Information Fusion 49 (2019) 227–239

Contents lists available at ScienceDirect

Information Fusion

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

Full Length Article

Non-linear adaptive closed-loop control system for improved efficiency in IoT-blockchain management

Roberto Casado-Vara[a,*], Pablo Chamoso[a], Fernando De la Prieta[a], Javier Prieto[a], Juan M. Corchado[a]

IoT Digital Innovation Hub, University of Salamanca. Edificio Multiusos I+D+i, Salamanca, 37007, Spain

ARTICLE INFO

Keywords:
Adaptive closed-loop
Control system
Algorithm design and analysis
Queuing theory
IoT
Non-linear control

ABSTRACT

IoT network generates a large amount of data. This means that the monitoring and control of these networks and the transfer of packets from the IoT network to the server can cause communications to collapse. On the other hand, due to the large volume of data stored in the databases the monitoring of the IoT network needs very powerful servers to have a high degree of efficiency. This paper presents a novel adaptive closed-loop control system and speed up searches model to improve the monitor and control efficiency in IoT networks, specially those which are based in blockchain. The non linear control model under consideration includes a new way to evaluate the optimal number of blocks that should be at the queue of the miners' network in order to make the process efficient through the use of queuing theory. Also, a new system to speed up searches is presented by using hashmaps, which makes the monitoring process faster, reliable and efficient. The efficiency of the presented approach is illustrated by a numerical case study.

1. Introduction

A Networked Control System (NCS) is a control system wherein the control loops are closed through a communication network [1]. Their advantages include low installation and maintenance costs, flexibility and reduced wiring [2]. These benefits make NCSs applicable to a wide range of fields [3]. However, NCS's weaknesses are caused by delays in random communications and packet loss. In terms of current industrial applicability, traditional point-to-point centralized control is unsuitable since it does not meet the new requirements such as modularity, decentralized/distributed control, quick and easy maintenance and low cost. For these reasons, in recent years NCS has been a major focus of attention in both academic research and industrial applications, contributing to significant progress in this field [4]. Internet of Things (IoT) networks usually consist of many sensors, as well as controller and actuator nodes. These distributed IoT nodes compete to send their data to the network. There is a need to implement a control system that monitors and controls the sending of packets from the IoT nodes to the network. In addition, IoT networks generate large amounts of data continuously. The volume of data makes it very difficult to monitor and control IoT networks. To control the IoT network it is necessary to search within the databases. Causing a delay in the functioning of the control system within the IoT network. This also entails high levels of consumption of energy and resources.

Although the problem of the limitation of communications in networks has been well studied, there are relatively few works on the optimization of queuing analysis. Furthermore, none of those works incorporate properties of the network into the control system [5–7]. The problem remains unresolved; the properties of IoT networks continue to affect system performance to a large degree. Thus, these properties must be taken into consideration by the control algorithm used by the networks [8]. Most of the research is based on optimizing energy consumption for improved performance of NCSs. However, the use of NCSs with IoT network properties, including queue analysis, service rate and packet dropout, must be subjected to further research. In order to make the monitoring and control of the IoT network more effective, some researchers proposed new techniques that increase the speed at which big data databases are searched. Zhou et al. designed a framework to bridge multi-target query needs between users and the data platform, including required query accuracy, timeliness, and query privacy constraints [9]. Another research proposed the use of binary hashing for greater search speed; Cao et al. reviewed and compared those hash techniques through different experiments [10].

This paper proposes a new IoT architecture that covers the research gaps in the monitoring and control of the queues that control the sending of packets from the IoT nodes to the big data databases. To optimize the process of sending data over the network, a novel adaptive control algorithm is proposed in this manuscript. The adaptive control algorithm

* Corresponding author.
E-mail addresses: rober@usal.es (R. Casado-Vara), chamoso@usal.es (P. Chamoso), fer@usal.es (F. De la Prieta), javierp@usal.es (J. Prieto), corchado@usal.es (J.M. Corchado).

https://doi.org/10.1016/j.inffus.2018.12.007
Received 31 August 2018; Received in revised form 6 November 2018; Accepted 31 December 2018
Available online 4 January 2019

analyzes the queues within the network nodes and the server, improving their performance and impeding the queue from becoming saturated. If the queue is saturated, the adaptive control algorithm will make the necessary changes to the network properties in order to desaturate the queues. In addition, we proposed a novel hashmap-based search system to speed up the search time in big data databases. However, one of the main problems of IoT networks is security. To secure communications, we have integrated blockchain technology in our architecture. Although this is not the main objective of our research, blockchain technology stores the hashmap that is used for speed searches in the big data databases. This improves the monitoring and control efficiency of the IoT network.

The main contribution can be summarized as follows:

1. A new model that determines the optimal number of blocks that should be in the queue of the miners' network to make the mining process efficient.
2. A new adaptive control algorithm to improve block sending in an IoT network. A $M|M|1$ queue model is used incorporate the IoT network properties into the queuing control model.
3. A novel model is proposed for increased search speed via hashmap. We show that using a hashmap stored in the blockchain we can improve the data search in big data databases.
4. An architecture capable of integrating the above contributions to provide an innovative IoT platform.

This paper is organized as follows: Section 2 overviews works related to the key aspects of this article. The details of our proposal are presented in Section 3. Section 4 presents two simulations, the results of which validate the competency of the proposal. Finally, Section 5 draws conclusions from the conducted research and describes future lines of work.

2. Related work

2.1. Blockchain and sidechains

A blockchain is a distributed data structure that is replicated and shared among the members of a network [11]. It was introduced with Bitcoin [12] to solve the double-spending problem [13]. As a result of how the nodes in Bitcoin (called miners) mutually validate the agreed transactions, Bitcoin's Blockchain establishes the owners and states what they own. A blockchain is built using cryptography. Each block is identified by its own cryptographic hash and each block refers to the hash of the previous block; this is how a link between the blocks is established, forming a blockchain [14,15]. For this reason, users can interact with blockchain by using a pair of public and private keys. In a blockchain, miners must agree on the transactions and the order in which they have occurred. Otherwise, the individual copies of this blockchain can diverge producing a fork; this means that miners have a different view of how the transactions have occurred, and it will not be possible to keep a single blockchain until the fork is not solved [16,17]. To achieve this, it is necessary to have a distributed consensus mechanism in every blockchain network [18]. Blockchain's way of solving the fork problem is to link each blockchain node with the next block. This is done by finding a correct random number with SHA-256 [19,20] so that the number of zeros corresponds to the figure required by blockchain. Any node that solves this puzzle has generated the so-called proof-of-work (pow) and shapes the chain's next block. Since a one-way cryptographic hash function is involved, any other node can easily verify that the given answer satisfies the requirement [21]. Notice that a fork may still occur in the network when two competing nodes mine blocks almost simultaneously. Such forks are usually resolved automatically by the next block [22,23]. The term "sidechain" was first described in the paper "Enabling Blockchain Innovations with Pegged Sidechains" [24]. The paper describes "two-way pegged sidechains", a mechanism that allows the user to move the cryptocurrency within a sidechain, proving that some cryptocurrency that had previously been in a user's possession had been "locked".

