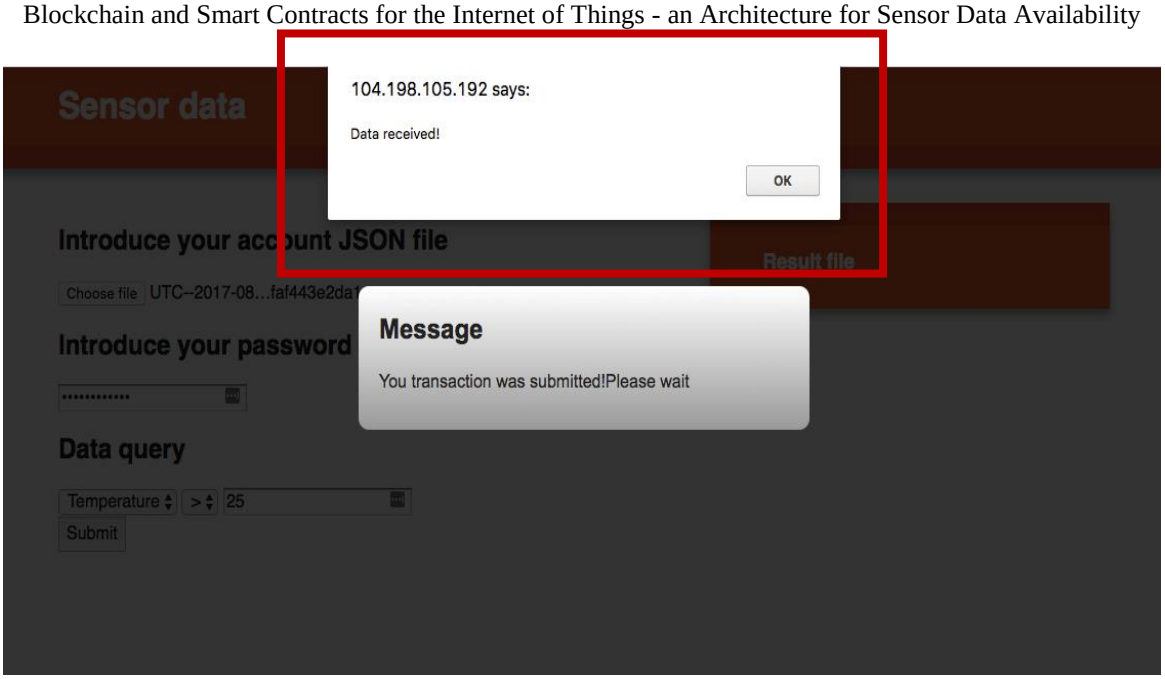

Figure 14 – Data received from the sensor.

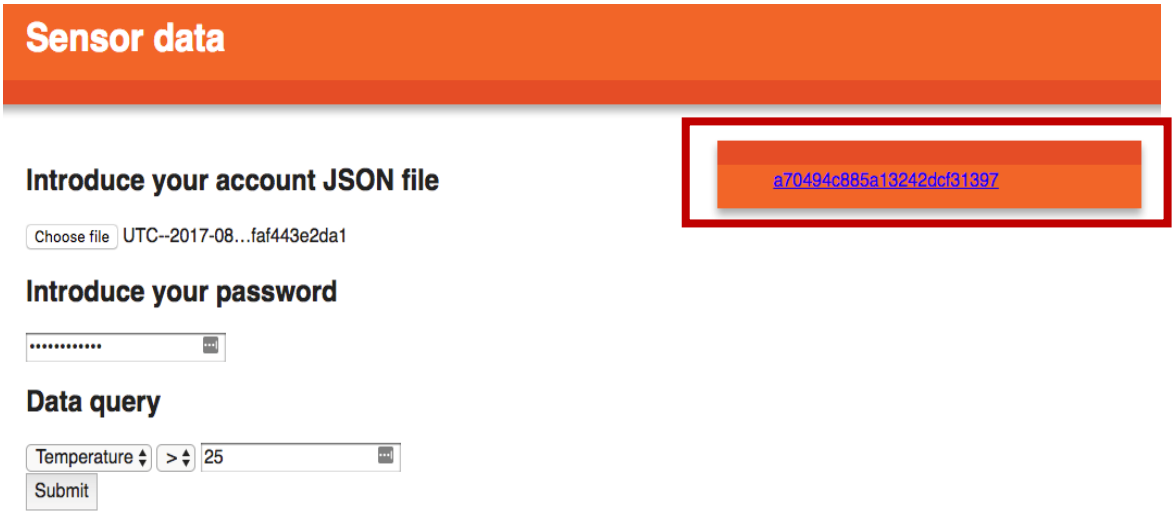

Figure 15 – File hash returned in the browser.

