Pontifícia Universidade Católica
do Rio de Janeiro

Adrian Concepción León

Secure distributed ledgers to support IoT technologies data

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Markus Endler

Rio de Janeiro

October 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Adrian Concepción León

Secure distributed ledgers to support IoT technologies data

Dissertation presented to the Programa de Pós-Graduação em Informática, of PUC-Rio, in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee.

Prof. Markus Endler
Advisor
Departamento de Informática - PUC-Rio

Profª. Noemi de la Rocque Rodriguez
Departamento de Informática - PUC-Rio

Prof. Sérgio Colcher
Departamento de Informática - PUC-Rio

Rio de Janeiro, October 18th, 2018

Adrian Concepción León

Computer Science, University of Havana 2013. Havana, Cuba. He is a researcher at Laboratory for Advance Collaboration – LAC-PUC-Rio at Pontifical Catholic University of Rio de Janeiro since 2016. His main studies are related to the area of distributed computing.

Bibliographic data

Concepción León, Adrian

Secure distributed ledgers to support loT technologies data / Adrian Concepción León ; advisor: Markus Endler. – 2018.
63 f. : il. color. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2018.
Inclui bibliografia

1. Informática – Teses. 2. Ledger distribuído. 3. Blockchain. 4. Internet das coisas. 5. IOTA. 6. Ethereum. I. Endler, Markus. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my best friend Raul, you´ve been my inspiration,
even now that you are not with us anymore

To that important part of my life: my wife; the one that
most encourages my madness. Eternally, I love you

To my family and my dreams…

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Markus Endler for the continuous support of my Master´s degree study; for his patience, motivation, and his knowledge.

I thank my fellow labmates at the Laboratory of Advanced Collaboration (LAC) and others friends at Pontificial Catholic University (PUC-Rio), especially those at Laboratory of Software Engineer (LES), with a special mention to Victor, Felipe, George and Marx.

I am also grateful to the Department of Informatics of the Pontificial Catholic University (PUC-Rio), for give me the opportunity of improve myself. Special thanks to the D.I. staff and the CAPES for funding my research.

To my family:

Thanks to my wife Claudia: for your support and for always getting the best out of me.

I am grateful to my sister Marilucita, my mother Mariluz, and my grandmother Caridad, who have given me moral and emotional support in my life, even when we are thousands of miles away. Special thanks to Adael.

I want to thanks to my father Rolando and my siblings, Michelle and Roly, for encouraging me to follow my dreams.

To my grandmother Nely *(in memoriam)*, who always waited for me….I miss you so much.

To my *abuela Cuca*. One hundred years of joy. *Te quiero.*

Thanks to my grandfather, for illuminating my childhood, and for my love for peanuts.

I am also grateful to my other family members and friends who have supported me along the way, especially my two aunts, Lidia and Alina, and my cousins Eivys, Alejandro and Ailin.

Thanks to my wife´s family: for those who became part of my own family, especially my *suegri* Maguie, Xiomara and Jesús´s family.

You are all always in my mind!

To Carlos Jiménez, thanks a lot for your motivation and for believing in me.

A special gratitude to my friends, those who has been a family here in Brazil and those who are far in the distance, but are close in heart.

Finally, I want to thanks all those who support me through all these years and those who help me to be here today, one way or another.

This study was financed in part by the Cordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil(CAPES)- Finance Code 001

Abstract

Concepción-León, Adrian.; Endler, Markus (Advisor). **Secure distributed ledgers to support IoT technologies data.** Rio de Janeiro, 2018. 63p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Blockchain and Tangle are data structures and protocols used to create an immutable public record of data ensured by a network of peer-to-peer participants that maintain a monotonic constantly growing set of data records known as ledgers. Both technologies provide a decentralized solution that guarantees the exchange, among billions of IoT devices, of large amounts of trusted messages, which are very valuable as long as they are valid and complete. This highly encrypted and secure peer-to-peer messaging mechanism is adopted in this project to manage the processing of IoT transactions. To maintain transactions private, and secured consensus algorithms are responsible for validating and choosing transactions and recording them in the global ledger. The results showed that the speed of the consensus algorithms can affect the creation in real-time of reliable stories that track the events of the IoT networks. After incorporating Complex Event Processing that allows selecting only those high-level events, it is possible to obtain an improvement in many situations. The result is a Middleware system that provides a framework for the construction of large-scale computer applications that use Complex Events Processing and different decentralized ledgers such as the blockchain of Ethereum or IOTA Tangle, for secure data storage.

Keywords

Distributed Ledger; Blockchain; Internet of Things; IOTA; Ethereum

Resumo

Concepción-León, Adrian.; Endler, Markus (Orientador). **Ledgers seguros e distribuídos para suportar dados de tecnologia IoT.** Rio de Janeiro, 2018. 63p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Blockchain e Tangle são estruturas de dados usadas para criar um registro público imutável de dados segurados por uma rede de participantes peer-to-peer que mantém um conjunto de registros de dados em constante crescimento conhecidos como ledgers. As tecnologias Blockchain e Tangle são uma solução descentralizada que garante a troca de grandes quantidades de mensagens confiáveis, entre bilhões de dispositivos de IoT conectados, os quais são muito valiosos, desde que sejam válidos e completos. Esse mecanismo de mensagens peer-to-peer altamente criptografado e seguro é adotado neste projeto para gerenciar o processamento de transações de IoT e a coordenação entre os dispositivos que interagem com o processo. Para manter as transações privadas e seguras, os algoritmos de consenso distribuídos são responsáveis por validar e escolher as transações e registrá-las no ledger global. Os resultados mostraram que a velocidade dos algoritmos de consenso pode afetar a criação em tempo real de histórias confiáveis que rastreiam os eventos das redes IoT. Após incorporar o Processamento de Eventos Complexos, que permite selecionar apenas os eventos de alto nível, é possível obter uma melhoria em muitas situações. O resultado é um sistema Middleware que fornece framework para a construção de aplicativos de larga escala onde podem usar Processamento de Eventos Complexos e diferentes ledgers descentralizados, como o blockchain da Ethereum ou IOTA Tangle, para armazenamento seguro de dados.

Palavras-chave

Ledger Distribuído; Blockchain; Internet das Coisas; IOTA; Ethereum

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CEP	Complex Event Processing
CQL	Continuous Query Language
DAG	Directed Acyclic Graph
DApps	Decentralized Applications
DQLs	Data Query Language
EDA	Event-Driven Architecture
EOAs	Externally Owned Accounts
EPA	Event Processing Agent
ETH	Ethereum
IoC	Investment of Control
IoMT	Internet of Mobile Things
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
MQTT	Message Queuing Telemetry Transport
OOP	Object-Oriented Programming
P2P	peer-to-peer
PoS	Proof-of-stake
PoW	Proof-of-work
REST	REpresentation State Transfer
RPL	Recursive Length Prefix

"The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so."

Kevin Ashton, 2009

1. Introduction

The “Internet of Things” (IoT) is related to the integration of the physical world with the virtual world of the Internet (Haller, 2010); and already has a strong impact in several areas such as smart homes and cities, environmental monitoring, asset monitoring, logistics, etc. Thanks to rapid advances in underlying technologies, IoT is opening tremendous opportunities for a large number of novel applications that promise to improve the efficiency of several economic sectors as well as the quality of our lives (Xia, 2012).

IoT advances are correlated with an increase in the collection and analysis of a large amount of data, by connecting multiple devices to each other and to the cloud; therefore, it is necessary to handle a large volume of information in real-timefrom a variety of sources and detect significant events in time by correlating this data. These data are valuable only maintaining their integrity and privacy and the security of the data affects the integrity of the information.

Currently, many IoT devices, data transmission protocols and databases are supported by security models susceptible to accidentals and malicious attacks, including data tampering (Razzaq *et al.*, 2017). The "distributed ledgers" offer an elegant solution to solve this problem, since its decentralized, autonomous and trustless capabilities make it an ideal component of IoT solutions (Buterin, 2015). The adoption of distributed ledgers with this important characteristic allows a secure peer-to-peer messaging mechanism to manage the processing of billions of IoT transactions and the coordination between the devices (Buterin, 2015). IoT companies have quickly become one of the first to adopt distributed ledgers technologies, although in addition to building a decentralized IoT, it is also necessary to create one that can scale while maintaining private transactions secure and trustless.

To ensure data storage using distributed ledgers, we use two different technologies: the Ethereum blockchain and the IOTA tangle. The Ethereum blockchain provides a persistent structure where data is stored in a reserved space in each transaction. Tangle is a blockless distributed ledger of the cryptocurrency platform called IOTA and was created as a cryptocurrency for the IoT industry

(Popov, The tangle, 2016). Despite these important characteristics, there are associated problems that must be resolved before a ledger based-storage system can be applied to IoT, such as: privacy of data and users, centralization of miners, vulnerability, the immutability of smart contracts, limits on data storage, consensus, andslow writing, among others

In this project, a filtering technique will be addressed to solve the problems related to the scalability and performance of the distributed ledgers, to be applied to an IoT system. In order to mitigate the problems previously mentioned, the network can be structured with an architecture where the computational intelligence is located closest to where the data is collected. Thus, instead of sending all the data collected by many endpoint devices, they will be analyzed locally before being sent to the ledger. To analyze, filter and collect data from the devices of an IoT System, the middleware ContextNet was used as a transport layer. In addition, it allows using techniques of Edge Computing such as CEP, and data buffer, for data filtering. By implementing these filtering techniques as part of this project, we intend to reduce the amount of data sent to the ledger locally to solve the growing problems of running out of physical bandwidth as the scale increases.

The aim of this project is the **implementation of a secure transaction storage system using distributed ledgers as a middleware service for the IoT Middleware ContextNet.**

Specific Aims:

- To solve problems of scalability and performance in the storage of IoT metadata in Tangle and Ethereum distributed ledgers through the use of Fog computing techniques.
- To implement the new DLedger service within ContextNet Middleware for access to the different distributed ledgers.
- To evaluate the performance of the new architecture implemented.

For these reasons, the implementation of a trustless peer-to-peer messaging protocol based on Distributed Ledgers for ContextNet is proposed to create a more secured transport protocol for IoT. This messaging protocol will be capable to scale while stored in a distributed way and maintain private, secure and trustless transactions. In the blockchains, all the transactions that are stored in the blocks are

validated, so the adoption of peer-to-peer computing such as the blockchain to process and store all these transactions in the IoT scale, can also facilitate and reduce the costs associated with the installation and maintenance of large centralized data centers (IBM, 2014). In addition, it is the first time that a middleware allows to select between two different distributed ledgers for the storage of IoT metadata. With this new service, the different functions of distributed ledgers can be combined or supplemented in order to increase their capabilities. The solution will be part of ContextNet as a new service for IoT gateways.

2. Fundamental Concepts and Technologies

2.1 Internet of Things

IoT represents a vision in which the Internet extends into de real world-embracing everyday objects (Mattern, 2010); to achieve that, machines and objects are connected via the Internet, and these devices are usually controlled and monitored remotely, generally wirelessly (Afshar, 2014). Internet of Things will increase the ubiquity of the Internet by integrating smart objects, which leads to a highly distributed network of devices communicating with human beings as well as other devices (Xia, 2012). These smart objects are often equipped with sensors, actuators and have some local processing capabilities. The mash-up of captured data from sensors with data retrieved from other sources, gives rise to new synergistic services which result in improved efficiency, accuracy and economic benefits that cannot be provided by an isolated embedded system (Kopetz, 2011).