2.2. IoT Platforms

Here, several existing IoT architectures are explained. 1) Three-Level Architecture. The three-level architecture is elementary for IoT, it has been implemented in a large number of systems. In [25] and [26], different scalable architectures for IoT large-scaled network are presented. Usually, the following three levels can be identified in IoT networks: Internet-oriented, sensors and actuators, and knowledge. But in [27], IoT architectures have the following three levels: Perception level, network level and application level. 2) SDN-Based Architecture. Qin et al. [28], proposes an SDN-based IoT architecture to increase the quality of service (QoS). Casado-Vara et al. [29], proposes an algorithm to increase the quality of data that can be used in IoT architectures, thus improving the QoS of IoT architectures. 3) QoS-Based Architecture. Jin et al. [30], propose four different IoT architectures that allow several applications for intelligent cities to include their QoS requirements. This architecture was improved in [31] to reduce the stress and congestion among nodes. 4) SoA-Based Architecture. SoA is a component-based model that is designed to connect various services through applications and interfaces [32]. SoA architectures consists of four cooperating levels: 1-perception level, 2-network level, 3-service level and 4-application layer. In the fourth level of SoA-based architecture it is used to store and analyze data from IoT devices. 5) CloudThings Architecture. In Zhou et al. [33] an IoT-enabled smart home scenario is proposed to analyze the IoT requirements. Hao et al. [34] proposes an architecture called Data Clouds, based on centralized information to increase the reliability of the new generation of Internet services. Nowadays, most of these architectures are implemented in the industry or smart cities. Although these architectures are effective for the time being, we cannot be sure that for the challenges of the future they are reliable, and thus need to be re-examined.

2.3. Blockchain in IoT platforms

At the moment there are 5 billion IoT devices connected, and this number will continue to grow to 29 billion in 2022 [35]. Each IoT device produces and exchanges data with the Internet. So, considering the large number of IoT devices, it's easy to understand that we're dealing with continuous massive production. In our opinion, blockchain represents the piece of the puzzle that solves the problems of privacy, large-amount of data+ and trust in IoT. As a blockchain it is decentralised, autonomous and without trustless features make it suitable to be applied in different scenarios such as smart cities [36,37], smart property [38] and smart homes [39,40] as well. Useful applications of blockchain in IoT include, Miller et al. [41] proposes an interesting application of blockchain in IoT to solve the challenges of the supply chain and Novo et al. [42] shows an architecture for scalable access management in IoT.

2.4. Queuing theory

The theory of queues is a field of mathematics that has been studied extensively over the years. The queuing theory has different applications in mathematical models [43], computer applications [44], linear statistical inference [45] and its applications and engineering systems [46]. Some of the newest work with queuing theory is in healthcare [47]. In this work authors present several applications of queuing theory to optimize the healthcare process. In another research, the relationship between telecommunications and queuing theory is shown. In their paper, the authors employ different queues and queue networks to increase the efficiency of telecommunication processes [48]. Srivastava et al. present an in-depth study on the use of queuing theory together with big data technology for the optimization of the computational analysis of the

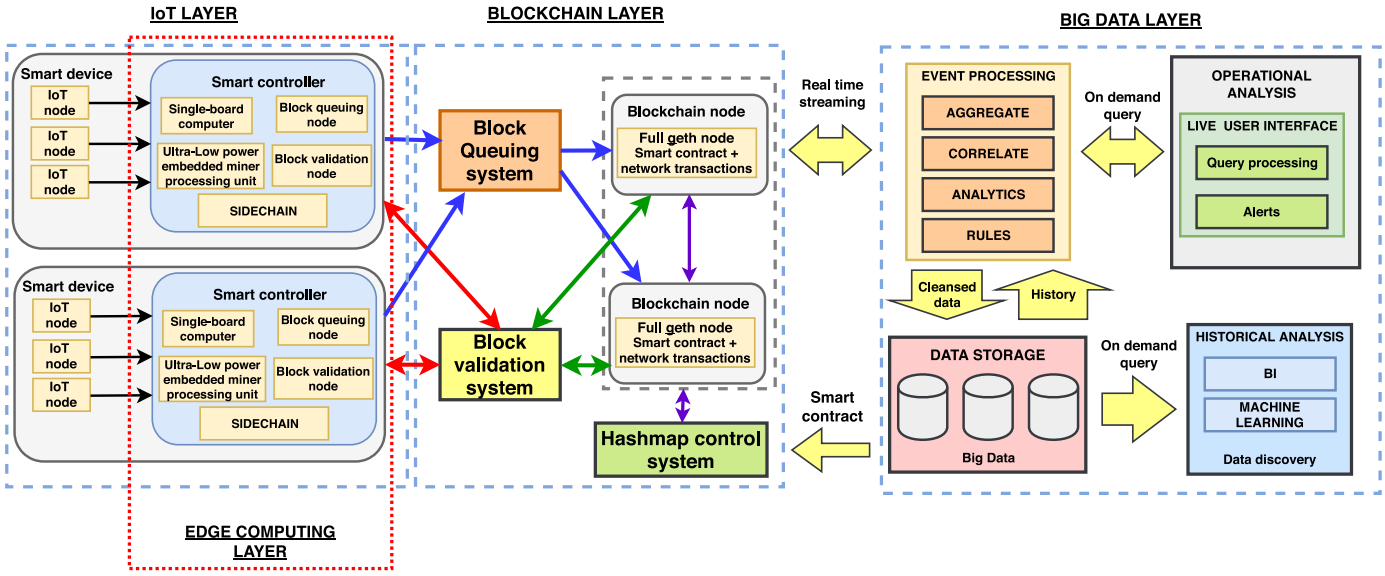

Fig. 1. Main architecture. This architecture improved the monitoring and control system of the blocks sent to the blockchain from the IoT nodes. Also, this architecture allows for faster search of data via hashmap in the blockchain.

data [49]. In recent years, some researchers have been optimizing the IoT network using queuing theory. Choi et al. present a $M|M|1$ queue system, in their approach they assume that the repository (server) is active when one (or more) vehicles are in motion [50]. Zhui et al. formulates the dynamic user scheduling and power allocation problem as a stochastic optimization problem with the objective to minimize the total power consumption of the whole network under the constraint of all users' long-term rate requirements [51]. Chekired et al. proposes in their work a hierarchical fog servers' deployment at the network service layer across different tiers. Using probabilistic analysis models, they prove the efficiency of the proposed hierarchical fog computing compared with the flat architecture [52].