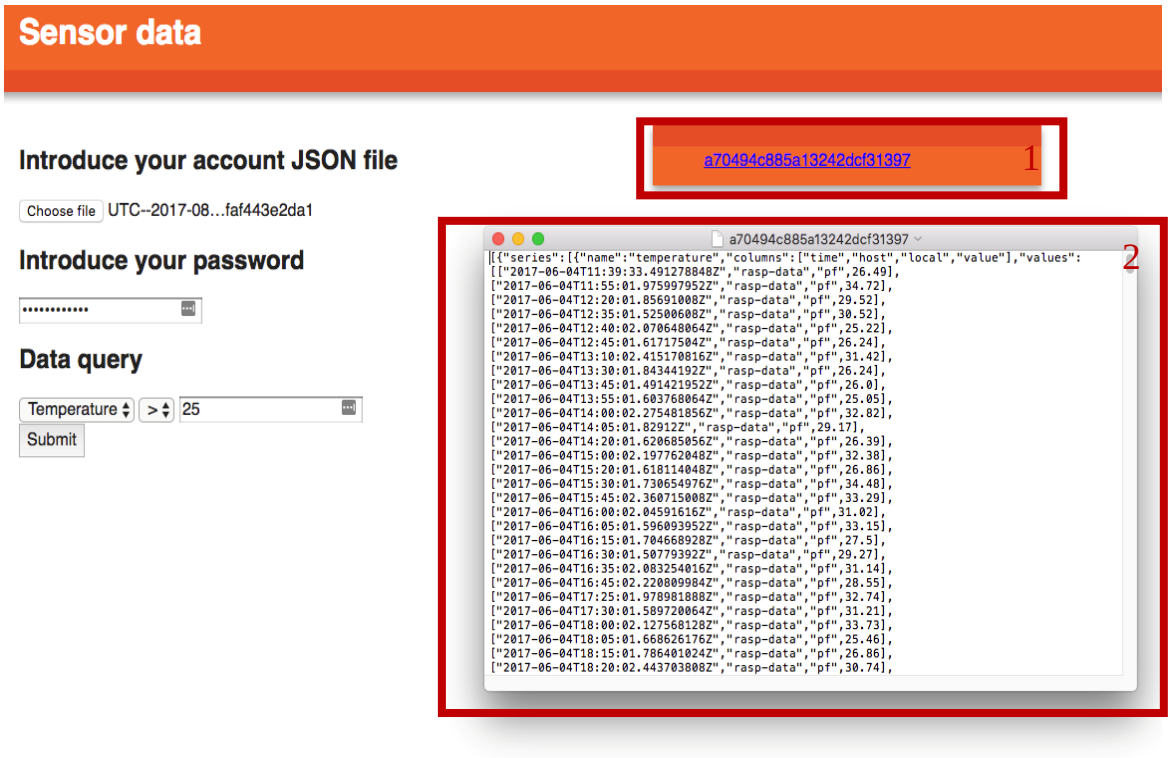

Figure 16 – File download and checking results.

'1' is the file hash, which is also a link to download the file. '2' is the downloaded file, where after opening it the results can be verified.

6 Results

After implementing the proposed design, several tests were conducted to validate if this architecture correctly responds to the research question. In other words, that once a transaction for data acquisition is validated, the corresponding access to the sensor is granted in the shortest delay possible.

To validate this assumption, the solution was analysed from two important aspects: the execution costs of the transactions, and the associated execution time, the ideal scenario being a good balance between these two.

Some authors (Yasaweerasinghelage, Staples, & Weber, 2017) report that gas price and confirmation time are related, although this relationship is not clearly understood. To characterize it better, we performed several experiments and measures, using the Ethereum Testnet - ropsten network - as infrastructure and source of data.

The first experiment consisted in measuring the total elapsed time between the user transaction submission instant in the browser, and the reception of the file hash, for different values of the gas price - 0.5, 1, 25 and 100 gwei -. The same query - 'temperature < 20' - was used ten times for each different gas price. This elapsed time includes the confirmation time of transactions tx1, tx2 and tx3, mentioned in the Proof of Concept Interplay section.

Table 1 – Results in total elapsed time varying the gas price.

	0.5 gwei	1 gwei	25 gwei	100 gwei
Average of elapsed time (seconds)	276,6837	212,7174	228,0287	200,129
Standard deviation	185,862789	35,5034385	39,4760413	37,5394452

In Table 1 the mentioned transaction average elapsed times and standard deviations are shown, for the above-mentioned gas price values.

However, according to (Yasaweerasinghelage et al., 2017) gas price is not the only factor affecting transaction confirmation time. In fact, the network delays, the submitted transaction fee, the current number of ongoing transactions, as well as the strategic choices made by miners can also influence the transaction confirmation time.

In this context and according to (Elkins, 2017), in the presence of stable network conditions - the main contributors to confirmation time are the following, assuming that the transaction is not being sent to/from a high-volume transaction address:

- gas price,
- hash power accepting the proposed gas price,
- number of transactions at or above the gas price in txpool,

Besides, other authors (Hu et al., 2018) have concluded that block finding in Blockchain systems - like Ethereum – can be modelled using a statistical approach. More precisely, the number of blocks expected to be found in a certain elapsed time follows a Poisson distribution.

Additionally, the number of found blocks necessary to confirm a transaction, usually named 'blockWait', is a particular instance of the mentioned block finding concept and is linearly related to the transaction confirmation time through the following expression:

$$confirmation\_time = blockWait * blockInterval$$

where $blockInterval$ represents the delay between blocks and is a constant value – assuming network stability conditions. Consequently, a Poisson regression model can be applied to Ethereum transaction timing data, with the purpose of capturing the relation between 'blockWait' – the variable that follows a Poisson distribution – and the main contributors to confirmation time mentioned above (Elkins, 2017).

Therefore, a second experiment was performed, with the goal of understanding how the mentioned variables relate to the confirmation time in the Ethereum Ropsten testnet, by performing a Poisson regression, assuming, as mentioned above, that *blockWait* follows a Poisson distribution and is somehow related to the referred influence factors. Then, $confirmation\_time$ can be calculated from *blockWait*, according to the expression introduced above.