According to Gérald Santucci, "the present *Internet of PCs* will move towards an *Internet of Things* in which 50 to 100 billion devices will be connected to the Internet by 2020" (Santucci, 2010). Another promising trend is the development of smart cities, since the IoT has the potential to transform entire cities by solving real problems citizens face each day. With the proper connections and data, it is possible to solve traffic congestion issues and reduce noise, crime, and pollution. In Barcelona, for example, several IoT initiatives have been implemented, helping to improve smart parking and the environment, and becoming a good example of how asmart city could be implemented. The automotive industry is also using of IoT withthe development of connected cars, where vehicles are equipped with Internet access and can share that access with other cars, creating a wireless network. AT&T, for example, added 1.3 million cars to its network in the second quarter of 2016, bringing the total number of cars connected to 9.5 million (Meola, 2016). The adoption of technological advances also improves medical care, where IoT plays an increasing role, as it does in specific applications.

2.1.1 Sensors used in IoT

IoT connects real-world objects to the Internet using various sensors and

therefore, the growing number of low-cost sensors available has been one of the main drivers of this technology. Some of the standard sensors include sound, light, electrical potential (through potentiometer), movement (by accelerometer), temperature and humidity. In addition, multiple sensors focused on the health area have emerged, such as those that record the electrical activity of the heart (ECG / EKG), measure the electrical activity of the muscles (EMG), read the electrical activity along the scalp (EEG) and measure the blood flow volume (PPG).

Another promising area is the development of positioning sensors used in the Internet of Things. Satellite navigation systems, such as GPS, GLONASS, or Galileo allow locating a person or a mobile device outdoors, but the situation is usually more complicated within buildings without a line of sight for a navigation system. In such cases, other solutions are employed, usually based on radio networks (e.g., IEEE 802.11-WiFi) and fingerprints of signal strengths of individual WiFi devices which transmit their signals inside a building (Brida *et al.*, 2011). Generally, there are already some WiFi access points in the building that more or less cover the entire construction with the radio signal that can be used for the location. However, localization accuracy is influenced by a number of elements, for example, by characteristics of transmitters and receivers and by characteristics of the environment that influences the radio signal propagation (Pavel *et al.*, 2016).

Bluetooth-based indoor localization has been widely used to overcome these problems, mainly with the arrival of Bluetooth 4.0 in 2010, including BLE (Bluetooth low energy) or Bluetooth Smart; which allow very low power devices to work for months or even years with small batteries such as coin cells. Due to the low energy consumption and configuration options (regarding the advertising interval and the transmitter output power), the utilization of this technology is much more promising, not only in comparison to previous versions of Bluetooth, but also in comparison with today's widespread WiFi-positioning. Comparisons between the BLE-based localization and the WiFi- location were performed by the deployment of BLE beacons in the same places where the WiFi access points were originally placed. The results showed that BLE is more accurate in identical places than WiFi (Zhao *et al.*, 2014).

Nowadays, the trend is the use of multisensor platforms that incorporate several sensor elements, such as the combination of an accelerometer, a GSR sensor, a temperature sensor and possibly a heart rate sensor; in order to develop a new generation of personalized automatic tracking products.

2.2 Distributed Ledgers

This technology allows transactions and data to be recorded, shared and synchronized in a distributed network of different network participants. A distributed ledger is a (growing) set of transactions (Popov, Local modifiers in the Tangle, 2018). Each distributed ledger system has a feature that allows storing all transactions and data immutably. Formally, a distributed ledger system is immutable because it fulfills that: at each times t> 0, there is a data set Ledger(t) thatdescribes the entire system history up to that point. In principle, it is stored in all the nodes of the network and is the same for all. This ledger is incremental, in the sense that Ledger (t) ⊂ Ledger (s) for all t <s; that is, the data is added but never deleted. For this reason, by choosing the storage of all sensor data in a distributed ledger network, as well as Ethereum or IOTA, it is guaranteed that these are immutable.

2.2.1 Blockchain technology

A blockchain is a data structure used to create a decentralized ledger (Narayan, 2017). It is an immutable public record of data secured by a network of peer-to-peer (P2P) participants that maintain a continuously growing set of data records called *blocks*. Each block is linked and identified by its cryptographic hash (Dorri, 2016), thus creating a chain of blocks, or blockchain (**Figure 1**).

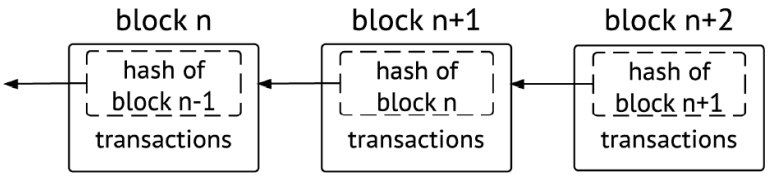

Figure 1. Graphic representation of a blockchain ledger *(Narayan, 2017)*.

Each blockchain address/account is represented by a pair of asymmetric cryptographic keys. Transactions between users or counterparties can only be made by the owner of the private key and are only accepted as valid if carry a valid signature for the public key. Each transaction is verified and recorded in a process called "*mining*" by the consensus of the majority of the participants in the system. Once the transaction entered the block, that information can never be erased or modified again (Christidis, 2016). After the assembly, each new block requires to be validated by applying an algorithm to the transaction to verify its validity in the P2P network and subsequently it is stored at the end of the chain, leaving a list of referenced blocks (Conoscenti, 2016) (**Figure 2**).

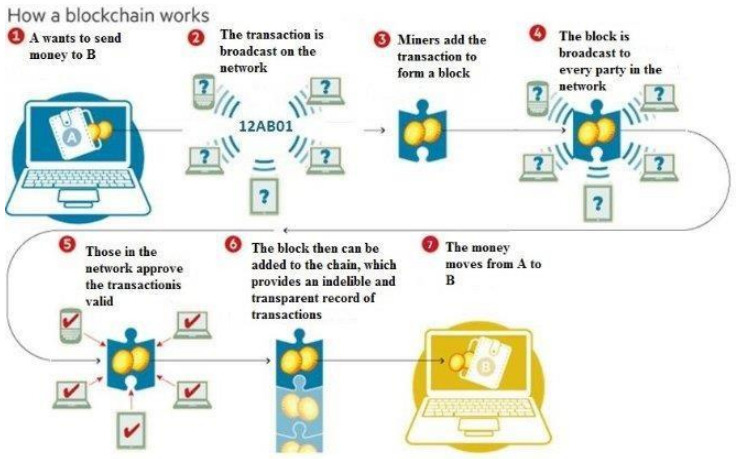

Figure 2. Graphical representation of the blockchain workflow *(Crosby, 2016)*.

2.2.1.1 Ethereum distributed storage

Ethereum (ETH) is a decentralized platform that executes smart contracts on a customized blockchain and allows the deployment of Decentralized Applications (DApps)[1] on top of it. Ethereum provides a decentralized peer-to-peer where participants can exchange trusted messages.

Decentralized blockchain applications have few options for storing data. In the Ethereum blockchain, the data could be stored in places reserved for the contract data. It is also possible to send someone a transaction and include some data in another place reserved for transaction input data.

There are two types of nodes in the Ethereum network: regular and mining nodes. The regular nodes are those that only have one copy of the blockchain, while the miners build the blockchain by mining blocks. In addition, Ethereum uses the Dagger mining algorithm, which consists of a hard memory work test based on moderately connected acyclic graphics (Wood, 2014) (Buterin, 2013). The rates of each transaction are paid using "Ether", the main internal crypto-fuel of Ethereum (Buterin, 2015).

In the Ethereum blockchain, the block is stored in a multi-level data structure named Merkle Patricia tree (trie) (**Figure 3**). The Merkle Patricia tree provides a persistent data structure for mapping between arbitrary-length binary data (byte arrays). It is defined in terms of mutable data structure to map between 256-bit binary fragments and arbitrary-length binary data, implemented as a database. When a block is composed, a function named RPL is used to code the structure. This RPL (Recursive Length Prefix) function is for encoding arbitrarily structuredbinary data (byte array).

[1] A complete decentralized application should consist of both low-level business-logic components, whether implemented entirely on a decentralized peer-to-peer network and high-level graphical user interface components written in any language that can make calls to its backend. No single node in the network has complete control over the DApp.

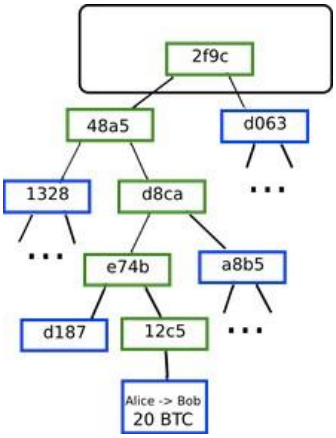

Figure 3. Representation of the multi-level data structure named Merkle Patricia tree (trie).

2.2.1.2 Tangle distributed storage

Instead of the global blockchain, Tangle is a directed acyclic graph (DAG) to store transactions. Tangle is blockless distributed ledger of the cryptocurrency platform called IOTA and was created as a cryptocurrency for the Internet of Things industry. In Tangle, all data is stored in distributed and trustless nodes among the network, and this design guarantees the integrity of the data that is sent. Tangle is scalable, lightweight, and allows transferring values without any fees; so that devices can trade exact quantities of resources on-demand, and store data from sensors and dataloggers securely and verified on the ledger (Popov, 2016). The nodes of the IOTA network are entities that issue and validate transactions. The IOTA network is asynchronous and may contain conflicting transactions; in this case, the nodes must reach a consensus and decide which transactions will be orphaned (Popov, 2016).

It is possible to transfer data without any payment by executing a transaction containing IOTA. Everyone in the connected cluster has a copy of the data. If there is an alteration of the original data, the rest of the nodes of the network could identify that there is incompatibility with their own copy (Serguei, Saa, & Finardi, 2018).

The data structure on IOTA is defined as a graph finite G = (V, E) where V is the set of vertices and E is the multiset of edges with the following properties:

- For u, v ∈ V, we say that u approves v, if (u, v) ∈ E.
- The state of the tangle at time t ≥ 0 is a DAG T (t) = (V_T (t), E_T(t)), where E_T(t) is the set of vertices and E_T (t) is the multiset of directed edges at time t. The process's dynamics are described in the following way:
- The tangle grows with time, that is, V_T (t_1) ⊂ V_T (t_2) and E_T (t_1) ⊂ E_T (t_2) whenever $0 \leq t_1 < t_2$.
- To issue a transaction, one node chooses two other transactions to approve according to an algorithm; therefore, users who issue a transaction contribute to the security of the network
- Each incoming transaction chooses two vertices v' and v'' (which, in general, may coincide), and we add the edges (v, v') and (v, v''). We say in this case that this new transaction was attached to v' and v'' (equivalently, v approves v' and v'').
- Specifically, if a new transaction v arrived at time t', then V_T (t'+) = V_T (t')∪{v}, and E_T(t'+) = E_T (t) ∪ {(v, v'),(v, v'')}

2.2.1.3 Blockchain transactions

A transaction is a signed data package that is sent by Externally Owned Accounts (EOAs) to other EOAs or contract accounts. They are the smallest building blocks of a blockchain system. Each transaction includes the recipient's address, the transaction data payload, and the transaction value. Transactions are broadcast to all participant nodes in the network. They are independently verified and "processed" by each node and included in the blocks in the mining process. Once transactions are buried under enough confirmations they can be considered irreversible (Madisetti, 2016).