The methods presented in the reviewed literature overcome different challenges, such as distributed IoT architecture, security issues and data confidentiality and authentication among other. All the investigations found in the state of the art present approaches of hybrid architectures between IoT and blockchain. However, we observed a gap in the literature review since there are no algorithms that could automatically control the data packet flow in the IoT network. In our work, we use an adaptive control algorithm to control packet queues from IoT devices to blockchain. In addition, we have found the optimal number of blocks that have to be waiting in the queue of the miners' network to make it profitable for the miners to mine those blocks. In this way, the innovative adaptive control algorithm can vary the mining capacity of the miners' network to optimise energy consumption and prevent tailings from becoming saturated. Finally, a new IoT architecture is proposed that can manage the proposed queuing system and also, by adding big data, you can speed up the data searches and their analysis. While with blockchain the data is secured and its reliability is increased.

3. Proposed architecture and devices

This paper describes the novelties of this work and the final architectures which integrates them. Our main focus was the design of a new self-adaptive control algorithm for the monitoring and control of block flow from the IoT devices to the blockchain (see Fig. 1). Also, we propose a new way of storing data in a big data ecosystem. The sensor ID, timestamp and query are stored in a hashmap in the blockchain. In this way, it is easier to search data what also contributes to optimized monitoring and control of the IoT devices. This architecture works as follows: 1) Smart devices collect data from the environment and start processing them. In the first layer, smart devices incorporate the edge computing paradigm which allows to execute smart contracts to insert the data from IoT devices into a sidechain (i.e., a permissioned blockchain designated for the sole use of smart devices). Once the smart devices have finished validating and inserting data into blocks, they execute a new smart contract that sends the blocks to the miners' network for inclusion in the blockchain. 2) Blockchain nodes (i.e., miners) receive blocks already built from the sidechain. Then, the blockchain miners' network validates the already built block in the sidechain and includes it in the blockchain. Thus, the miners only have to calculate the hash that allows to add the node to the blockchain and computing power is not required to build the block. 3) Once data of the IoT devices is already in the blockchain, through a real time streaming this data is inserted into the big data layer. Once data is in HDFS (Hadoop Distributed File System) a smart contract is executed to send the addresses of the data stored in HDFS (since HDFS replicates the content and distributes it within the database) and the query to have access to the data (this query is built in MapReduce) that are stored in a hashmap. In this way, data are secured in the blockchain. Since they are stored in HDFS, their availability is high and the monitoring and control of the IoT devices is optimized by big data technologies (Business intelligence, data discovery, machine learning, etc.). The following paragraphs give an in-depth description of our proposal; the new architecture and its functioning.

3.1. Smart device layer

Fig. 2 shows how smart devices (i.e., Event producer layer) collect and pre-process (i.e., Pre-processing layer) data before sending them to the blockchain. Smart devices have the following layers: 1) Event producer layer. In this layer, IoT nodes collect data (temperature, humidity, etc.) and send them to the smart controllers for real time preprocessing. 2) Pre-processing layer. In this layer the single-board computer (in our case Raspberry Pi, although other similar devices could also be used) begins to build the block with the data collected by the IoT devices, once the block is built, the Raspberry Pi runs a smart contract that introduces data in the sidechain. To improve the processing power of the Single-board computer, a device called the "Ultra-Low power embedded miner processing unit" is added via USB. This supports the Single-board computer in the construction of the blocks. This device improve the computational power of the smart device. This way, the smart controller can run complex algorithms in the single-board

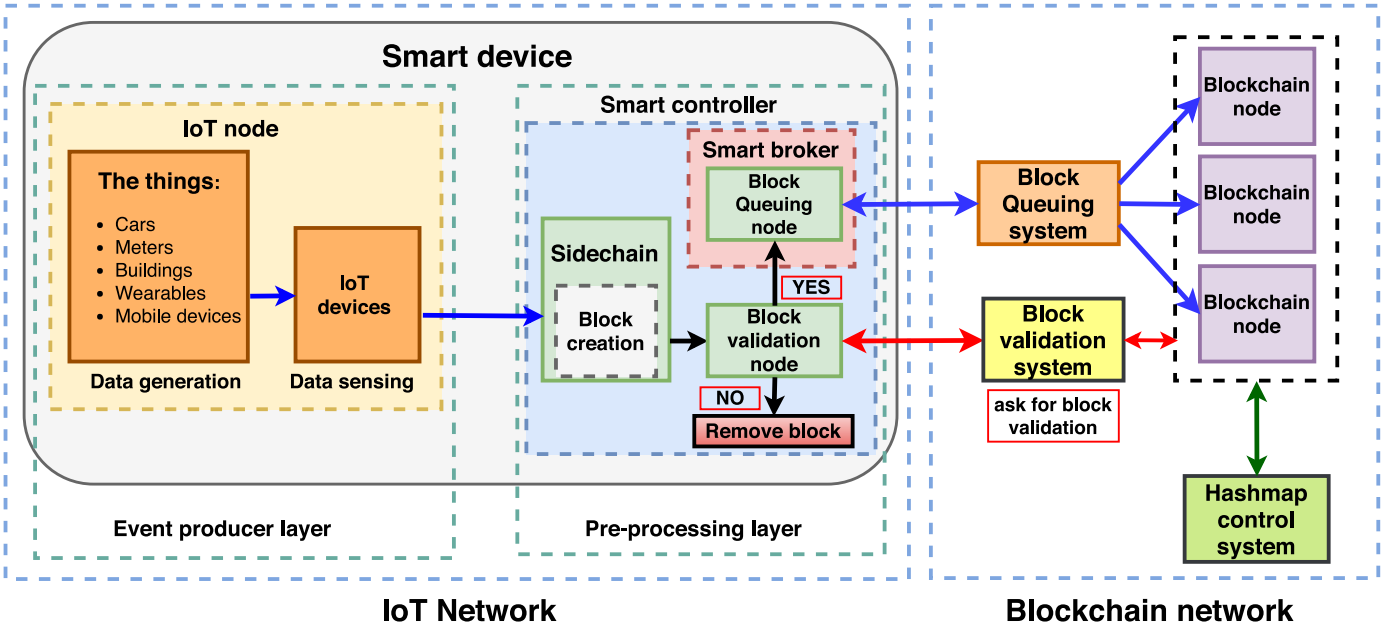

Fig. 2. Edge computing architecture. IoT nodes collect data from the things, in this architecture this occurs in the event producer layer. Moreover, in the pre-processing layer the smart controllers build the blocks. With a smart contract this block is inserted into a permissioned blockchain. Once the block is in the blockchain, a smart broker who controls a block queuing system sends the sidechain blocks to the miners.

computer without the need for a computer with high computing power. The smart controllers are in the edge computing layer. Since the smart controllers process the data, they push the computing power away from the blockchain. Once the block is built, the block validation node asks the miners network for a validation request. Then, the block validation node (this node has the right to read the blockchain) tries to validate the block it has just built. In case the block is validated, the block validation node sends the block to the smart broker to manage the block submission queue to the miners' network, which inserts the block into the blockchain. In this way, the blockchain's miners' network only has to insert the blocks into the blockchain. Moreover, the smart controllers can run the mining algorithm to help with mathematical computations of the proof of work (PoW) of the next block that is attached to the blockchain.