Their corresponding values were captured on the Ropsten testnet, using two python scripts (cf. Attachment I) that, once connected to the Ethereum node, extract the required information from executed transactions. The results of fitting these values to a Poisson regression model are shown below, where a significance level of 0,05 is being considered. The Minitab software version 18 was used for the statistical calculations.

Table 2 – Statistical summary of the Poisson regression model.

Source	DF	Adj Dev	Adj Mean	Chi-Square	P-Value
Regression	38	1337,61	35,200	1337,61	0,000
hashpower_accepting	1	3,55	3,550	3,55	0,060
tx_atabove	1	570,08	570,076	570,08	0,000
round_gp_10gwei	36	891,32	24,759	891,32	0,000
Error	11068	2782,82	0,251		
Total	11106	4120,43			

In Table 2 'hashpower_accepting' is the percentage of hash power accepting the proposed gas price, 'tx_atabove' is the number of transactions at or above the gas price in txpool and 'round_gp_10gwei' is the gas price in gwei unit. It's possible to see that hashpower_accepting - P-Value - is not statistically significant – it is greater than 0.05 -, in opposition to 'tx_atabove' and 'round_gp_10gwei'.

Table 5 (cf. Appendix I) shows the regression coefficients for the considered variables. Since the relationship between time and gas price is non-linear, the gas price is modelled as a categorical variable. In Table 6 (cf. Appendix II), the regression equation for each value of gas price present in the captured data is presented.

After obtaining the model, a goodness test was performed to assess if it is a fit for the presented transaction data.

Table 3 – Poisson Goodness-of-Fit Test for confirmation time – Chi-Square Test

Null hypothesis	**H_0: Data follow a Poisson distribution**	
Alternative hypothesis	H_1: Data do not follow a Poisson distribution	

DF	Chi-Square	P-Value
7	18939,3	0,000

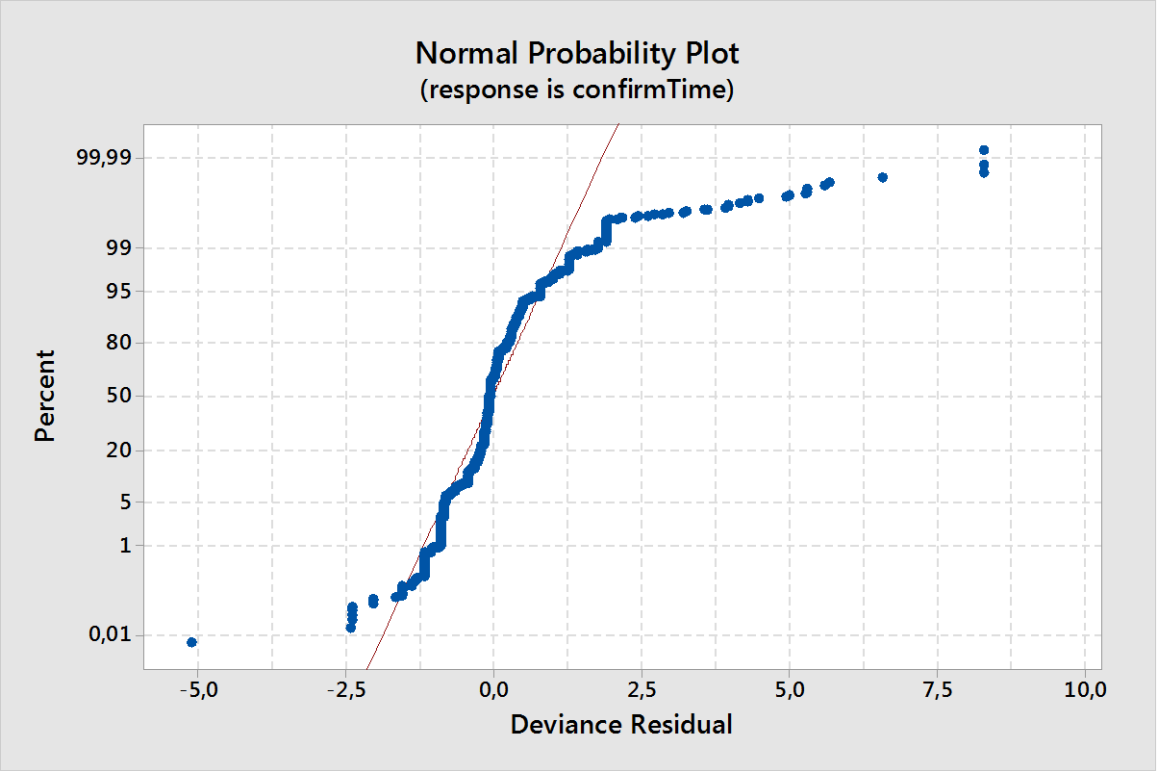

Figure 17 – Normal probability plot of the residuals, displaying the residuals versus their expected values when the distribution is normal.

Table 4 – blockWait vs predicted blockWait

hashpower_accepting	round_gp_10gwei	tx_atabove	blockWait	Predicted blockWait
0	1	2	3	22,2121
0	0	100	5	12.4863
99	10	68	18	2,25332
98	40	5	21	1,63862

In Table 4, the comparison between predicted and observed blockWait was made for the most distant points – two points with the lowest residual 'negative' deviance and two points with the highest residual 'positive' deviance -, in normal probability plot - Figure 17. 'blockWait' is the 'blockWait observed' and 'Predicted blockWait' are the predicted values for this variable.