2.2.1.3.1 Ethereum transaction

An Ethereum transaction (Wood G. , 2014) is formally defined as:

$$\sigma_{(t+1)} = \gamma(\sigma_t + T)$$

Where **γ** represents the Ethereum stat transaction and **σ** allows storing states between transactions.

In Ethereum, the transaction formally is a tuple denoted by the letter **T**. The fields of the transaction structure are described in Table I.

For an Ethereum transaction to be executed, it must pass the following validations:

1- The transaction is a well-formed Recursive Length Prefix (RPL), with no additional subsequent bytes.

2- The signature of the transaction is valid.

3- Nonce transaction is valid (Equivalent to the current nonce shipping account).

4- The gas limit is not less than the intrinsic gas g0, used by the transaction.

5- The balance of the sender's account contains at least the cost, v0 required in the initial payment.

Table I. Fields of the Ethereum transaction structure.

FIELD NAME	FORMALLY NAME AS	DEFINITION
Nonce	**Tn**	A scalar value equal to the number of transactions sent by the sender
GasPrice	**Tp**	A scalar value equal to the number of Wei[2] to be paid per Unit of gas
gasLimit	**Tg**	A scalar value equal to the maximum amount of gas that should be used in executing this transaction
to	**Tt**	The address of the message call recipient
value	**Tv**	A scalar value
v	**Tw**	The value corresponds to the signature of the transaction
r	**Tr**	The value corresponds to the signature of the transaction
s	**Ts**	The value corresponds to the signature of the transaction
init	**Ti**	Unlimited byte size array
data	**Td**	Unlimited size array

** All components are integer values except for Ti and Td, where Ti* $\in \mathbb{B}$ *y Td* $\in \mathbb{B}$.

[2] Wei is the smallest subdenomination of Ether, and thus the one in which all integer values of the currency are counted.

2.2.1.3.2 IOTA transaction

A transaction in IOTA consists of 2673 trytes (if encoded). When you decode the trytes you get a transaction object which has the following values:

Table II. Fields of the IOTA transaction structure.

FIELD NAME	DEFINITION	FORMALLY NAME AS
hash	String 81-trytes unique hash of this transaction.	**TIh**
signatureMessageFragment	String 2187-trytes signature message fragment. In case there is a spent input, the signature of the private key is stored here. If no signature is required, it is empty (all 9's) and can be used for storing the message value when making a transfer. More to that later.	**TIs**
address	String 81-trytes address. In case this is an *output*, then this is the address of the recipient. In case it is an *input*, then it is the address of the input which is used to send the tokens from (i.e. address generated from the private key).	**TIa**
value	Int value transferred in this transaction.	**TIv**
timestamp	Int timestamp of the transaction. It is important to know that timestamps in IOTA are not enforced.	**TIt**
currentIndex	Int the index of this transaction in the bundle.	**TIc**
lastIndex	Int the total number of transactions in this bundle.	**TIl**

FIELD NAME	DEFINITION	FORMALLY NAME AS
bundle	String81-tryte bundle hash, which is used for grouping transactions of the bundle together. With the bundle hash you can identify transactions that were in the same bundle.	**TIb**
trunkTransaction	String 81-trytes hash of the first transaction that was approved with this transaction.	**TItt**
branchTransaction	String 81-trytes hash of the second transaction that was approved with this transaction.	**TIbt**
nonce	String 81-trytes hash. The nonce is required for the transaction to be accepted by the network. It is generated by doing Proof of Work (either in IRI via the attachToTangle API call, or with one of the libraries such as ccurl).	**TIn**

The process in IOTA looks as follows:

1. **Signing**: Sign the transaction inputs with your private keys, which can be done offline.

2. **Tip Selection**: MCMC is used to randomly select two tips, which will be referenced by your transaction (branchTransaction and trunkTransaction).

3. **Proof of Work**: In order for your transaction to be accepted by the network, you need to do some Proof of Work - similar to Hashcash[3], not Bitcoin (spam and sybil-resistance). This usually takes a few minutes on a modern PC. Each transaction within IOTA has an empty field called "signatureMessageFragment" that can be used to store data when a transfer is made (IOTA Guide, 2018).

[3] A hash function is any function that can be used to map data of arbitrary size to data of a fuxed sized. The values returned by a hash function are called hash values, hash codes, digests, or simple hashes.

2.2.1.4 Reaching Consensus on the Network

In the network, each computer holds a copy of the blockchain, and a group of validators ("miners" are the nodes in the network that mine blocks), who digitally sign the blocks they create, are responsible for choosing the transactions and recording them in the global ledger (Greenspan, 2015). The nodes of the network must agree on the transactions with a distributed consensus scheme to prevent a minority of validators from taking control of the chain. Otherwise, individual copies of the blockchain might diverge and will end up having forks (Christidis, 2016).

The distributed consensus consists of a competition where each miner competes to solve a difficult mathematical problem (puzzle) based on a cryptographic hash algorithm, while others verify that the solution to that puzzle is correct. The first miner to solve the problem gets a reward and the right to register the new block in the blockchain. The puzzle is basically in finding those nonce values that will be used in the creation of the block (Crain, 2017). Each blockchain technology has its own puzzle for the miners to solve. Proof-of-work (PoW) is the puzzle used in the bitcoin blockchain but there are other alternatives like Proof-of-stake (PoS). PoW can use hashing algorithms such as SHA-256, Blake-256 and scrypt (Percival, 2009) (used in Litecoin). Myriad, for example, is a mechanism that combines several of these algorithms together.

2.3 Middleware

A Middleware is software that works as an intermediate layer between applications, and it abstracts the complexity and heterogeneity of the underlying distributed environment with its multitude of network technologies, machine architectures, operating systems, and programming languages. Fundamentally, it allows communication and data management for distributed applications. According to Coulson G. *et al.* (2002), middlewares facilitate the tasks of designing, programming, and managing distributed applications by providing a simple, integrated and consistent distributed programming environment (Coulson, 2002).

2.3.1 ContextNet Middleware

ContextNet is a scalable “Internet of Mobile Things” (IoMT) middleware developed by the Laboratory of Advanced Collaboration (LAC) of Pontificial Catholic University (PUC-Rio. Rio de Janeiro, Brazil). The project ContextNet provides context services for wide- and large-scale pervasive collaborative applications such as online monitoring or coordination of mobile entities' activities. These entities may be users of portable devices, such as smartphones, vehicles, or autonomic mobile robots. In the ContextNet project, communication and context distribution capabilities are implemented in the Scalable Data Distribution Layer (SDDL), while other services and extensions are built as software modules on top of this distribution layer. Together, these software modules form the ContextNet middleware (Laboratory for Advanced Collaboration (LAC), 2017).

2.3.1.1 Mobile Hub

The Mobile Hub (M-Hub) is a general-purpose gateway that manages and connects smart and also mobile objects with the ContextNet services executing in the cloud through different WPAN technologies such as BLE and Classic Bluetooth. The main function of the M-Hub serves as an intermediary between the mobile objects and the services in the SDDL core of the ContextNet architecture. This middleware is responsible for discovering and connecting the smart objects to the SDDL Core (Talavera, 2016).

2.4 Complex Event Processing

Complex Event Processing is responsible to recognize relevant events from multiple sources that later will be processed and directed to smart objects with actuators. The goal of complex event processing is to identify events and respond to them as quickly as possible.

The Event Processor Language is a data query language (DQLs) with SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY clauses. Streams replace tables as the source of data with events replacing rows as the basic unit of data. Since events are composed of data, the SQL concepts of

correlation through joins, filtering through sub-queries, and aggregation through grouping may be effectively leveraged. These events are represented as POJOs (Plain Old Java Object) following the JavaBeans conventions (EsperTech, 2018). Event properties are exposed through getter methods on the POJO and the results from EPL (Event Processing Language) statement execution are also returned as POJOs. The EPL processing model is continuous: results are output as soon as incoming events are received that meet the constraints of the statement. Two types of events are generated during output: insert events for new events entering the output window and remove events for old events exiting the output window. Listeners may be attached and notified when either or both types of events occur (**Figure 4**).

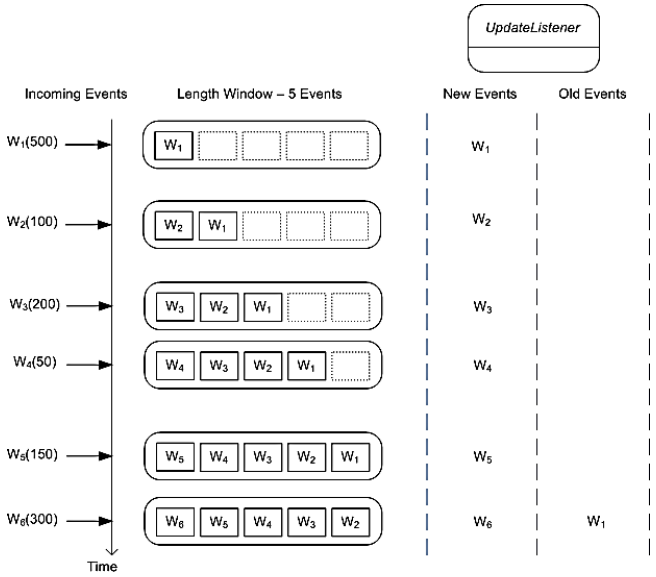

Figure 4. Complex Event Processing in ContextNet *(Oracle, 2018)*.

Incoming events may be processed through either sliding or batched windows. Sliding windows process events by gradually moving the window over the data in single increments, while batched windows process the events by moving the window over the data in discrete chunks. The window size may be defined by the maximum number of events contained or by the maximum amount of time to keep an event.

3. Implementation

3.1 Scalability challenge.

The architecture proposed by this project provides support for the storage of data in distributed ledgers. This ensures the storage of the metadata of all the actions of the IoT network, in an immutable and easily traceable way. The actions are related to the events generated by the network.

The first solution to address the problem is to execute a single transaction for each event that is issued from each sensor. As shown in **Figure 5**, the sensor emits an action that is received by the M-Hub to which it is connected. The M-Hub redirects the transaction to the ledger where the metadata will be stored. Once the ContextNet middleware receives the metadata, these are encrypted for later storage in the selected Ledger.

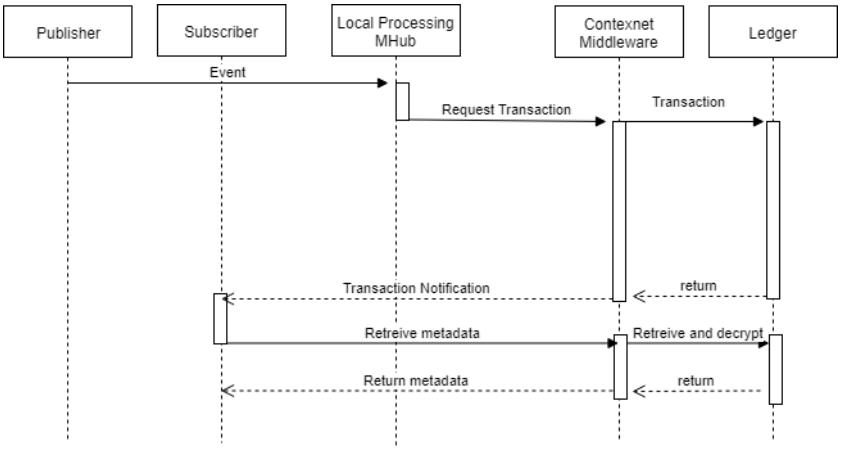

Figure 5. Sequence diagram for transactions to the distributed ledger.