A detailed description of the smart controller elements and their features is presented below:

3.1.1. Sidechain

Sidechains are blockchains that are created according to the needs of the system. In this paper, the sidechains are created to store the data collected by the IoT devices. Fig. 3 shows the relationship between the blockchain and the sidechains. In the figure it can be observed that the IoT nodes send their information to the sidechains. The sidechains then share information (in the form of blocks) with the main blockchain. In this way, the sidechains can process the data of the IoT nodes in parallel, in the edge computing layer. These sidechains work in the same way as the main chain (blockchain). The IoT devices send the data to the smart controllers. Once the data is in the smart controller it reaches the sidechain. Then, the sidechain stores data until it has enough data to build a block. Blocks do not have a fixed size, but they are less than 1 MB. Once the block is created in the sidechain, data flow continues in the smart controller and the block validation node starts working.

3.1.2. Block validation node

Once the block is built in the sidechain, the smart controller has to validate the block before sending it to the miners' network. To do this, it uses the Block validation system which has a node in each of the smart controllers that are connected to the central system in the miners' network. The block validation node is asked to validate the new block that has just been built in the sidechain. Then, it sends a copy of the block to the block validation system. This system validates that data in the block are correct and are within the range of certain pre-established parameters (e.g., if the IoT nodes measure temperature in a building and send out temperatures of 100 °C, the block validation system will detect that it is an anomalous temperature and will not validate that block). Once the block validation system finishes the validation process, it returns a yes or a no to the block validation node. If the answer is no, the block is deleted from the sidechain. If the answer is yes, that block is sent to the smart broker. The block is built in edge computing near the IoT nodes. Once ready to be stored in the blockchain, it is sent to the blockchain's miners' network.

3.1.3. Smart broker

The smart broker is the manager of the queues of blocks that reach the miners' network. The block queuing system has a block queuing node on each of the smart controllers. Each of the queuing node blocks will manage a queue of its own blocks. The smart broker sends the blocks to the block queuing system that manages the queue of all smart devices. Block queuing nodes have a queue management system called $M|M|1$. Thus, there is only one block queue, whose capacity is infinite, and only one server (i.e., block queuing system). The discipline is FIFO (First In First Out). Arrivals occur according to a Poisson process of λ reason, where λ is the average number of blocks arriving per unit time and $\frac{1}{\lambda}$ is the average time between the arrival of the blocks.

Theorem 1. *Let T be the random variable that represents the time between two consecutive arrivals. Let $t > 0$ and $n(t)$ be the number of arrivals in the system up to the instant t. Since the increments are independent:*

$$P(T \leq t) = 1 - P(T \geq t) = 1 - P(n(t) = 0) = 1 - e^{-\lambda t} \quad (1)$$

then $T = Exp(\lambda)$. Reciprocally, if $T_1, \cdots, T_n$ are independent, where T_i is the time that elapses between the arrivals $(i-1)$-th and i-th, all of them with $Exp(\lambda)$ distribution, then $n(t) = P(\lambda t)$.

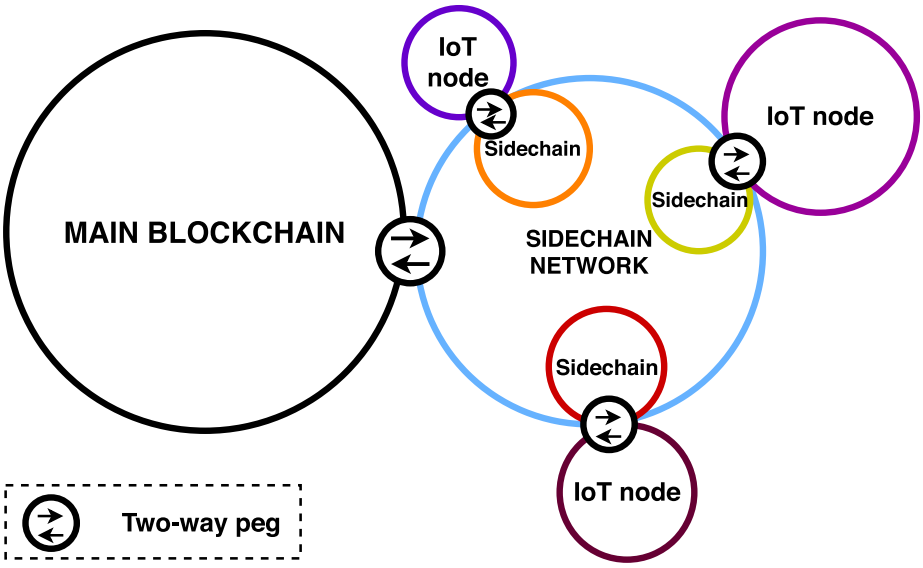

Fig. 3. Blockchain and sidechain relationship. Sidechains allow complex data updates in blockchain to be validated in a realistic environment (where current blockchain is at risk and utilizing actual network mining resources). If these updates fail, they can be removed; if they succeed, they can be incorporated into the mainchain.

Proof.

$$\begin{aligned}
P(n(t)\le n) &= P(T_1+\cdots+T_n+T_{n+1}>t)\\
&= \int_t^\infty e^{-\lambda t}\frac{\lambda^{n+1}x^n}{n!}dx = \{v=x-t; dv=dx\}\\
&= \int_0^\infty e^{-\lambda(v+t)}\frac{(v+t)^n\lambda^{n+1}}{n!}dv\\
&= \int_0^\infty e^{-\lambda(v+t)}\frac{\lambda^{n+1}}{n!}\sum_{i=0}^{\infty}\binom{n}{i}t^i v^{n-i}dv\\
&= \sum_{i=0}^{\infty}\int_0^\infty e^{-\lambda(v+t)}\frac{\lambda^{n+1}t^i v^{n-i}}{i!(n-i)!}dv\\
&= \sum_{i=0}^{n}\frac{\lambda^{n+1}t^i}{i!(n-i)!}e^{-\lambda t}\int_0^\infty e^{-\lambda v}v^{n-i}dv\\
&= \{u=\lambda v; dv=\frac{du}{\lambda}\}\\
&= \sum_{i=0}^{\infty}\frac{\lambda^{n+1}t^i}{i!(n-i)!}e^{-\lambda t}\int_0^\infty \frac{e^{-u}u^{n-i}}{\lambda^{n-i}\lambda}du\\
&= \sum_{i=0}^{n}\frac{\lambda^i t^i}{i!}e^{-\lambda t} \qquad (2)
\end{aligned}$$

□

Then, the times between arrivals of the blocks are distributed exponentially, $Exp(\lambda)$. On the other hand, the times between services will also be distributed exponentially, $Exp(\mu)$, so that μ is the average number of clients the server is capable of serving per unit of time, and $\frac{1}{\mu}$ is the average time of service. To avoid system saturation, it is shown that if $\lambda \ge \mu$ the system is not saturated, thus, the number of blocks in the queue grows indefinitely over time. Therefore, the condition in which the system is not saturated is represented by Eq. (3).

$$\rho < 1, \text{ where } \rho = \frac{\lambda}{\mu} \qquad (3)$$

In this paper we are going to assume that block queues are not saturated. When a queue is not saturated, it is also said to be in a stationary state.