7 Conclusions and Perspectives

7.1 Conclusions

It is clear from the results that the gas price is not the only variable that influences confirmation time. The high standard deviation values registered on the first experiment seems to corroborate this scenario, as the elapsed times measured consecutively varied substantially within the same gas price and query, and the relation between these two values is not linear – elapsed time grows from 1 to 25 gwei, as opposed to what is empirically expected - the decrease of confirmation time with the rise of gas price. From the second experiment, we can conclude that the proposed regression model does not provide a good fit for all the data, as the P-Value is smaller than the established significance level - Table 3. In fact, from Figure 17 it is obvious the presence of outliers, which is supported by the comparison between observed 'blockWait' and 'predicted blockWait' - Table 4.

A possible global explanation for these discrepancies lies in the fact that the tests were conducted in the Ethereum Ropsten testnet, which suffers from network stability issues. In a testnet, many nodes are continually being removed and added, which causes relevant and frequent changes in the network hashrate and in the number of available miners – which affect the number of transactions at or above the gas price in txpool - tx_atabove from regression model. Other possible reason for this oscillation of values - in both experiments - is the use of the Oraclize API. Since this API relies on the execution of calls to third-party systems – in this scenario REST services, InfluxDB and Storj - whose performance is not controllable by Ethereum, it becomes very complex to predict its performance, since the proposed regression model is applied only to typical Ethereum transactions – within the chain -, which is not the case. Besides, the current Oraclize documentation does not provide enough detail about the API dynamics as to allow predicting its transactions confirmation times. It would have been useful to deploy the implemented architecture in a private Ethereum chain and repeat the experiments to compare the results, given that in such a scenario it would be possible to define and manage some variables such as hashrate and network latency, which would undoubtedly improve our model predictability and performance.

In general terms, it may be considered that the presented implementation responds positively to the research question, since it proved that a system, designed and implemented according to the Blockchain paradigm, namely the Ethereum platform, allowed clients to transact and receive sensor data as a consequence of previously negotiated contracts. Dynamic

aspects, such as performance, confirmation time and cost should nevertheless be the subject of more detailed analysis to improve the solution, but this work clearly demonstrates the potential of the IoT and Blockchain paradigms combined.

7.2 **Future Research**

Although it was possible to develop a proof of concept that performs in acceptable terms, some areas of improvement were identified, that can produce both a better client experience and a more flexible response.

Firstly, a concept like 'transaction rollback' should be implemented. Due to its modular nature, failures may occur in different parts of the implemented architecture. As it is based on Ethereum transactions, this could be a problem because Blockchain transactions are immutable, and therefore there is no possibility of rollbacking them after being confirmed (Sankar, Sindhu, & Sethumadhavan, 2017). For example, in step 6 of the model, if something wrong happens in the WebSocket connection, there is no way to rollback transaction 'tx1', because it was already confirmed. This means that although clients have already paid for the service, they will not receive the file hash. To avoid these issues, an escrow service can be used, i.e.: instead of transferring the service fee directly to the data owner, the client transfers it first to an escrow entity, which is responsible for holding the funds until the paid data is retrieved. Only after is the money transferred to the data owner, or returned to the client in case an error happens during the data retrieving process (Lu & Xu, 2017).

Secondly, it should be possible to apply different data pricing models, since in our case, the same value is always charged - 0.1 ETH - regardless the number of records returned. For instance, a concept such as Smart Data Pricing (SDP), a research initiative that uses pricing incentives to enhance network performance and support data management (Sen, Joe-Wong, Ha, & Chiang, 2013), could be used. According to this approach, during busy periods, a flexible pricing can be applied to defer non-urgent users from entering the network, thus increasing quality of service – QoS – and performance. Instead of using static pricing - usage-based method or byte-counting, which requires clients to pay the corresponding price per unit of data reached, SDP implements flexible and dynamic pricing tools based on the demand and requirements of the users (Niyato et al., 2016). For adopting an SDP dynamic pricing approach in this work, two additional improvements are necessary:

1) to estimate the number of returned records, to be called after the client specifies the criteria and before the transaction being submitted - step 1 on the proposed model -, the service

fee being proportional to the total amount of returned records;

2) to allow the client to set the gas price for the service, in a qualitative scale – low 'cheaper', medium and high returning speed 'more expensive' -, as it is known that confirmation time decreases with higher gas prices.

A final last area of improvement worth mentioning is the possibility of implementing real-time scenarios. Although Blockchain can guarantee high stability and data reliability, the current solutions still have a few critical scalability barriers, such as low-throughput and high-latency. As it was demonstrated in the results section, predicted transaction confirmation time in Ethereum is far from the real-time scenario. The EBCM - E-commerce Blockchain Consensus Mechanism (Y. Xu et al., 2017) - is a possible alternative, since it constructs a two-layer Blockchain, called the Peer Blockchain, to ensure high-throughput and real-time transactions. The benefit of creating a second layer in a blockchain system lies in the fact that it minimizes the amount of data saved onto the underlying ledger, thus decreasing the load and therefore increasing transaction throughput, while keeping the whole process decentralized. To ensure transaction credibility, EBMC also includes a separate chain dedicated to block validation: the Validation Blockchain (cf. E-commerce Blockchain Consensus Mechanism). By exploring a solution like EBMC, real-time and reliable transactions could be implemented, which would bring significant benefits to several IoT business scenarios (Y. Xu et al., 2017).