This metadata related to the system individually does not require much space, but if we reach a large volume of transactions with scales similar to the dimensions of a real IoT system, this becomes a problem. The structure of the Ethereum blockchain affects the scalability and performance as a consequence of the consensus and slows writing, which is an inconvenience to be adopted by the IoT systems.

In this chapter, some Fog computing techniques were developed to face this problem of scalability and to evaluate the behavior of these techniques with the Tangle of IOTA, whose structure aims to offer a solution to the problems of scalability by facilitating the parallelization in the validation of transactions.

3.1.1 Ethereum Blockchain as a Metadata Store

The first solution to store IoT data in the ledger is that all the sensors make their own transaction to the Ethereum network. In this way, only the metadata related tothis sensor can be registered at that moment in each transaction.

Let us define formally the set of transactions that are expected to be inserted into the blockchain in a pre-established period of time is defined as:

$$U = \{T_1, T_2, T_3, ..., T_k\}.$$

3.1.1.1 Data Space

To calculate the space occupied by all the metadata of the transactions of this set, the RPL function is used according to the official yellow paper of the Ethereum network (Woo14). With this function, the metadata that will be inserted in the transaction is serialized. In this way, the total space occupied by the set U is:

$$S = \sum_{T \in U} RPL(T_d)$$

3.1.1.2 Compute the cost

To calculate the cost, we will first define the function gas G_0 to calculate the amount of gas[4] required to execute one transaction, defined as follows:

$$G_0 = \sum_{i \in T_i, T_d} \begin{cases} G_{txdatazero} & if\ i = 0 \\ G_{txdatanonzero} & otherwise \end{cases}$$

$$+ \begin{cases} G_{txcreate} & if\ T_t = \emptyset \wedge H_i \geq N_H \\ 0 & otherwise \end{cases} + G_{transaction}$$

where T_i and T_d are the fields of the transaction that we had previously defined that allow an unlimited array of Bytes. G is a scalar that corresponds to the relative costs, in the gas of the operations in the Blockchain. In the previously definedformula, the next variables intervene:

Gtxdatanonzero: corresponds to 68 gas units that must be pay by every non-zero byte of data or code for the transaction.

Gtxcreate: 32000 units of gas that must be paid for all contract-creating transactions after the Homestead transaction.

Gtxdatazero: 4 gas units to Pay for every zero byte of data or code for a transaction.

Gtransaction: 21000 units of gas that must be paid for each transaction.

With this function we can calculate the gas spent to execute all the transactions of the set U:

$$G_R = \sum_{T \in U} \sum_{i \in T_i, T_d} \begin{cases} G_{txdatazero} & if\ i = 0 \\ G_{t\ \ datanonzero} & otherwise \end{cases}$$

$$+ \sum_{T \in U} \begin{cases} G_{txcreate} & if\ T_t = \emptyset \wedge H_i \geq N_H \\ 0 & otherwise \end{cases} + \sum_{T \in U} G_{transaction}$$

3.1.1.3 Data buffer with Ethereum transactions

Because the speed at which transactions are stored in the blockchain limits the performance of a network of IoT devices, a more efficient technique was developed, where the number of transactions would be reduced by using a data buffer. The fields Ti and Td, responsible for the storage of data of an Ethereum transaction, have unlimited space; so is possible to temporarily accumulate locally all the metadata of a set of devices.

Periodically, these accumulated data will be sent to the Ethereum network forming a single transaction. In this way, the number of transactions would be reduced and would also allow gas savings, since the total amount of gas that would be spent for all transactions of the set U defined above will be reduced.

Before defining the accumulation function, we must define the concatenation function:

$$A \cdot B = \{a \cdot b \mid a \in A \wedge b \in B\}$$

Since the concatenation is associative, it is not necessary to worry about the occurrence of asynchronous events.

The concatenation is associative: (AB) C = A (BC)

This, there are no conflicts with the insertion of events at different times, besides that each event can be associated with metadata that indicates the exact moment in which it was executed.

Now we can define the accumulation function that will allow reducing the costs to execute the transactions of the set U. This accumulation function represents the concatenation of the individual action metadata of the set U.

$$A(T', T'') = \{ A \mid A_i = T'_i \cdot T''_i \wedge A_d = T'_d \cdot T''_d \}$$

The Ethereum network has a condition in which the sum of the transactions gas of the block (Tg), must not be greater than the GasLimit of the block. For this reason, it is necessary to calculate the gas limit of the transaction and ensure it does not exceed the gas limit of the block. If it exceeds the gas limit, then we define a new set U' with the same initial characteristics.

For the gas calculation, the number of bits corresponding to the concatenation of all the metadata of the U set is first calculated. That is, A (U) = RLT (A (U)). Where RLT is the function of the Ethereum blockchain responsible for the conversion of byte to Hex.

To simplify the demonstration, we assumed that the set U does not exceed the gas that would be spent in a block. Then, let A be the result transaction corresponding to the accumulation of all the metadata of the set U. The function G_A was used to represent the total amount of gas that would be spent in all the transactions of the set U.

$$G_A = \sum_{i \in A_i, A_d} \begin{cases} G_{txdatazero} & if\ i = 0 \\ G_{txdatanonzero} & otherwise \end{cases} + \begin{cases} G_{txcreate} & if\ A_t = \emptyset \wedge H_i \geq N_H \\ 0 & otherwise \end{cases} + G_{transaction}$$

In this formula A_i or A_d represent the concatenation of the fields T_i or T_d $\forall\ T \in$ U, respectively. That is to say:

$$\sum_{i \in A_i, A_d} \begin{cases} atazero & if\ i = 0 \\ G_{txdatanonzero} & otherwise \end{cases} = \sum_{T \in U} \sum_{i \in T_i, T_d} \begin{cases} atazero & if\ i = 0 \\ G_{txdatanonzero} & otherwise \end{cases}$$

The reduction function will be named with the letter R and is responsible for combining the data related to several events in a single transaction. What was previously sent in different transactions, now with the function R, reduces the number of transactions by unifying the data.

$$G_A = \sum_{T \in U} \sum_{i \in T_i, T_d} \begin{cases} G_{txdatazero} & if\ i = 0 \\ G_{txdatanonzero} & otherwise \end{cases} + \begin{cases} G_{txcreate} & if\ T_t = \emptyset \wedge H_i \geq N_H \\ 0 & otherwise \end{cases} + G_{transaction}$$

Since Gtxdatanonzero, Gtxcreate, Gtxdatazero, Gtransaction, are always numbers greater than 0, the G_A series is bounded superiorly by the G_R serie.

$$G_R > G_A$$

3.1.1.4 Expected time

The Ethereum network only allows mining one block at a time. No matter how many nodes are trying to solve the consensus algorithm, the block is mined by the first one that solves the algorithm. In addition, this algorithm becomes increasingly difficult to solve as the number of blocks grows up to the genesis node.

In summary, the blockchain of Ethereum allows to obtain immutability during the storage of data, but its use in an IoT system would be limited due to its reduced

scalability. Previously, Wood, G. (2018) described the problems of ETH scaling, due to the difficulty in partitioning and parallelizing transactions (Wood, 2014).

3.1.2 Tangle of IOTA as a metadata Store

The Tangle network is asynchronous and aims to offer a solution to the problems of scalability and facilitates the parallelization in the validation of transactions. The Tangle is a distributed ledger whose main structure is a tree instead of a blockchain. Its long-term goal is to enhance microtransactions between IoT devices. The network is composed of nodes, that is, the nodes are entities that issue and validate transactions. Nodes do not have to achieve consensus on which valid transactions have the right to be in the ledger, which means that all of them can be in the tangle. Transactions issued by nodes constitute the site set of the tangle graph, which is the ledger for storing transactions.

3.1.2.1 Expected time

Transactions are issued by a large number of independent entities, so the process of incoming transactions can be modeled by a Poisson point process. Let **λ** be the rate of that Poisson process. For simplicity, let us suppose that this rate remains constant.

The previous property of the network indicates that in order to calculate the time it takes the Tangle network to validate a transaction, the number and computing power of the devices that are connected and mining must be taken into account. In this case, it is true that the time to validate a transaction is inversely proportional to the number of devices that are in the mining network.

Let also suppose that all devices have approximately the same computing power and that **h** is the average time a device needs to perform calculations required to issue a transaction. The transactions attached to the tangle at time **t**, only become visible to the network at time **t+h**. Thus, the expected time (Popov, The tangle, 2016) for a transaction to be approved for the first time is approximately:

$$L_0/(2\,\lambda) = 1.45\ h\ O(h)$$

On the contrary, in the case that a transaction is waiting for approval in a time interval much greater than $\mathbf{L_0 / (2 \lambda)}$, a good strategy would be to promote this latent transaction with an empty additional transaction.

3.1.2.2 Data buffering for Tangle.

The data buffering technique for the Tangle network is the same proposal that was made for the Ethereum network. Over a period of time, all sensor metadata will accumulate and be sent in a single transaction to the Tangle network.

To formalize the process of data buffering for Tangle, the set of transactions will be defined in the same way as before:

$$U = \{T1, T2, T3, \ldots Tn\}$$

Tangle transactions have limited space in the data field. The metadata sent by the sensors is stored in the "signatureMessageFragment" field, which has a limit of 2187 trytes. Therefore, each transaction must satisfy that value of Ti <2187 trytes. The accumulation technique for Tangle during the period of time, cannot exceed the transaction data limit.

Taking into account that there is a limit of data that can be sent to the message field of the transaction and, depending on the amount of metadata that this node accumulates, it will be necessary to share the load in different nodes to avoid bottlenecks.

3.1.2.3 Compute costs

The Tangle network fee to perform a transaction can be 0, since each node is responsible for mining its own transactions. The cost for the mining nodes would correspond to the energy it consumes to carry out the POW algorithm.

The fee of each transaction is not associated with the network fee, but with the amount of energy that the node itself spends to approve the transactions. In this way, it is possible to consider that the fee of the network remains at 0, while there is an increase in the expenditure of energy on the part of the sensor network.

In summary, for the 2 ledgers, it is possible to apply the accumulation technique that allows reducing the number of transactions executed in the network. For each network, the state diagram would be exactly the same as the one in **Figure 6**.

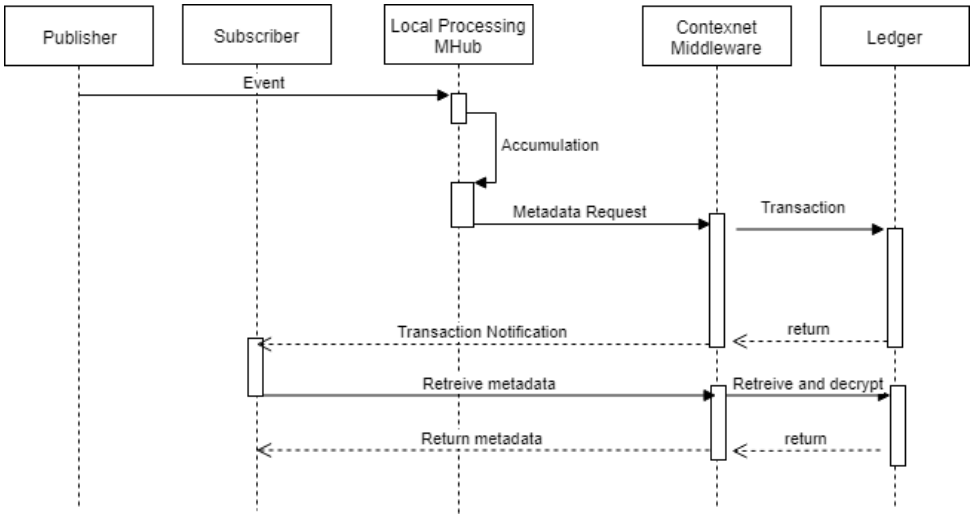

Figure 6. Sequence diagram representing the process of insertion in the leger with data buffering.