ρ parameter is called traffic intensity of the system, since it measures the relationship between the number of arriving blocks and the capacity to process them. Assuming the system is not saturated, the following formula is deducted from the Eq. (4) for the p_n odds of there being any blocks in the system, where $n \in \mathbb{N}$.

$$p_n = \rho^n(1-\rho) \qquad (4)$$

The parameters used to measure the performance of the queues are: 1) Average number of customers (L). To measure the average number of blocks in the system, the Eq. (5) is used.

$$\begin{aligned}
L &= \sum_{j=0}^{\infty} j p_j = \sum_{j=0}^{\infty} j\rho^j(1-\rho)\\
&= (1-\rho)\sum_{j=0}^{\infty} j\rho^j = (1-\rho)\frac{\rho}{(1-\rho)^2}\\
&= \frac{\rho}{(1-\rho)} \qquad (5)
\end{aligned}$$

2) Average response time (W). The average response time is the average time a block remains in the system. If we suppose that when a block arrives at the system it is ahead of the other j blocks, the average time taken to exit the system will be $j+1$ times the average time of the service. The calculation of the average response time formula can be found in Eq. (6).

$$W = \sum_{j=0}^{\infty}(j+1)\frac{1}{\mu}p_j = \sum_{j=0}^{\infty} j\frac{1}{\mu} + \sum_{j=0}^{\infty}\frac{1}{\mu}p_j = \frac{L}{\mu}+\frac{1}{\mu} = \frac{1}{\mu-\lambda} \qquad (6)$$

where $(j+1)\frac{1}{\mu}$ is the time it takes a block to go through the system if there is j block ahead when it arrives. p_j is the probability that there are j blocks ahead when the block arrives at the system. In this way, the management of queues is mathematically characterized by the block queuing system.

3.2. Blockchain layer

The blockchain layer contains the blockchain network and all the systems that interact with it directly. In this paper, we propose to add three new systems to the typical blockchain ecosystem used in any architecture. These systems are described in this subsection.

3.2.1. Block validation system

The block validation system is the central node that coordinates all the block validation nodes that are in each of the smart controllers. This node is where the values allowed for the data collected by the IoT nodes are defined. In addition, the block validation node queries the blockchain with the sensor ID and timestamp, to avoid the duplication of data (i.e., the block validation node validates the block if there is no record in the blockchain of the same sensor ID and timestamp in the block.

3.2.2. Block queuing system

The block queuing system is the central node of the queuing control of this architecture. Each of the block queuing nodes creates a queue $M|M|1|\infty|FIFO|1$ (i.e., $M|M|1$ abbreviated) with the blocks built by the sidechain. Then, all these queues form a queue network (a queue network is a system where several queues exist and the blocks flow from one queue to another). All queues that coordinate block queuing nodes are directed to the queue that coordinates the block queuing system (probabilistic routing). In addition, the queuing network used in this paper is open (i.e., each block enters the system at a given time, and after passing through one or more queues, exits the system) and cyclic (i.e., a work cannot return to the same queue).

Definition An open queuing network is said to be Jackson's if (if and only if):

- There is only one kind of work (i.e., blocks)
- Each i node is a queue $.|M|c_i$
- On the one hand, routing is probabilistic, where $r_{ij} \geq 0$ is the probability of reaching the node j after leaving the node i. On the other hand r_{i0}, is the probability of leaving the system after leaving node i, where $r_{i0} = 1 - \sum_j r_{ij}$

Furthermore, the rate of external arrivals to i^{th} node is denoted by γ_i. Moreover, the total number of nodes in the network denotes K.

The queuing network used by the block queuing system is a Jackson network with $r_{ij} = 1$. Thus, all the queues of smart brokers continuously send their blocks to the block queuing system.

Since total input flow to a block queuing node must be equal to the total output flow of the node, the equations of equilibrium must be given by the following equation:

$$\lambda_i = \gamma_i + \sum_{j=1}^{K} \lambda_j r_{ij}, \quad \forall i \in \{1, \cdots, K\} \tag{7}$$

The K equations shown in Eq. (7) form a linear system with a single solution, which we will solve to find the rates of arrival λ_i at the block queuing system. In order to avoid saturation of any of the queuing block queuing system tails, the condition of the Eq. (8) must be verified.

$$\forall i \in \{1, \cdots, K\}, \quad \rho_i < 1, \text{ where } \rho_i = \frac{\lambda_i}{c_i \mu_i} \tag{8}$$

Theorem 2. *(Jackson's Theorem) In an open Jackson network of m $M|M|1$ queues where the utilization is $\rho_i < 1$ at every queue, the equilibrium state probability distribution exists and for state $\{(n_1, n_2, \ldots, n_m)\}$ is given by the product of the individual queue equilibrium distributions:*

$$p(n) = \prod_{i=1}^{K} p_i(n_i), \ \forall n_1, \cdots, n_K \geq 0 \tag{9}$$

where $p_i(n_i)$ is the probability of n_i clients in i^{th} node, calculated according to the equations of model $M|M|c$.

Proof. See [24], page 228 □

The implications of this theorem are as follows: 1) Overall rate of system outputs (throughput), which is the average number of works leaving the system per unit time, coincides with the number of works entering the system:

$$\lambda_{network} = \sum_{i=1}^{K} \gamma_i \tag{10}$$

2) Average number of works in the system, $L_{network}$, which is the sum of the average numbers of works in each of the nodes:

$$L_{network} = \sum_{i=1}^{K} L_i \tag{11}$$

3) Average time in the system, $W_{network}$, which is the average time it takes a task to go in and out of the network:

$$W_{network} = \frac{L_{network}}{\lambda_{network}} \tag{12}$$

Ratio of visits to node i, V_i, which is the average number of times a work visits node i from the time it enters the network until it leaves:

$$\forall i \in \{1, \cdots, K\}, \ V_i = \frac{\lambda_i}{\lambda_{network}} \tag{13}$$

An example of a queuing network is shown in Fig. 4.