References

Aron, J. (2015). Automatic world. *New Scientist*, *227*(3038), 18–19. https://doi.org/10.1016/S0262-4079(15)31168-4

Bahga, A., & Madisetti, V. K. (2016). Blockchain Platform for Industrial Internet of Things. *Journal of Software Engineering and Applications*, *9*, 533–546. https://doi.org/10.4236/jsea.2016.910036

Bellevue Linux Users Group. (2006). Database Definition. Retrieved December 9, 2017, from http://www.linfo.org/database.html

Biswas, K., & Technology, A. B. (2016). Securing Smart Cities Using Blockchain Technology. *2016 IEEE 18th International Conference on High Performance Computing and Communications*, 5–6. https://doi.org/10.1109/HPCC-SmartCity-DSS.2016.178

Blockchain basics: Introduction to distributed ledgers. (2017). Retrieved October 10, 2017, from https://www.ibm.com/developerworks/cloud/library/cl-blockchain-basics-intro-bluemix-trs/index.html

Brody, P., & Pureswaran, V. (2014). *Device democracy Saving the future of the Internet of Things IBM*.

Buterin, V. (2014). A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM.

Christidis, K., & Devetsikiotis, M. (2016). Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, *4*. https://doi.org/10.1109/ACCESS.2016.2566339

Coindesk. (2017). What is a Distributed Ledger? Retrieved October 10, 2017, from https://www.coindesk.com/information/what-is-a-distributed-ledger/

Conoscenti, M., Vetro, A., & De Martin, J. C. (2016). Blockchain for the Internet of Things: A systematic literature review. *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, *4*, 2292–2303. https://doi.org/10.1109/AICCSA.2016.7945805

Conoscenti, M., Vetro, A., & De Martin, J. C. (2017). Peer to Peer for Privacy and Decentralization in the Internet of Things. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* (pp. 288–290). IEEE. https://doi.org/10.1109/ICSE-C.2017.60

Dannen, C. (2017). *Introducing Ethereum and Solidity* (1st ed.). Apress. https://doi.org/10.1007/978-1-4842-2535-6

De Jong, L. (2015). *Decentralized Link Sharing, Towards a Framework for Decentralized Applications*. University of Groningen.

Dix, P. (2017). InfluxData | Documentation | InfluxDB Version 1.3 Documentation. Retrieved October 6, 2017, from https://docs.influxdata.com/influxdb/v1.3/

Electron. (2017). Using Native Node Modules | Electron. Retrieved October 29, 2017, from https://electron.atom.io/docs/tutorial/using-native-node-modules/

Elkins, J. (2017). Progress in Dynamic Gas Price Estimation – ETH Gas Station – Medium. Retrieved November 28, 2017, from https://medium.com/@ethgasstation/progress-in-dynamic-gas-price-estimation-7afa16b56c5a

Ellis, S., Juels, A., & Nazarov, S. (2017). *ChainLink A Decentralized Oracle Network*. Retrieved from https://link.smartcontract.com/whitepaper

Ethereum. (2017). Contracts — Solidity 0.4.18 documentation. Retrieved October 6, 2017, from http://solidity.readthedocs.io/en/develop/contracts.html#events

Garrod, J. Z. (2016). The real world of the decentralized autonomous society. *TripleC*, *14*(1), 62–77.

Grosskurth, A., & Godfrey, M. W. (2005). A reference architecture for Web browsers. In *21st*

IEEE International Conference on Software Maintenance (ICSM'05) (pp. 661–664). IEEE. https://doi.org/10.1109/ICSM.2005.13

Hashemi, S. H., Faghri, F., Rausch, P., & Campbell, R. H. (2016). World of Empowered IoT Users. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)* (pp. 13–24). IEEE. https://doi.org/10.1109/IoTDI.2015.39

Hu, Y., Manzoor, A., Ekparinya, P., Liyanage, M., Thilakarathna, K., Jourjon, G., … Ylianttila, M. E. (2018). A Delay-Tolerant Payment Scheme Based on the Ethereum Blockchain. https://doi.org/arXiv:1801.10295v1

Huh, S., Cho, S., & Kim, S. (2017). Managing IoT devices using blockchain platform. In *2017 19th International Conference on Advanced Communication Technology (ICACT)* (pp. 464–467). IEEE. https://doi.org/10.23919/ICACT.2017.7890132

Johnston, D., Yilmaz, S. O., Kandah, J., Bentenitis, N., Hashemi, F., Gross, R., … Mason, S. (2014). The General Theory of Decentralized Applications, DApps.

JSON RPC API. (2018). Retrieved from https://github.com/ethereum/wiki/wiki/JSON-RPC

JSON Web Tokens. (2015). Retrieved July 20, 2016, from https://jwt.io/

Kosba, A., Miller, A., Shi, E., Wen, Z., & Papamanthou, C. (2016). Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In *2016 IEEE Symposium on Security and Privacy (SP)* (pp. 839–858). https://doi.org/10.1109/SP.2016.55

Krishnamurthi, K., Thapa, S., Kothari, L., & Prakash, A. (2015). Arduino Based Weather Monitoring System. *International Journal of Engineering Research and General Science*, *3*(2), 452–458. Retrieved from http://pnrsolution.org/Datacenter/Vol3/Issue2/64.pdf

Kshetri, N. (2017). Can Blockchain Strengthen the Internet of Things? *IT Professional*, *19*(4), 68–72. https://doi.org/10.1109/MITP.2017.3051335

Larchevêque, E. (2016). Hardware Pythias: bridging the Real World to the Blockchain - Ledger. Retrieved October 7, 2017, from https://www.ledger.fr/2016/08/31/hardware-pythias-bridging-the-real-world-to-the-blockchain/

Lu, Q., & Xu, X. (2017). Adaptable Blockchain-Based Systems: A Case Study for Product Traceability. *IEEE Software*, *34*(6), 21–27. https://doi.org/10.1109/MS.2017.4121227