In addition to the accumulation technique, it is possible to optimize the solution using CEP in FOG computing, reducing the amount of necessary metadata information.

3.1.3 Use of CEP rules

The CEP event detection module is part of the M-Hub and has an Event-Driven Architecture (EDA). Following this architecture, events are analyzed in real ime to find situations of interest. The higher-level events that pass the CEP filter are mapped to Java objects using Esper (EsperTech, 2018).

For the processing of these events in real-time with CEP, a query is made in the declarative language CQL (Continuous Query Language) Language, where some criteria are used to select only those events that fulfill a certain pattern. The next code fragment shows an example of how they can be detected, taking as a case of using the detection warning of a bad location.

```
SELECT *

FROM Sensor.win:position(5 min)

WHERE Sensor(id) > 110
```

These CEP queries are executed by an Event Processing Agent (EPA), where this EPA continuously reacts to incoming events, producing exit events. These exit events can be consumed by any client that subscribes to this EPA. The way to subscribe to these EPA is through the use of the connectivity protocol MQTT (Message Queuing Telemetry Transport) which is based on the exchange of messages through a publish-subscribe pattern.

The processes to detect higher-level events that come from the M objects and to notify the clients are shown in the following diagram (**Figure 7**). The MHubs are responsible for the incorporation of this event detection component.

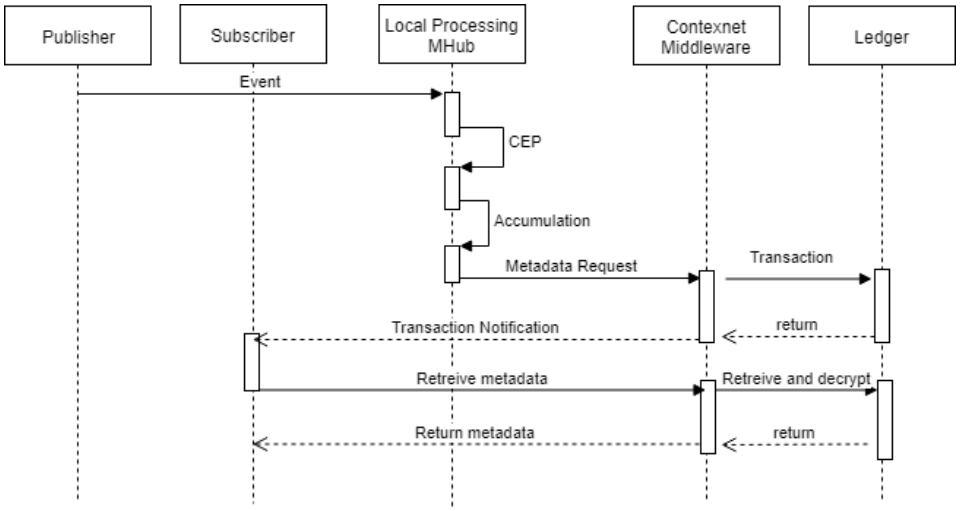

Figure 7. Sequence diagram representing the process of insertion in the leger with CEP and data buffering.

3.2 Developing a Client

The Client is responsible for the communication with the SDDL and the EPA, which is done through the MQTT publish-subscribe protocol. All the events that are filtered in the EPA are consumed by the client and sent in the form of a request

to the SDDL. The Client can be used for connecting to SDDL of ContextNet and sending requests.

Depending on the final destination of the data, the client instances will be created. In each case, the instance will be carried out with different parameters, applying the principle of ""Investment of Control "(IoC). The parameters will be injected into each instance through the constructor, and will depend on the type of API within the SDDL to which reference will be made. The possible APIs would be responsible for the functionalities of Tangle IOTA and for the functionalities of the Ethereum blockchain.

The declaration and implementation to configure a singleton of ILedgerAPI on the client side with dependency injection are shown below. In the Application Level you can create an instance of the client object and fill indirectly with the constructor for its configuration.

```
     ILedger_APPI ethereumClient = new Ledger_API(new EthereumAPI(“local
host”,”443”));

     //configure client

     ethereumClient.ConnectToNetwork();
```

On the client side, you do the same through IOTA:

```
     ILedger_APPI iotaClient = new Ledger_API(new IOTAAPI(“localhost”,”4
43”));

     //configure client

     iotaClient.ConnectToNetwork();
```

These clients, in addition to being responsible for communication by the communication SDDL with the M-HUB can consume the services offered by each of the APIs. The following table lists each of the services:

Table III. DLedger service functions and description.

FUNCTION	DESCRIPTION
GetBalance	Return the confirmed balance of an address
NewAddress	Return a new unique address
ConnectToNetwork	Connect the client instance with the ledger network
SendTransfer	Attach the specific transaction to the distributed ledger.
FindTransfer	Find a transaction that matches the specific input hash
SendToBufferTransfer	Accumulate transactions in a buffer and send a single transaction

3.4 Architecture Overview

The DLedger service implemented is part of the ContextNet middleware, so the final result is a distributed system for high-scale computing applications with a multilayer architecture that can be physically separated by 3 tiers: application tiers, middleware tiers and ledger tiers (**Figure 8**). This three-tier architecture allows the project to be easy to extend and scale, with distributed and decentralized programming as a basic principle. Each of the tiers has associated logical layers where the corresponding technologies and patterns are used for each service.

Communication between levels is done through the REST protocol (REpresentation State Transfer) using APIs to simplify communication. There is only one communication between consecutive tiers, so the Application tier does not directly access the third level of data.

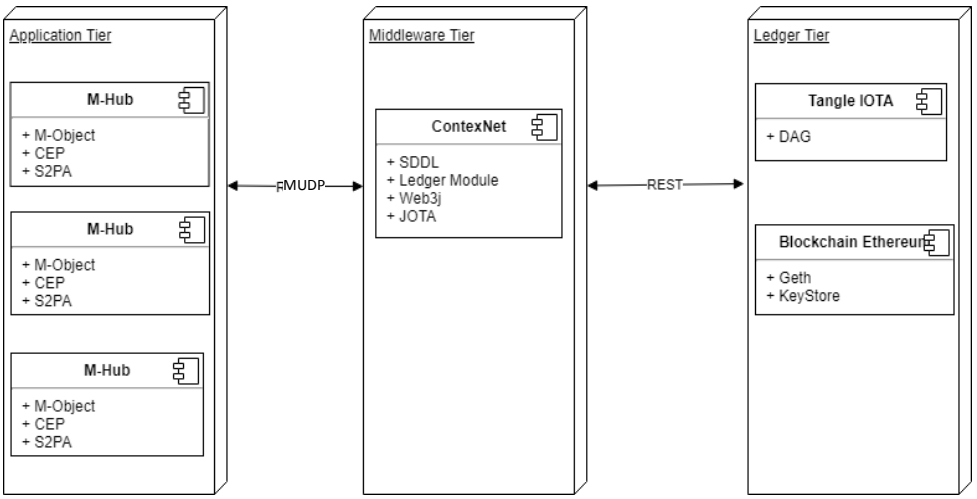

Figure 8. Diagram representing the three-tiers architecture of the system.

3.4.1 Application Tier

The Application Tier consists of one or more instances of applications that consume specific services of ContextNet. For example, this layer could contain applications that run in a smart car or can be a data marketplace. Applications can be connected to ContextNet through a REST service of the M-Hubs.

Each application is divided into multiple layers, as shown in **Figure 9**. The connection with the SDDL is through the M-Hub. The M-Hub also is capable of handling multiple M-OBJs using different WPAN technologies (for example, BLE, Classic Bluetooth).

All that data flow of the events, that are the product of the M-Object, is processed using CEP. Only those events that pass the CEP element are routed to the ContextNet SDDL through the CDDL, where they are processed as a transaction and stored in the ledgers. The application can also use the API to create new addresses associated with M-Objects.

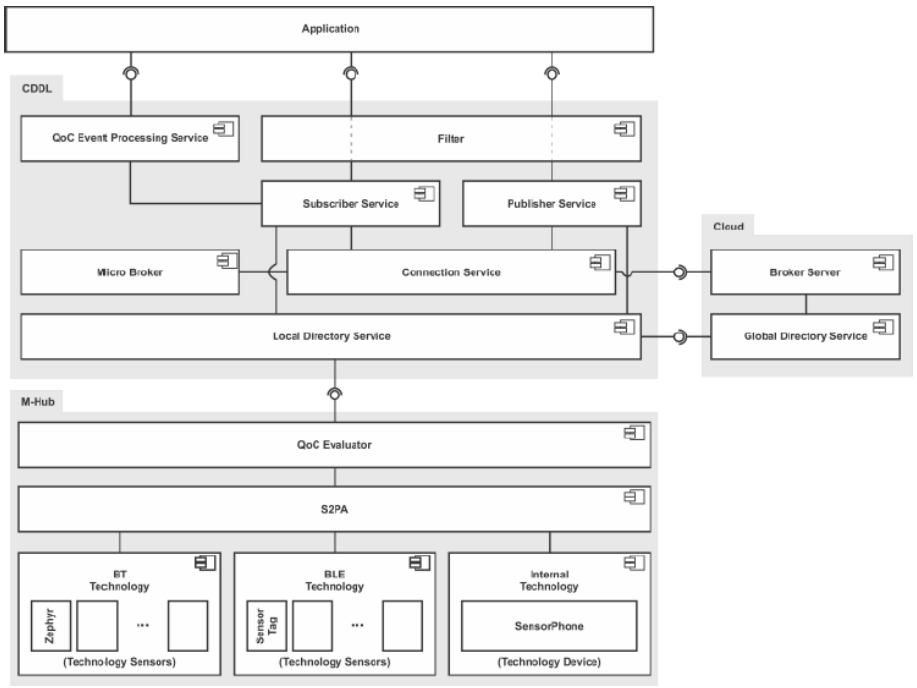

Figure 9. M-Hub/CDDL Middleware architecture (Oliveira-Carvalho, 2017).

3.4.2 Middleware Tier

A new module was implemented within the SDDL of the ContextNet that is responsible for providing a storage service in a secure and distributed way through the ledgers. Therefore, all interactions between any mobile node (M-Hub) or stationary node may be subject to a secure and immutable chain of blocks. This service is the responsible and the intermediary of communication with two ledgers technologies: Tangle and Ethereum; and is part of the Middle tier and is encapsulated in the SDDL Core in Java language.

IoT systems, machines and devices will perform data transactions in real-timeand the new service is responsible for validating these transactions and storing themin a distributed ledger. It is also presented at each of the gateways within the SDDLcore network and it will improve the middleware with new functions related to security and storage, allowing adaptation to environments where distributed applications require blockchain technologies (**Figure 8**).

DLedger service conforms a REST architectural style that can be consumed using clients that create instances of some Java object. The service is available as an API and allows you to control all the functions of each ledger. For the

implementation of the API, the Tangle and Ethereum technologies were consumed through their libraries in the Java versions: Web3j for Ethereum and IOTA Java for Iota. Web3j is a simple library of Java and Android to work with smart contracts and to integrate with clients (nodes) in the Ethereum network with applications. On the other hand, IOTA Java is the official Java library for IOTA.