The model proposed for M|M|1 queue optimization

We are going to optimize the performance of a $M|M|1$ system in which the server (miners' network) sometimes adjusts the PoW. The server runs blocks until it is empty. It then withdraws and does not offer its service again until there is a Q number of blocks in the queue. The arrival rate is λ, and the service rate is μ, which is $\rho < 1$. The following system costs are defined:

- C_k= cost of monetary units each time the server returns (fixed cost).
- C_h= cost of monetary units per block in the miner's network and time unit (maintenance and mining cost).

The aim is to determine the Q value, enabling the system to operate at minimum cost per time unit. The maintenance cost per unit of time (on average) is $E(N) \cdot C_h$, where $E(N)$ is the number of clients expected in the system. While the fixed cost per time unit is $\frac{C_k}{E(T_0)+T_1}$ where T_0 is the length of a cycle of unemployment and T_1 is the length of a duty cycle. Since the (average) length of a cycle of occupancy and unoccupancy (in steady state) is $E(T_0 + T_1)$. Thus, the function that will be optimized is:

$$C_h E(N) + \frac{C_k}{E(T_0 + T_1)} \tag{14}$$

Theorem 3. *Let $Q \in \mathbb{N}$ be the number of blocks needed for the miners' network to mine another block again. Let λ be the arrival rate and let μ be the service rate with $\rho < 1$. Let C_k be the monetary units cost each time the server returns and let C_h be the monetary units cost per block in the system and unit time. The optimum value of Q is:*

$$Q^* = \sqrt{\frac{2C_k \lambda(1-\rho)}{C_h}} \tag{15}$$

Proof. The following notation will be used to demonstrate the theorem:

- $P_n = P(N = n)$
- $P_n(1) = P(N = n)$ and there is a server
- $P_n(0) = P(N = n)$ and there is not a server

It's easy to prove that $P_n = P_n(0) + P_n(1)$.

In the stationary state (from here on, we already assume $\rho < 1$) as a result we obtain the following equations, called equilibrium equations:

Top equation:

$$\begin{aligned} &(n = 0) && \mu P_1(1) = \lambda P_0(0) \\ &(n = 1) && \lambda P_0(0) = \lambda P_1(0) \\ &(n = 2) && \lambda P_1(0) = \lambda P_2(0) \\ & && \vdots \\ &(n = Q - 1) && \lambda P_{Q-2}(0) = \lambda P_{Q-1}(0) \end{aligned} \tag{16}$$

Bottom equation:

$$\begin{aligned} &(n = Q) && \lambda P_{Q-1}(1) + \lambda P_{Q-1}(0) + \\ & && + \mu P_{Q+1}(1) = \\ & && \lambda P_Q(1) + \mu P_Q(1) \\ &(n = 1) && (\lambda + \mu) P_1(1) = \mu P_2(1) \\ &(n \geq 2, n \neq Q) && \lambda P_{n-1}(1) + \mu P_{n+1}(1) = \\ & && = (\lambda + \mu) P_n(1) \end{aligned} \tag{17}$$

Merging both systems of equations:

$$\begin{aligned} \lambda P_0(0) &= \lambda P_1(0) = \cdots = \lambda P_{Q-1}(0) \\ \mu P_1(1) &= \lambda P_0(0) \\ \lambda P_{Q-1}(1) &+ \lambda P_{Q-1}(0) + \mu P_{Q+1}(1) = (\lambda + \mu) P_Q(1) \\ \mu P_2(1) &= (\lambda + \mu) P_1(1) \end{aligned}$$

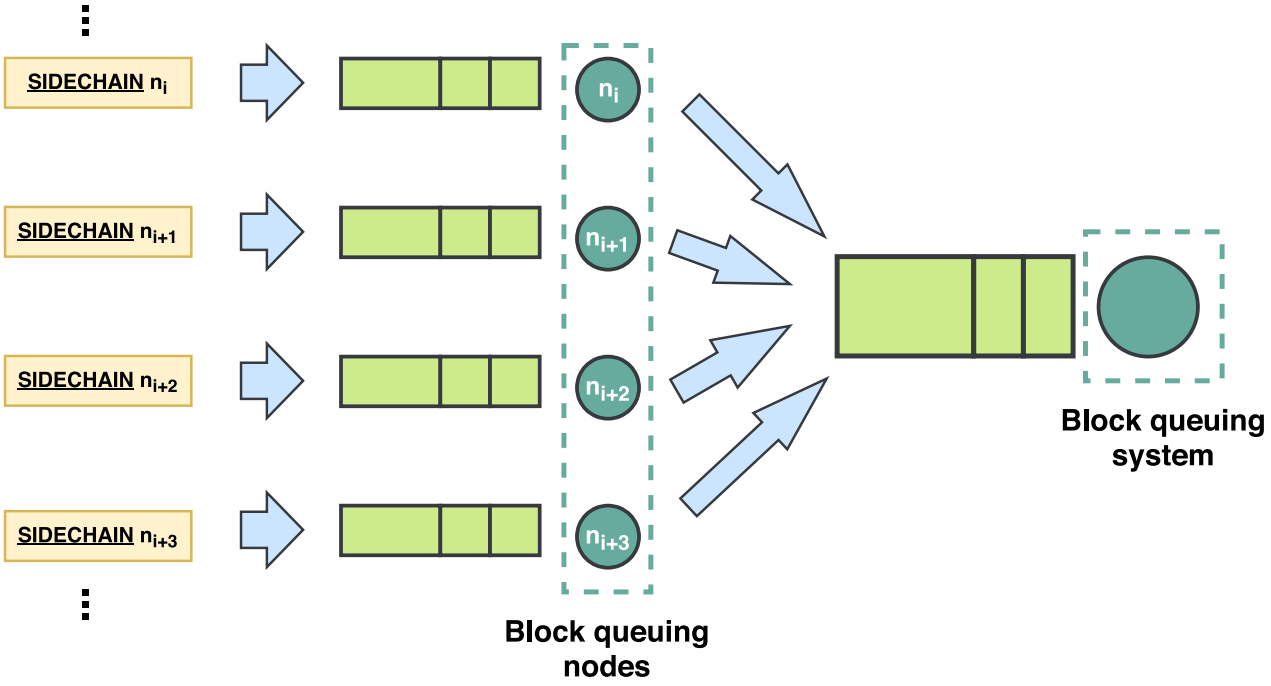

Fig. 4. Block queuing system scheme. Every single sidechain has its block queuing node. In this way, every block queuing node sends its blocks to the block queuing system.

$$\lambda P_{n-1}(1) \quad +\mu P_{n+1}(1) = (\lambda+\mu)P_n(1), \quad n \geq 2, n \neq Q \tag{18}$$