Lundqvist, T., de Blanche, A., & Andersson, H. R. H. (2017). Thing-to-thing electricity micro payments using blockchain technology. In *2017 Global Internet of Things Summit (GIoTS)* (pp. 1–6). IEEE. https://doi.org/10.1109/GIOTS.2017.8016254

Macdonald, M., Liu-Thorrold, L., & Julien, R. (2017). The Blockchain: A Comparison of Platforms and Their Uses Beyond Bitcoin. https://doi.org/10.13140/RG.2.2.23274.52164

McCrory, D. (2015). *RIAK ® TS DATASHEET*. Retrieved from http://basho.com/wp-content/uploads/2015/04/Basho-Riak-TS-Datasheet-WEB.pdf

McLeod, R. D., Ferens, K., & Friesen, M. R. (2016). The IoT: Examples and trends. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 336–339). https://doi.org/10.1109/CSCI.2015.130

Millet, J. (2014). Danish Political Party May Be First to Use Block Chain For Internal Voting. Retrieved July 14, 2016, from http://www.newsbtc.com/2014/04/22/danish-political-party-may-first-use-block-chain-internal-voting/

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System. Www.Bitcoin.Org*. Retrieved from https://bitcoin.org/bitcoin.pdf

Nguyen, Q. K. (2016). Blockchain - A Financial Technology for Future Sustainable Development. In *2016 3rd International Conference on Green Technology and Sustainable Development (GTSD)* (pp. 51–54). IEEE.

https://doi.org/10.1109/GTSD.2016.22

Niyato, D., Hoang, D. T., Luong, N. C., Wang, P., Kim, D. I., & Han, Z. (2016). Smart data pricing models for the internet of things: a bundling strategy approach. *IEEE Network*, *30*(2), 18–25. https://doi.org/10.1109/MNET.2016.7437020

Oraclize. (2016). Oraclize Documentation. Retrieved October 6, 2017, from https://docs.oraclize.it/#home

Paik, H. H.-Y., Lemos, A. L., Barukh, M. C., Benatallah, B., & Natarajan, A. (2017). *Web Service Implementation and Composition* (1st ed.). Springer International Publishing. https://doi.org/10.1007/978-3-319-55542-3

Panagiotakis, S., Kapetanakis, K., & Malamos, a G. (2013). Architecture for Real Time Communications over the Web. *International Journal of Web Engineering 2013*, *2*(1), 1–8. https://doi.org/10.5923/j.web.20130201.01

Peterson, J., & Krug, J. (2014). *Augur: a Decentralized, Open-Source Platform for Prediction Markets*. https://doi.org/10.13140/2.1.1431.4563

Peyrott, E. (2016). *JWT Handbook* (0.12.0). Auth0.

Pilkington, M. (2015). Blockchain Technology: Principles and Applications. In F. X. Olleros & M. Zhegu (Eds.), *Research Handbook on Digital Transformations* (p. 39). Dijon. Retrieved from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2662660

Protocol Labs. (2017). *Filecoin: A Decentralized Storage Network*. Retrieved from https://filecoin.io/filecoin.pdf

Pureswaran, V., Panikkar, S., Nair, S., & Brody, P. (2015). *Empowering the Edge: Practical Insights on a Decentralized Internet of Things*. Retrieved from https://www-935.ibm.com/services/multimedia/GBE03662USEN.pdf

Raval, S. (2016). *Decentralized Applications - Harnessing Bitcoin's Blockchain Technology*. (T. McGovern, C. Lobner, & Octal Publishing, Eds.) (1st ed.). O'Reilly Media, Inc.

Sankar, L. S., Sindhu, M., & Sethumadhavan, M. (2017). Survey of consensus protocols on blockchain applications. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICACCS.2017.8014672

Seibold, S., & Samman, G. (2016). *Consensus - Immutable agreement for the Internet of Value*. Retrieved from https://assets.kpmg.com/content/dam/kpmg/pdf/2016/06/kpmg-blockchain-consensus-mechanism.pdf

Sen, S., Joe-Wong, C., Ha, S., & Chiang, M. (2013). *Smart Data Pricing (SDP): Economic Solutions to Network Congestion*. Retrieved from http://sigcomm.org/education/ebook/SIGCOMMeBook2013v1_chapter5.pdf

Shah, S. H., & Yaqoob, I. (2016). A survey: Internet of Things (IOT) technologies, applications and challenges. In *2016 IEEE Smart Energy Grid Engineering (SEGE)* (pp. 381–385). IEEE. https://doi.org/10.1109/SEGE.2016.7589556

Sharma, P. K., Chen, M.-Y., & Park, J. H. (2017). A Software Defined Fog Node based Distributed Blockchain Cloud Architecture for IoT. *IEEE Access*, *PP*(99), 1–1. https://doi.org/10.1109/ACCESS.2017.2757955

Suresh, P., Daniel, J. V., Parthasarathy, V., & Aswathy, R. H. (2014). A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. In *2014 International Conference on Science Engineering and Management Research (ICSEMR)* (pp. 1–8). IEEE. https://doi.org/10.1109/ICSEMR.2014.7043637

Svensson, C. (2017). web3j - Lightweight Java library for integration with Ethereum clients. Retrieved October 10, 2017, from https://docs.web3j.io/

Swan, M. (2015). *Blockchain - Blueprint for a new economy*. (T. McGovern, R. Monaghan, &