This API can be represented by the following class diagram (**Figure 10**), which shows the use of dependency injection and the relation of each class with the common interface from which they inherit.

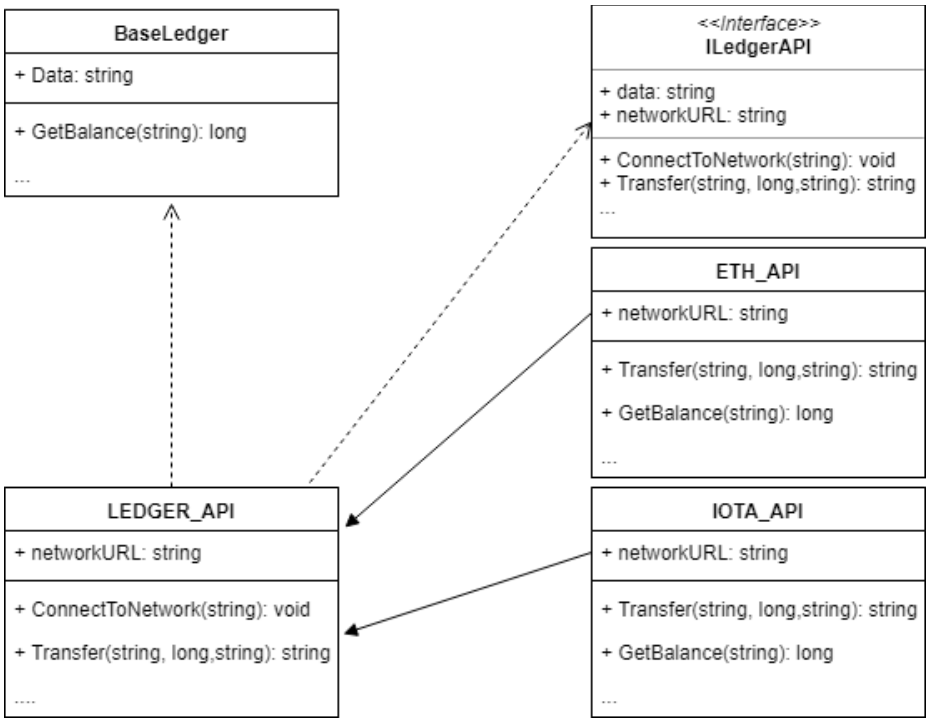

Figure 10. Class diagram for the ContextNet Ledger Module.

3.4.3 Ledger Tier

Ledger Tier is in charge of storing all the data that the sensors of the system emit. The way to store them is through transactions made in the network. Each transaction is an object that has a field where any type of data can be stored. This is the most basic layer of the proposed system and contains two different ledgers technologies, but both allow the storage of data immutably and safely. Applications that use the ContextNet middleware as a data access layer can choose which of the two technologies they prefer to store the data.

This layer has a local copy of the blockchain and Tangle. Each of the nodes that participate in the network, has a replica of this litter, providing the property to store the data in the network in an immutable manner. Each of these technologies has a different structure to store the data. In the case of the IOTA network, the structure is a DAG known as Tangle, as previously mentioned. Each of the transactions is represented as a vertex of the graph and has the information that will be stored. This information is stored in the "signatureMessageFragment" field, which has a limit of2187-trytes.

The structure of the Ethereum network is a blockchain. As previously mentioned, each block stores a set of transactions, and in each of these transactionsthe data is stored. Each transaction has a corresponding gas value and this is proportional to the amount of data stored. Although there is no limit to the information field in an Ethereum transaction, each block has a gas limit that depends on the gas of all its transactions. Therefore, in order to store data in Ethereum transactions, it is necessary to take into account the amount of gas limit in the block.

The access to this layer is through the ContextNet middleware using REST protocol. This layer simplifies the access to the data stored Layer. In the tier ledger, the data is never eliminated, so the only operations are to add new data or retrieve it through queries.

4. Performance Experiments and Results

To evaluate the system, an application that simulates the detection of people in different places within a building has been created. To achieve that, a total of 157 sensors that sent data for the IOTA and Ethereum network were simulated, and they can identify 150 possible people. Each people ("user") were detected inside the room using stations with MHubs installed. The station proceeds to locate the person who is present in a short-range radio, using Bluetooth action to obtain better accuracy. User-related data and the sensor metadata were stored in the general ledger distributed through a WiFi infrastructure.

To this end, a set of 157 stations distributed in different rooms were simulated; and they can identify 150 possible people. The metadata of the sensors (MObjects) was sent as a triple, as shown in **Figure 11**. A sample of this metadata might look like:

AA)

```
{1538836924675;66;9CSLAZMI9KMVRVICV9QHDQW9GWTCQDMOKMWD
ATQJQRCAVNGJUULDSILY9DV9GTCOCEGQYVGCSYDHDEDPD}
```

BB)

```
ODVAZAXABBBBXA9BCBWAYA9BABZAEB9B9BEBCBMBBCVBKBICWBSBCB
UBWBECACECSBMBECCB9CRBNB9CFCCBQBFCCCMB9CNBWBYBUBWBFCNBKBCC9
CTB9CACMBKBECXBQBTBDCDCVBNBBCSBVBHCCBNBECCBQBCCMBYBMBOBQB9C
HCECQBMBBCHCNBRBNBOBNBZBNBQD
```

Figure 11. Standard transaction of Ethereum **(A)** and IOTA **(B).**

The context data of the sensors (MObjects) were sent as a triple, and contains data related to the moment in which the event was executed (first element) in milliseconds, a specific value related to the identifier of the person who was detected, and the address within the network. The triple is stored in the IOTA and Ethereum transactions as previously described. The tests of both networks were carried out in environments with the same conditions.

To evaluate the system, three **(**3**)** possible environments or system configurations were analyzed. In the first configuration (Configuration 1), simple transactions were sent without using the accumulation technique or the CEP service. In this way, each sensor sent directly through ContextNet a transaction with the metadata corresponding to the event that was made. In the second configuration (Configuration 2), the data accumulation technique was taken into account. A set of metadata of the events are concatenated and sent forming a single transaction. After being sent, the process is repeated. This technique allows the accumulation of approximately 12 transactions without exceeding the block limit, being very important not to exceed this block limit. Finally, the third configuration (Configuration 3) also used the accumulation technique and CEP in the FOG computing, where a rule to filter the data has been established. This rule consists of detecting only people who have identifiers above 110. This system has a special interest in this group of people, and any extra data that is obtained will simply be discarded.

The following code fragment shows the rule that was applied with the use of CEP rules:

```
SELECT *
FROM Sensor.win:position(5 min)
WHERE Sensor(id) > 110
```

Each sensor has a unique address within the network that identifies it. To test Tangle and the Ethereum blockchain, a single-node instance [5] was used in each case, which acts as developer sandbox[6].

[5] An instance, in object-oriented programming (OOP), is a specific realization of any object. Each time a program runs, it is an instance of that program.

[6] A **sandbox** is a testing environment that isolates untested code changes and outright experimentation from the production environment or repository, in the context of software development.

An analysis of each configuration in both networks (IOTA and Ethereum) was performed to evaluate different parameters, such as the number of events or actions that each sensor issued, which could end in a transaction or not, and the number of transactions per sensor. The average of the time it takes to process each transaction per sensor (in milliseconds) was also analyzed. In addition, the number of pending transactions per instance of time was taking into account. The final parameter is the Fee Cost, which is the total fee that must be paid to the network so that the miners insert the transaction in the corresponding ledger.

For each configuration, the experiments were repeated three (3) times to make a comparison of the results. The instances of time analyzed in each configuration are defined as the exact moment (in milliseconds) in which a transaction was issued. A transaction is considered pending when it was started but has not yet been completed and inserted into the ledger. In the Ethereum network, when measurements of gas prices crossed the gasLimit threshold, they are no longer selected. When this happens, the previous transactions are accumulated and a single transaction with the data previously obtained; and a new data accumulation period immediately begins.

To test which environment performs best, the system was tested in three different distributed environments: using an individual computer (Single PC), in two coupled computers (Double PC) and finally, in a system of three connected computers (Triple PC). All the computers had the same conditions: 7.0 GHz and 8 GB of RAM with the W8 operating system installed. The SDDL / Contexnet protocol was used for communication and all tests performed using a local Wi-Fi network (IEEE 802.11bgn). The experiments were carried out in a period of time of 25 minutes.

4.1 Ethereum Network Evaluations

To evaluate the performance of the Ethereum Network, three (3) different configurations and four (4) different parameters were analyzed in each system. The average number of events sent per sensor varied between 8529 ± 5 for the first configuration and 11290 ± 13 for the third. The analysis of the average of the time it takes for each transaction to be complete was performed (**Figure 12a**), and it was possible to observe a decrease in the average running time of Configuration 2

and 3 in relation to the first condition. In all configurations, the use of a system of three connected computers increased the efficiency of the process, with Configuration 3 (simple transaction with data buffering and CEP) using a three-computer system, being the lowest value.

The evaluation of the number of pending transactions was made by analyzing, at each instant of time that a new transaction was executed, how many transactions had not yet been completed. As a result of the decrease in the average execution time, it was also observed a smaller number of pending transactions in the configuration 3 and 2 in relation to the previous configurations, the first configuration using a single computer being the one that showed the highest value (**Figure 12b**).

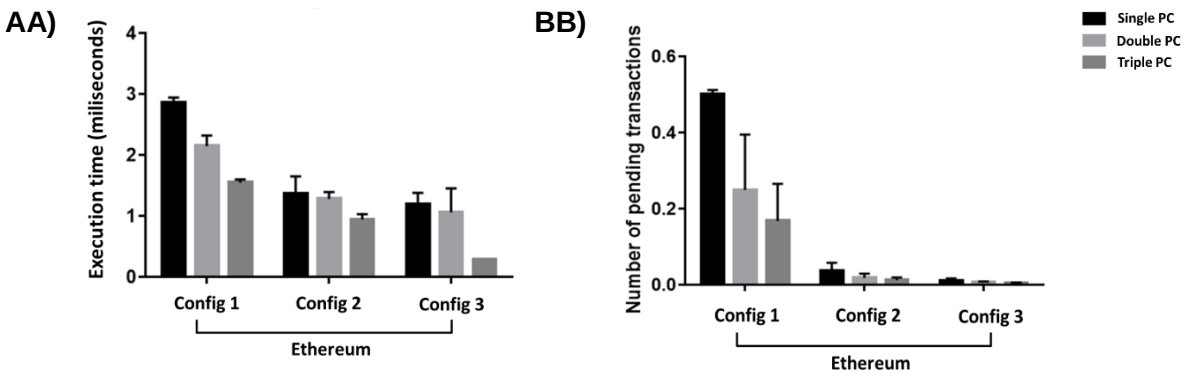

Figure 12. Graphical representation of the performance of Ethereum network, evaluating the average.

4.2 IOTA Network Evaluations

To evaluate the performance of the IOTA network, the same configurations used in the Ethereum network were analyzed. When analyzing the average time it took for the set of transactions to be executed per sensor, the use of data accumulation by buffering data with simple transactions showed visible differences in the results related to the use of simple transactions in the IOTA network. In this Configuration 2, all sensors emit the events and a buffer accumulates all the transactions and sends them in groups of 10. In the third configuration, a reduction of the operating time per sensor was also observed in relation to the previous configurations (**Figure 13a**).