To solve this system, we will use the probability generator function $G(s)$ associated with the random variable N:

Let $|s| \leq 1$.

$$G(s) = \sum_{n=0}^{+\infty} P_n s^n \quad \left(= \sum_{n=0}^{Q-1} P_n(0)s^n + \sum_{n=1}^{+\infty} P_n(1)s^n \right) \tag{19}$$

By defining

$$G_0(s) = \sum_{n=0}^{Q-1} P_n(0)s^n \qquad G_1(s) = \sum_{n=1}^{+\infty} P_n(1)s^n \tag{20}$$

Then

$$G(s) = G0(s) + G1(s) \tag{21}$$

Moreover

$$G'(s) = \sum_{n=1}^{+\infty} nP_n s^{n-1} \tag{22}$$

Thus

$$G(1) = 1 \qquad G'(1) = E(N) \tag{23}$$

If we multiply by s^{n+1} the nth equation obtained in the system:

$$\begin{aligned} &s\mu P_1(1) + \lambda \sum_{n=2}^{+\infty} P_{n-1}(1)s^{n+1} \lambda P_0(0) + \mu \sum_{n=1}^{+\infty} P_{n+1}s^{n+1} \\ &= (\lambda+\mu) \sum_{n=1}^{+\infty} P_n(1)s^{n+1} + \lambda P_0(s) \\ &\Rightarrow \lambda s^2 G_1(s) + \lambda s^{Q+1} P_0(0) + \mu G_1(s) \\ &= (\lambda+\mu)sG_1(s) + \lambda s P_0(0) \Rightarrow \\ &G_1(s) = \frac{\rho s}{1-\rho s}(1 + s + \cdots + s^{Q-1})P_0(0). \end{aligned} \tag{24}$$

On the other hand,

$$G_0(s) = \sum_{n=0}^{Q-1} s^n P_n(0) = \sum_{n=0}^{Q-1} s^n P_0(0) = (1 + s + \cdots + s^{Q-1})P_0(0) \tag{25}$$

Thus, by evaluating $G(s)$ in $s = 1$ you have to

$$P_0(0) = \frac{1-\rho}{Q} \tag{26}$$

So

$$G(s) = G_0(s) + G_1(s) = \cdots = \frac{1 + s + \cdots + s^{Q-1}}{Q} \frac{1-\rho}{1-\rho s} \tag{27}$$

Now we should calculate $E(N)$ and $E(T_0 + T_1)$

$$G'(1) = \frac{Q-1}{2} + \frac{\rho}{1-\rho} = E(N) \tag{28}$$

$E(T_0)$ = average time for the system to proceed from 0 to Q clients. Since the time between arrivals follows a $Exp(\lambda)$ and this distribution has memory loss, then it is equivalent to calculating Q times, the average time for a customer to arrive.

$$E(T_0) = QE[e^{\lambda}] = Q\ \frac{1}{\lambda} = \frac{Q}{\lambda} \tag{29}$$

$\frac{E(T_0)}{E(T_0+T_1)}$ = failure probability that there are $0.1, \ldots, Q-1$ clients in the system in steady state and that there is no server

$$\begin{aligned} &= P_0(0) + P_1(0) + \cdots + P_{Q-1}(0) = QP_0(0) \\ &E(T_0 + T_1) = \frac{E(T_0)}{QP_0(0)} = \frac{Q\frac{1}{\lambda}}{QP_0(0)} = \frac{1}{\lambda\frac{1-e}{Q}} = \frac{Q}{\lambda(1-\rho)} \end{aligned} \tag{30}$$

So

$$\begin{aligned} f(Q) = \quad & C_h E(N) + \frac{C_k}{E(T_0+T_1)} = \\ & C_h\left(\frac{Q-1}{2} + \frac{\rho}{1-\rho}\right) + \frac{C_k\lambda(1-\rho)}{Q}, \qquad Q = 1, 2, 3, \cdots \end{aligned} \tag{31}$$

We look for the value where f reaches its minimum; if Q takes real values

$$\begin{aligned} f'(Q) = \quad & \frac{C_h}{2} - \frac{C_k\lambda(1-\rho)}{Q^2} = 0 \Rightarrow \\ \tilde{Q} = \quad & \sqrt{\frac{2C_k\lambda(1-\rho)}{C_h}} \end{aligned} \tag{32}$$

Since

$$f''(\tilde{Q}) = \frac{3C_h}{2}\sqrt{\frac{C_h}{2C_k\lambda(1-\rho)}} > 0 \tag{33}$$