B. Russell, Eds.), *O'Reilly Media, Inc.* (First). Retrieved from http://shop.oreilly.com/product/0636920037040.do

Tuwanut, P., & Kraijak, S. (2015). A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends. In *11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015)* (pp. 26–31). Institution of Engineering and Technology. https://doi.org/10.1049/cp.2015.0714

Wang, V., Salim, F., & Moskovits, P. (2013). The Definitive Guide to HTML5 WebSocket. *The Definitive Guide to HTML5 WebSocket*, 208. https://doi.org/10.1007/978-1-4302-4741-8

Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A., & Kishigami, J. J. (2015). Blockchain contract: A complete consensus using blockchain. In *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)* (pp. 577–578). IEEE. https://doi.org/10.1109/GCCE.2015.7398721

Wilkinson, S., Boshevski, T., Brandoff, J., Prestwich, J., Hall, G., Gerbes, P., … Pollard, C. (2016). *Storj - A Peer-to-Peer Cloud Storage Network*. Retrieved from https://storj.io/storj.pdf

Wood, G. (2014). *Ethereum: a secure decentralised generalised transaction ledger*. Retrieved from http://yellowpaper.io/

Wright, A., & De Filippi, P. (2015). Decentralized Blockchain Technology and the Rise of Lex Cryptographia, 58. Retrieved from http://papers.ssrn.com/abstract=2580664

Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., … Rimba, P. (2017). A Taxonomy of Blockchain-Based Systems for Architecture Design. In *2017 IEEE International Conference on Software Architecture (ICSA)* (pp. 243–252). https://doi.org/10.1109/ICSA.2017.33

Xu, Y., Li, Q., Min, X., Cui, L., Xiao, Z., & Kong, L. (2017). E-commerce Blockchain Consensus Mechanism for Supporting High-Throughput and Real-Time Transaction. In S. Wang & A. Zhou (Eds.), *Collaborate Computing: Networking, Applications and Worksharing: 12th International Conference, CollaborateCom 2016, Beijing, China, November 10--11, 2016, Proceedings* (pp. 490–496). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-59288-6_46

Yasaweerasinghelage, R., Staples, M., & Weber, I. (2017). Predicting Latency of Blockchain-Based Systems Using Architectural Modelling and Simulation. In *2017 IEEE International Conference on Software Architecture (ICSA)* (pp. 253–256). IEEE. https://doi.org/10.1109/ICSA.2017.22

Zhang, F., Cecchetti, E., Croman, K., Juels, A., & Shi, E. (2016). Town Crier: An authenticated data feed for smart contracts. In *23rd ACM Conference on Computer and Communications Security, CCS 2016* (pp. 270–282). https://doi.org/10.1145/2976749.2978326

Zhang, Y., & Wen, J. (2017). The IoT electric business model: Using blockchain technology for the internet of things. *Peer-to-Peer Networking and Applications*, *10*(4), 983–994. https://doi.org/10.1007/s12083-016-0456-1

Glossary

Blockchain: a type of a distributed ledger in which all transactions were verified, recorded and are immutable, being shared by the participants of the network (Watanabe et al., 2015).

Blockchain Oracle: An Oracle, in the context of Blockchains and Smart Contracts, is an agent that detects and validates real-world occurrences and submits this data to a Blockchain to be used by Smart Contracts. Smart Contracts carry value and only unlock that value if specific pre-defined conditions are met. When a value is given, the Smart Contracts alternates its state and runs the predefined algorithms, automatically triggering an event on the Blockchain. The primary task of Oracles is to present these values to the Smart Contracts in a secure and trusted way (Oraclize, 2016).

Decentralized Application: Applications that run on a peer-to-peer network rather than a particular machine. Dapps have present since the advent of P2P networks, being a type of software designed to exist on the internet in a way that is not controlled by any single entity (Raval, 2016).

Decentralized Storage: A computer network where information is stored on more than one node, often in a replicated way, which behave as one storage system although data is distributed between the nodes (Wilkinson et al., 2016)

Distributed ledger: A component consisting of a consensus of replicated, shared and synchronized data spread across multiple nodes. It can also be a database held and updated independently by each or node in a vast network (Coindesk, 2017).

E-commerce Blockchain Consensus Mechanism – a consensus algorithm that does not rely on computing power and tokens but achieves the same level of security and credibility as Nakamoto consensus. Instead, it uses a negotiated consensus algorithm, where consensus is reached on the link value of the validation block. Each validation peer in the validation network generates a random number, signs it, and then sends it to other validation peers. When a validation peer receives all verification peers' random numbers, it combines them to form the link value. Since information may be lost or peer failure can occur, two-time thresholds - TV1

and TV2 – are set to prevent such situations. If the time that peer A waits for peer B's random number is greater than TV1, peer A requests peer B's random number to the verification network. And if the time is greater than TV2, peer B is removed from validation network. The remaining peers reach a consensus on the link value (Y. Xu et al., 2017).

Ethereum: An open-source, public, Blockchain-based distributed computing platform highlighting Smart Contract functionality. It implements a decentralized Turing-complete virtual machine, the Ethereum Virtual Machine – EVM -, which can run scripts using a global network of public nodes. Ethereum also produces a cryptocurrency token called 'ether', which can be shifted between accounts and used to reward participant nodes for computations performed (Dannen, 2017).

Ethereum mainnet: the real-world Ethereum network, used for production purposes, where data on the chain—including account balances and transactions—are public, and anyone can create a node and begin verifying transactions. Ether on this network has a market value and can be exchanged for other cryptocurrency or fiat currencies like US Dollars. (Dannen, 2017).