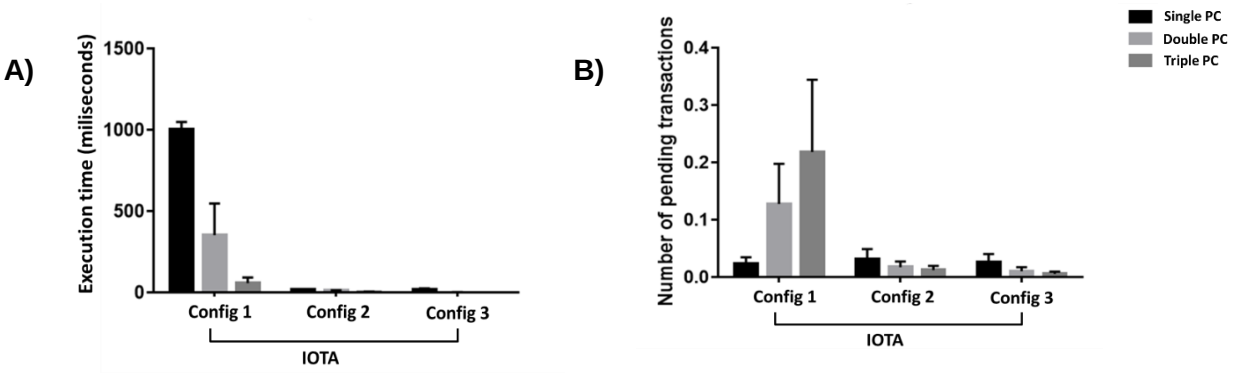

Figure 13. Graphical representation of the performance of the IOTA network, evaluating average running time **(A)** and the number of pending transactions **(B)**.

Finally, the number of pending transactions in each instance of time was also evaluated. A lower number of pending transactions was generally observed in the second configuration in relation to the first condition. The third configuration evaluates the behavior of the system using CEP from FOG computing, where a rule for filtering the data was previously established. The data accumulation using data buffering was also implemented. The use of these tools allowed decreasing the number of pending transactions in relation to the previous configurations, as observed in **Figure 13b**.

The use of a distributed system using three computers had a significant effect on the reduction of execution time in all the configurations analyzed. Similar behavior was observed for the number of pending transactions, except in Configuration 1, where the number of transactions accumulated was independent of the number of computers used.

4.3 Ethereum and IOTA Network Comparison

The performance of the Ethereum and IOTA network in the three Configurations was compared, taking into account two (2) of the parameters previously mentioned. **Figure 14** shows the graphical representation of the performance of both networks, including the averages and standard deviations of the three independent experiments. All statistical analyses were carried out using GraphPad Prism version 6.00 for Windows (GraphPad Sofware, San Diego

California USA). A one-way ANOVA test was used to compare the differencesamong groups, with a confidence interval of 95% for all experiments.

In all the parameters previously evaluated (**Figure 12 and 13**), statistically significant differences between Configuration 3 values of both ledgers were observed. Lower values of the average time of execution and pending transactions were observed in Configuration 3 of Ethereum using three computers, achieving a reduction of 52% and 32%, respectively, in relation to the same configuration in IOTA.

In order to compare the efficiency of CEP and data buffer in Ethereum in relation to IOTA, the percentage of reductions of sent transactions was also analyzed. For that, the number of transactions sent per sensor, in relation to the number of total events received, was compared (**Figure 14a**). The following formula was used:

$$\%\, reduction = \frac{No.events - No.transaction}{No.events} \times 100$$

where *No. events* represents the number of events, and *No. transaction* the corresponding number of transaction.

Configuration 1 was used as a control, where no event was filtered and no transaction was accumulated. The use of both, data buffer and CEP, increased the percentage of reduction in the number of transactions in both ledgers (Ethereum: 97.83% and IOTA: 94.28%), in relation to Configuration 2 (Ethereum: 92.28%and IOTA: 90.08%), where data buffer was only used; and Ethereum achieved the greatest efficiency in the reduction, with a percentage 3.54% higher than the reduction percentage obtained with the Tangle ledger in the same condition. In general, the use of a distributed system using one, two, or three computers did not have a visible effect on the percentage of transaction reductions (**Figure 14a**).

The cost per sensor was also analyzed (**Figure 14b**). In the IOTA network thereis no need to pay a fee, so the total cost to pay is 0 for all the configurations. In Ethereum, this cost is measured in gas units and represents the fee that is paid to theminers for inserting the transactions into the blocks. Normal-sized transactions havea gas price of 90000. The total cost per sensor is directly related to the

number of transactions, therefore, lower costs were observed in configurations 3 and 2 in relation to the first, with the third one showing the lowest values. A reduction of 95,96% in the fee cost of the Ethereum network was observed, comparing Configuration 3 with the Configuration 1 values using a three computers system.

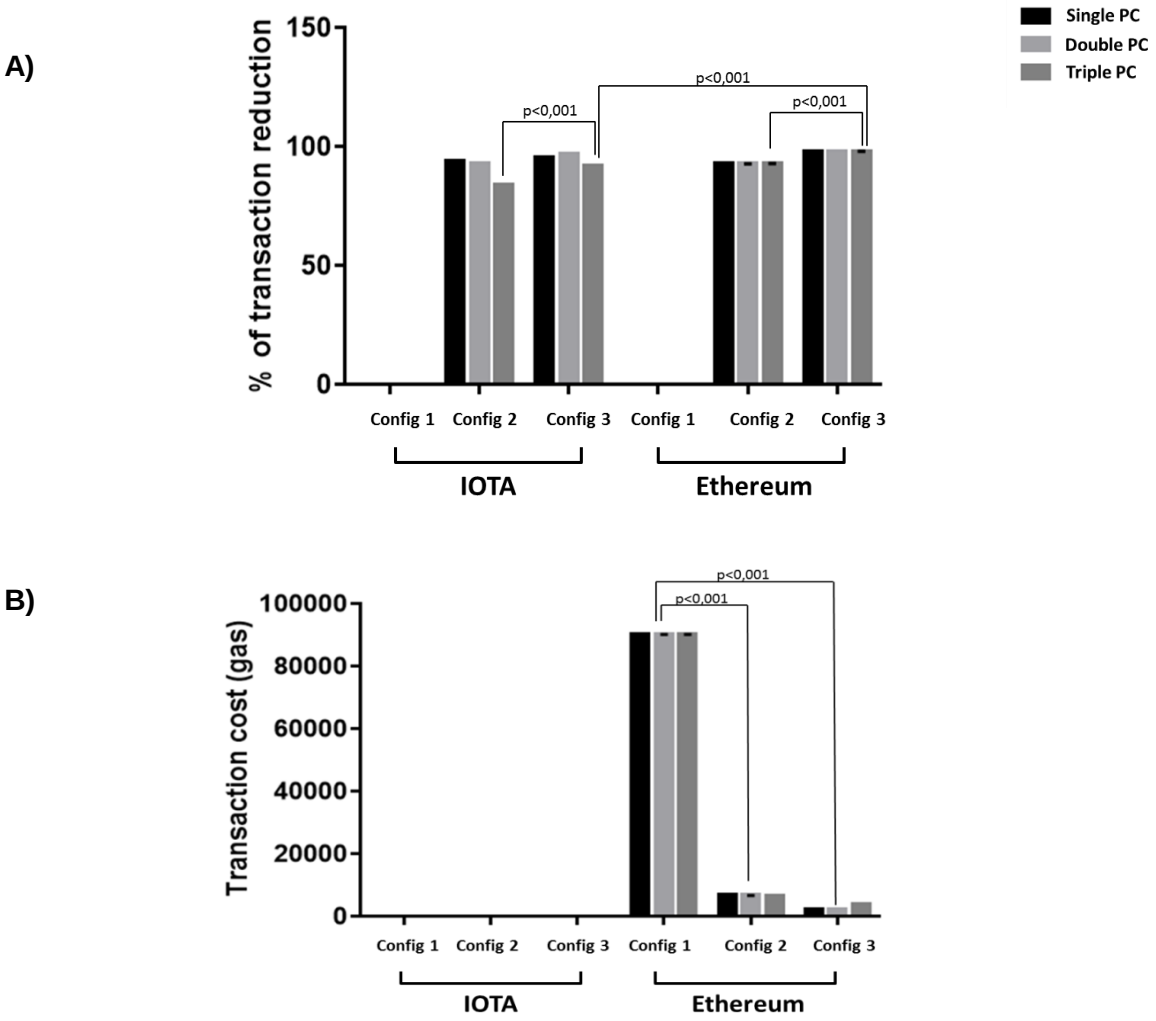

Figure 14. Graphical representation of the performance of Ethereum and IOTA network, evaluating the percentage of transactions reduction **(A)** and transactions cost **(B)**.

5. Related Work

The issue of privacy and the anonymity of data as a problem for IoT have been widely discussed and several authors, such as (IBM, 2014) and (Zyskind, 2015) suggest the use of blockchain to solve it. Bitcoin (Nakamoto, 2008) or Ethereum (Buterin, 2015) blockchain allows the storage of data in the distributed ledger, but with very little storage capability and at a very high cost. Because blockchain is a relatively new technology, its use in the Internet of Things is increasing;however, there are few research papers published to date.

Blockstack (Muneed, 2016) and Metadisk (Wilkinson, 2014) are two projects that use blockchain networks for data storage with proof of work algorithm, the same consensus algorithm of Ethereum or Bitcoin. Blockstack is an open-source project where a name storage system, based on the blockchain of Namecoin, was created. The aim of Blockstack was to create a secure and human-readable network of names, to link these names with some arbitrary values. To solve the problems related to the limitation of the storage capacity, some lists of links of named pairs of values were made. Initially, the project was using the Namecoin blockchain, which is an initial bitcoin fork; but in the end, the whole project migrated to the blockchain of Bitcoin due to security problems related to the vulnerability of the previous network, in relation to a critical security problem in which a single miner consistently had more than 51% of the total computing power.

Metadisk, on the other hand, is another open-source data storage project created to demonstrate conceptually that blockchain data storage can be more decentralized, secure, and efficient. They propose an autonomous and trustless cloud storage model in which users can upload and download files from the network in a secure manner. The blockchain is used as a data store only for file metadata, and the cryptocurrency is used as a payment mechanism to exchange storage space and bandwidth. Both projects have scalability limitations because of the block structure and the consensus algorithm (Wood, 2014) which makes it difficult to reuse them in IoT systems. In addition, in order to insert data in the blockchain that exceed the space limit of the transaction, extra non-relational databases were used, which solvethe storage difficulties of these networks.

Some projects have also described the storage of healthcare data using blockchain, as described by Esposito *et al.* (Esposito, 2018), who proposed an IoT application that allows the storage of clinical data related to patients, family, and friends. Our project is also related to data storage in distributed ledgers to improve healthcare management; but we describe data storage related to the position of each person within the hospital (whether patients, doctors, or staff) and not clinical data.

The data storage of Edge computing devices in the blockchain has already been explored by Huang *et al.* (Huang, Chen, Wu, & Huang, 2018), who created a communication system between devices using blockchain, being used as a payment method. In our project, the "position data" stored by our system are received by sensors and different devices that are part of Edge computing. However, the Contexnet Middleware was used for distributed communication with the blockchain. This middleware allows the use of Edge Computing techniques such as CEP, and data buffer, for data filtering.

6. Discussion and Conclusion

6.1 Discussion

To access the distributed ledgers through ContextNet, the DLedger service was implemented, which is responsible for connecting the IoT applications with the different immutable public records, where the data is stored. With this service it is possible to choose between different technologies for data storage in distributed ledgers, maintaining the integrity and privacy of the data. One of the advantages of using ContextNet as an access layer is the use of its functionalities that allow improving performance locally, at the edge of the network, using Fog computing. The use of this middleware also reduces the amount of data that is required to be sent to the cloud, by allowing the use CEP (Laboratory for Advanced Collaboration(LAC), 2017) and data buffer. CEP allows filtering events and only those that are relevant and necessary are stored; while the data buffer sends data events from thesensor as a bulk message to the cloud, which reduces the frequency of data insertioninto the DLedger.

To store data using distributed ledgers, two different technologies were used: the Ethereum blockchain and the IOTA tangle. Both technologies have the same consensus algorithm (PoW) for the validation of transactions; although its use difficult to handle multiple IoT device requests, resulting in the loss of some data before its insertion into the ledger. To minimize these data losses, some of the main features of ContextNet were used to work with IoT devices.

The first blockchain integrated into the project was Ethereum, which has a large community of new developers, as well as being the first blockchain to implement smart contracts for autonomous applications. The Ethereum blockchain provides a persistent structure where data is stored in a reserved space in each transaction. This ledger has difficulty in partitioning and parallelizing the transactions (Wood, 2014),so it cannot manipulate much information, which limits the scalability during the implementation of an IoT system. The use of data buffer with the Ethereum blockchain allowed the reduction of transactions in relation to the simple transactions due to the accumulation technique, which allows reducing a set of events to a single transaction by concatenating the data of the events. This led to a reduction in the average execution time and, as a result, the number of

pending transactions and the total fee cost also decreased. In addition, the combination of data buffering and the CEP service further reduced the values of previously analyzed parameters in relation to the use of data buffer alone. These transaction reduction techniques, each separately, provided an increase in the performance of the network nodes, but the best results of the system were obtained with the combination of both of them.

In addition, the IOTA ledger was also added to the DLedger service. When analyzing the Tangle performance using the different filtering and accumulation techniques, a similar behavior to the Ethereum network was observed. As mentioned above, filtering queries made by CEP allow reducing the number of events that become transactions, while the data buffer concatenates numerous transactions, sending several at the same time. The decrease in the number of transactions improves the performance of the system, also decreasing the waiting time to process new transfers and the number of pending transactions.

Overall, in both distributed ledgers the best results were obtained through the combination of data buffer with CEP, where a statistically significant reduction in the values of all the parameters was observed. Thus, an improvement in system performance was also observed. In addition to reducing the number of transactions, the use of both techniques also reduces the rate that must be paid for the validation of transactions. In the case of Ethereum, it would be the gas paid to the network by the mining company and there was a reduction in units of gas by using CEP and data buffer, which represents a considerable saving. In the case of Tangle, only the energy consumed by the nodes is taken into account, since the cost of inserting the transaction is 0. This is an important parameter when deciding which ledger to use, since even with the reduction in this value obtained with the use of both accumulation techniques, the expense can be considerable, making the entire process more expensive.

When analyzing the reduction in the number of transactions it is possible to conclude that the Ethereum network, in combination with data buffer and CEP, showed better performance in relation to the IOTA network in the same conditions. Previous works, such as Cowry Platform (Odiete, 2018), have used blockchain for the storage of IoT metadata, but this is the first time that transaction

accumulation and filtering techniques are used to increase the efficiency of two different ledgers. Although we strongly recommend the use of the Ethereum network using CEP and data buffer, due to the results obtained; we consider that multiple factors must be analyzed individually in order to select the ledger and the ideal configurations for each project.

In distributed ledgers, the validation of the network is an intrinsic property, so the adoption of peer-to-peer computing, such as the blockchain of Ethereum or Tangle of IOTA, to process and store all these transactions in the IoT scale, can also facilitate and reduce the costs associated with the installation and maintenance of large centralized data centers. The solution will be part of ContextNet as new services for IoT gateways and the use of decentralized ledgers will significantly increase the security of data storage with this middleware.

6.2 Conclusions

We have developed a secure transaction storage system using distributed ledgers as a middleware service for the IoT Middleware ContextNet. The use of ledgers for the storage of context data of IoT devices is an interesting solution, due to its decentralized, trustless, and distributed architecture. With the proliferation of blockchain technologies, the number of available ledgers has increased, as well as the optimization mechanisms to improve the quality of the service. Factors such as the reduction of the number of transactions, the execution time, the number of pending transactions, and the cost of the process must be taken into account, having different relevance depending on the aims of each project. After evaluating the performance of each ledger using accumulation and filtering techniques, we observed that the use of CEP and data buffer significantly increased the performances in both, the Ethereum and IOTA networks, in relation to the absence of these techniques or the use of data buffer alone; which was reflected in a reduction of the frequency and number of transactions transmitted.

6.3 Future works

The results presented in this project demonstrated the effectiveness of the use of accumulation and filtering techniques to improve the performance of distributed ledger networks. However, the IOTA network, one of the analyzed ledgers, has not

implemented intelligent contract services. These smart contracts are very useful for the development of autonomous IoT applications. An alternative is to use the DLedger service of ContextNet, which allows combining the best of the two analyzed ledgers. The IOTA network facilitates a DAG that solves scalability problems with the parallelization of validations; and with the DLedger service, a layer of the smart contracts offered by the Ethereum network over the IOTA tangle can easily be created. It would be interesting if future research implements this alternative in order to obtain a more complete system so that it can be used by autonomous applications for the Internet of Things. We also propose to explore new options for the reduction of transactions to the network with the compression of the data, as well as to incorporate new technologies of different distributed ledgers into the ContextNet middleware.

7. Bibliography

AFSHAR, V. (2014). *Huffpost.* Retrieved 2017, from The Internet of Things 2014: http://www.huffingtonpost.com/vala-afshar/the-internet-of-things-20_b_5693200.html

BUTERIN, V. (2013). **Dagger: a memory-hard to compute, memory-easy to verify scrypt alternative.**

BUTERIN, V. (2015). *A* **next generation smart contract & decentralized application platform.** Ethereum White Paper.

CHRISTIDIS, K. (2016). **Blockchains and Smart Contracts for the Internet of Things**. 4 (2169-3536).

CONOSCENTI, M. V. (2016). **Blockchain for the Internet of Things: a Systematic Literature Review.** Agadir: IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA).

COULSON, G. B. (2002). T**he design of a configurable and reconfigurable middleware platform.** 15(2: 109-126).

CRAIN, T. G. (2017). **Blockchain Consensus.**

CROSBY, M. N. (2016). **BlockChain Technology: Beyond Bitcoin**. *2* (6-19).

DORRI, A. K. (2016). **Blockchain in Internet of Things: Challenges and Solutions.**

EsperTech. (2018). **Esper - Complex Event Processing.** Retrieved from http://www.espertech.com/esper/

ESPOSITO, C. D. (2018). **Blockchain: A Panacea for Healthcare Cloud-Based Data Security and Privacy.**

GREENSPAN, G. (2015). **Avoiding the pointless blockchain project.** Retrieved 2018, from Private blockchains: https://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project/

ZYSKIND, G. (2015). **Decentralizin Privacy: Using Blockchain to Protect Personal Data.** Security and Privacy Workshops (SPW), IEEE. IEEE.

HUANG, H. CHEN, X.; WU, Q.; HUANG, X. and SHEN, J. (2018)**. Bitcoin-based fair payments for outsourcing computations of fog devices.**

HALLER, S. (2010). **The Things in the Internet of Things.** Tokyo, Japan: Internet of Things Conference.

IBM. (2014). **Device democracy: Saving the future of the Internet of Things.**

IOTA GUIDE. (2018). Retrieved 2018, from https://domschiener.gitbooks.io/iota-guide/content/chapter1/transactions-and-bundles.html

KOPETZ, H. (2011). **Internet of Things.** Springer.

Laboratory for Advanced Colaboration (LAC). (2017). Retrieved from http://www.lac-rio.com/dokuwiki/doku.php

LABS, P. (2017). **Filecoin: A descentralized Storage Network.**

MADISETTI, B. A. (2016). **Blockchain Platform for Industrial**. *9* (553-546).

MATTERN, F., & FLOERKEMEIER, C. (2010). F**rom the Internet of Computers to the Internet of Things.** From active data management to event-based systems and more. Springer, Berlin, Heidelberg, 242-259.

MEOLA, A. (2016, 12 19). **Business Insider.** Retrieved 2017, from Internet of Things devices, applications & examples: http://www.businessinsider.com/internet-of-things-devices-applications-examples-2016-8

RAZZAQ, M. A; GILL, S. H.; QURESHI, M. A. and ULLAH, S. (2017). Security Issues in the Internet of Things (IoT): A Comprehensive STUDY. (6).

MUNEED, A.; NELSON, J.; SHEA, R. and FREEDMAN M. J. (2016). **Blockstack: A Global Naming and Storage System Secured by Blockchains.** USENIX Annual Technical Conference.

NAKAMOTO, S. (2008). **Bitcoin: A peer-to-peer electronic cash system.**

NARAYAN, P. (2017). **Building Blockchain Projects.** Packt Publishing Ltd.

ODIETE, O. L. (2018). **Using Blockchain to support data and service managment in IoV/IoT.** 733(344-362).

OLIVEIRA-CARVALHO, F. (2017). **Descoberta Contínua de Serviços em IoT.** Dissertação de Mestrado.

ORACLE. (2018). Retrieved from Oracle: https://docs.oracle.com/cd/E13157_01/wlevs/docs30/epl_guide/overview.html

BRIDA, P.; GABORIK, F.; DUHA, J. and MACHAJ, J. (2011). I**ndoor positioning system designed for user adaptive systems.** New Challenges for Intelligent Information and Database Systems, 351 , 237–245.

PAVEL, K.; MALY, F. and KOZEl, T. (2016). **Improving Indoor Localization Using Bluetooth Low Energy Beacons.** http://dx.doi.org/10.1155/2016/2083094, 11.

PERCIVAL, C. (2009). **Strongerker derivation via sequential memory-hard functions.**

POPOV, S. (2016). **The tangle**. (cit. on, 131).

POPOV, S. (2018). **Local modifiers in the Tangle.**

SANTUCCI, G. (2010). **The internet of things: Between the revolution of the internet and the metamorphosis of objects.** In Vision and Challenges for Realising the Internet of Things. 11-24.

SERGUEI, P., SAA, O., & FINARDI, P. (2018). **Equilibria in the Tangle.**

TALAVERA, L. E. and ENDLER, M. (2016). **An Energy-aware IoT Gateway, with Continuous Processing of Sensor Data.** SBRC.

WILKINSON, S. L. (2014). **Metadisk: Blockchain-Based Decentralized File Storage Application.** Technical Report. Technical Report .

WOOD, G. (2014). **Ethereum: A secure decentralised generalised transaction ledger.** Ethereum project yellow paper 151, 1-32.

ZHAO, X.; XIAO,Z.; MARKHAM, A.; TRIGONI, N. and REN, Y. (2014). **Does BTLE measure up against WiFi? A comparison of indoor location performance**. Proceedings of 20th European Wireless Conference(IEEE, Barcelona, Spain), 1-6.

XIA, F.; YANG, L. T.; WANG L. and VINEL, A. (2012). **Internet of Things.** International journal of Communication System, 25:1101–1102.