Ethereum ropsten testnet: one of the Ethereum existing network for testing purposes, where ether doesn't have a real value (Dannen, 2017).

Internet of Things: The IoT technology can be defined as a connection between humans, computers and things. All the equipment we use in our daily life can be controlled and monitored using the IoT concept (Suresh et al., 2014).

IoT Device: A computing device that connects wirelessly to a network and has the ability to capture and to transmit data. A 'thing' on the internet of things (Bahga & Madisetti, 2016).

Smart Contract: Created by Nick Szabo, Smart Contracts are essentially digital contracts that are enforced automatically by a set of computer protocols (Garrod, 2016). Ethereum is one of the Blockchain engines that enables 'Smart Contracts', which enforce agreements between two parties in an automatized way.

Smart Data Pricing: a suite of pricing and policy strategies proposed or being explored as pricing options instead of the traditional flat-rate model. Such models can include the following mechanisms:

a) Usage-based pricing/metering/throttling/capping;

b) Time/location/congestion-dependent pricing;

c) App based pricing/sponsored access;

d) Paris metro pricing;

e) Quota-aware content distribution;

f) Reverse billing or Sponsored Content (Sen et al., 2013).

Appendices

Appendix I

Table 5 - Regression coefficients for the considered variables – 'hashpower', 'tx_atabove' and 'round_gp_10gwei'.

Term	Coef	SE Coef	VIF
Constant	2,3888	0,0681	
hashpower_accepting	0,001237	0,000658	4,89
tx_atabove	0,001358	0,000058	2,53
round_gp_10gwei			
1	0,709	0,140	1,31
10	-1,7913	0,0936	54,83
20	-1,728	0,231	1,12
30	-1,724	0,710	1,01
40	-2,0230	0,0793	29,58
50	-1,712	0,175	1,34
60	-1,821	0,509	1,04
70	-1,729	0,505	1,02
100	-1,3784	0,0900	7,07
130	-1,765	0,187	1,33
150	-1,816	0,154	1,58
200	-1,8213	0,0938	6,28
210	-1,8622	0,0952	21,64
220	-1,769	0,184	1,35
230	-1,594	0,274	1,13
250	-1,846	0,135	1,90
300	-1,821	0,254	1,16
310	-1,818	0,713	1,02
360	-1,821	0,713	1,02
400	-1,821	0,419	1,05
420	-1,819	0,283	1,12
500	-1,779	0,151	1,62
560	-1,821	0,419	1,05
800	-1,817	0,713	1,02
840	-1,820	0,509	1,04
1000	-1,743	0,118	2,78

1010	-1,472	0,260	1,15
1180	-1,708	0,174	1,42
1380	-1,821	0,713	1,02
2000	-1,821	0,713	1,02
3000	-1,878	0,260	1,15
4000	-1,821	0,366	1,07
10000	-1,821	0,509	1,04
12000	-1,821	0,713	1,02
60000	-1,821	0,509	1,04
1000000	-1,821	0,713	1,02

Appendix II

Table 6 – Regression equations for each value of gas price.

Gas price	Equation		
0	Y'	=	2,389 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
1	Y'	=	3,098 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
10	Y'	=	0,5975 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
20	Y'	=	0,6613 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
30	Y'	=	0,6652 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
40	Y'	=	0,3658 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
50	Y'	=	0,6769 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
60	Y'	=	0,5678 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
70	Y'	=	0,6596 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
100	Y'	=	1,010 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
130	Y'	=	0,6242 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
150	Y'	=	0,5725 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
200	Y'	=	0,5674 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
210	Y'	=	0,5266 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
220	Y'	=	0,6201 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
230	Y'	=	0,7950 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
250	Y'	=	0,5432 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
300	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
310	Y'	=	0,5705 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
360	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
400	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
420	Y'	=	0,5694 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
500	Y'	=	0,6093 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
560	Y'	=	0,5676 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
800	Y'	=	0,5718 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
840	Y'	=	0,5692 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
1000	Y'	=	0,6454 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
1010	Y'	=	0,9164 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
1180	Y'	=	0,6805 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
1380	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove

2000	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
3000	Y'	=	0,5109 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
4000	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
10000	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
12000	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove
60000	Y'	=	0,5680 + 0,001237 hashpower_accepting + 0,001358 tx_atabove

Appendix III

This appendix refers to all the source code developed to implement the proof of concept. Due to its size, it's not presented here, being only in a CD distributed with this document. It is divided into three components:

- Sensor-Client: the user interface –web page -, where it can be started the process of buying the data and where the result is returned. It was developed using HTML, CSS and JavaScript;
- Sensor-Services: the services to interact with the Ethereum network and for fetching the data from the sensor, developed with Java language – Spring Boot Framework. It also contains the code for the WebSocket server and the Storj API for uploading and downloading the query results, being both developed in Node.js – JavaScript.
- Sensor-Contract: The Smart Contract code, used to query the sensor through Oraclize API and to store the result file hash in the transaction log. It was developed with solidity language and truffle framework to manage the corresponding deploy into Ethereum Ropsten Testnet.

Attachments

Attachment I

Scripts executed on the Ethereum node, captured the relevant data for performing the statistical analysis - courtesy of Jake Elkins.

Two scripts were used:

- egs_ref.py script: data model for the captured information - https://github.com/ethgasstation/ethgasstation-adaptive-Oracle/blob/master/egs_ref.py
- ethgasstation.py script: data capture process, storage and reporting - https://github.com/ethgasstation/ethgasstation-adaptive-Oracle/blob/master/ethgasstation.py