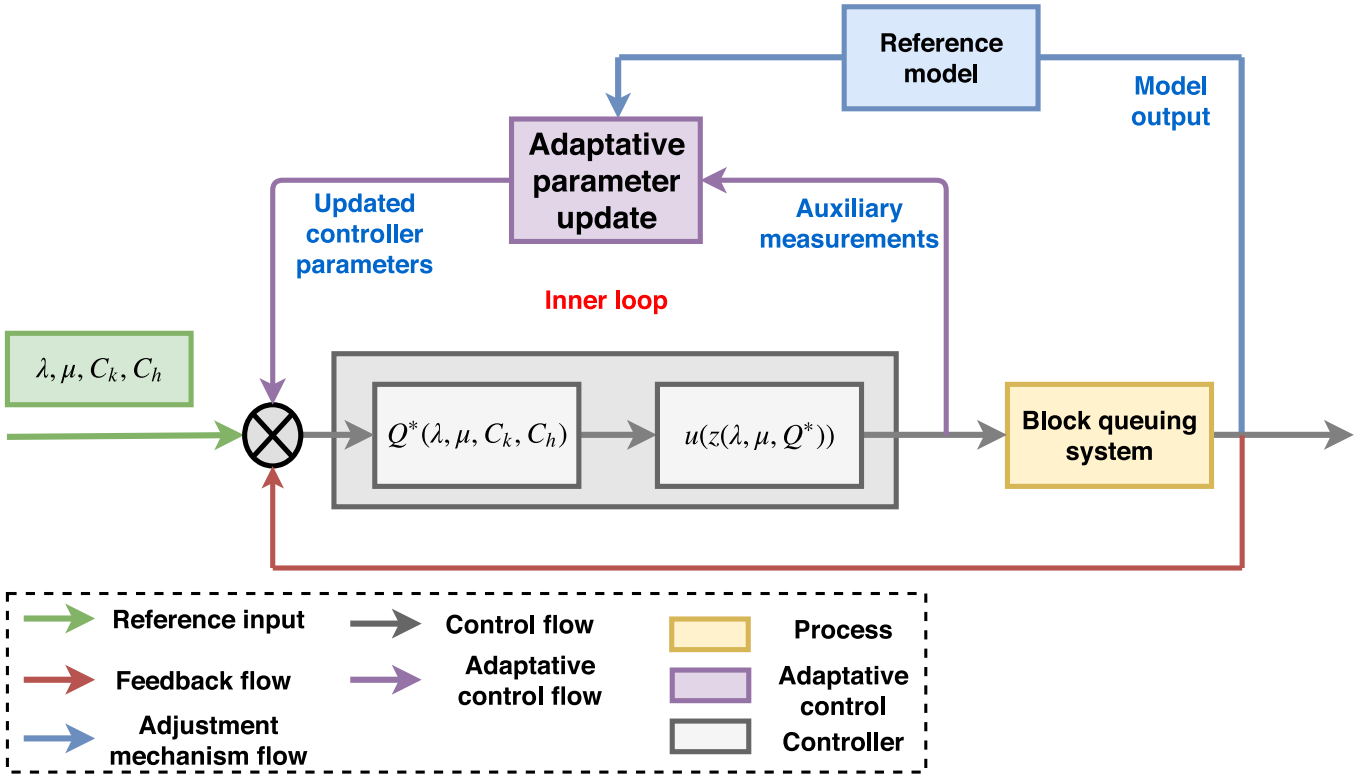

Fig. 5. Auto-adaptive control algorithm. This algorithm can correct the error in the parameters via the auto-adaptive parameter update function and the reference model (feedback function).

Then $\tilde{Q}$ would be local and global minimum.

As Q only takes natural values and the function f is convex, then the optimum solution Q^* will be the one that gives the minimum between $f([Q^*])$ and $f([Q^*]+1)$. Then,

$$Q^* = \sqrt{\frac{2C_k\lambda(1-\rho)}{C_h}} \tag{34}$$

As we wanted to prove.

□

3.2.3. Queuing control algorithm

Here we describe how the adaptive control algorithm works. This algorithm is used by the block queuing system to monitor and control the flow of blocks from the queue network to the miners' network. In Fig. 5, the set point (green arrow) with the reference input are the following variables: 1) λ. It is the average number of blocks per unit of time that reach the block queuing system. Each step of the algorithm per time unit, λ is introduced into the control algorithm to update this parameter at each time unit. 2) μ. It is the average number of blocks that the queuing system is able to manage. This parameter enters the flow in each of the steps of the algorithm. However, the adaptive parameters update controls this variable for the optimal performance of the block queuing system. 3) C_k. It is the cost of monetary units each time the server reinitiates its work. 4) C_h. It is the cost of monetary units per block in the miner's network and time unit. Parameters 3 and 4 are calculated considering all the costs associated with each of them.

The adaptive control system controller is composed of the control functions $Q^*(\lambda, \mu, C_k, C_h)$ and $u(z(\lambda, \rho, Q^*))$. The Q^* function estimates the optimal number of blocks for the block queuing system to work on optimizing the energy consumption. We assume that $Q^* = 0$ if $\rho \geq 1$. This is because if $\rho \geq 1$ the Q^* value have a negative square root and in this step of our research we are working with the Real field. Furthermore, the u function decides if μ of the block queuing system changes its value.

The Q^* function is defined using the result obtained from the Theorem 3. We define an auxiliary function z as follows:

$$z(\lambda, \rho, Q^*) = \begin{cases} -1 & if \quad \rho \geq 1 \\ 0 & if \quad \lambda \leq Q^* \& \rho < 1 \\ 1 & if \quad \lambda > Q^* \& \rho < 1 \end{cases} \tag{35}$$

Then, if $z(\lambda, \rho, Q^*) = 0$, the block control algorithm reduces the number of blocks (μ) that the block queuing system sends to the miners' network. Otherwise, if $z(\lambda, \rho, Q^*) = 1$ the algorithm enables the block queuing system to send blocks into the miners' network. If $z(\lambda, \rho, Q^*) = -1$, the queue is saturated, then the controller does not change the value of μ. The u function is defined by using the auxiliary function z. The value of the parameter $\mu_{controller}$ (μ_c) is given by the following equation:

$$u(z(\lambda, \rho, Q^*)) = \begin{cases} \mu & if \quad u = \pm 1 \\ \frac{1}{2}(Q^* + \lambda) & if \quad u = 0 \\ \mu_a & if \quad \mu \in \text{Inner loop} \end{cases} \tag{36}$$

Once the controller has sent the control signal to the block queuing system, it changes the value of μ according to the controller. This is the control flow of the algorithm. Moreover, Fig. 5 shows the reference model flow (blue flow), which sends the parameter information μ to the set point which passes it to the controller. In this way, the algorithm receives the reference model information of the process status and confirms whether the queue network collapses or not ($\rho < 1$). The reference model uses the reference function (e) to determine whether the adaptive control algorithm is working properly. Let $\mu_{reference} = \mu_f = w(\rho, Q^*, \lambda)$ be the parameter μ sent by the reference model to the adaptive parameter update. Then, e is defined in the following equation:

$$w(\rho, Q^*, \lambda, \mu_c) = \begin{cases} \frac{(1+\rho^5)}{\pi}(Q^* + \lambda + \mu_c) & if \quad \rho < 1 \\ \frac{1}{5}\lambda\rho^{\frac{7}{3}}e^{\rho-\pi} & if \quad 1 \leq \rho < \frac{5}{3} \\ \frac{1}{5}\lambda\rho^{2e}e^{\rho-(\frac{e\pi}{2})} & if \quad \rho \geq \frac{5}{3} \end{cases} \tag{37}$$

The adaptive parameter update module sends the auxiliary measurements (μ_c) that come out of the controller to the set point, where the value of the μ parameter is changed, so that the controller has the new value of $\mu_{adaptive} = \mu_a = \theta(\rho)$ in the next step of the algorithm. Then, θ is defined as follows:

$$\theta(\rho, \mu_r, \mu_c, Q^*, \lambda) = \begin{cases} \frac{1}{2}(\mu_c + \mu_r) & if \quad \rho < 1 \& \lambda > Q^* \\ \frac{9}{10\rho^{2\pi e}}(\mu_r + \lambda) & if \quad 1 \leq \rho < \frac{5}{3} \\ \frac{(1+\rho)}{\rho^4}(\mu_r + \lambda) & if \quad \rho \geq \frac{5}{3} \end{cases} \tag{38}$$

Hence, the updated controller parameter μ_a is sent to the set point where it is used in one of the steps of the adaptive control algorithm. Also, this updated controller parameter μ_a is sent via the controller to the block queuing system as the new auto-adapt update μ_c. Since the controller system detects that the μ_a comes from the inner loop (i.e., adaptive function), it sends the updated parameter to the block queuing system. Thus, the final signal sent by the controller is $\mu_c = \mu_a$.

3.2.4. Hashmap control system

Data are streamed and stored in the blockchain in real time. Once in blockchain, data are sent to the big data layer. In this layer, the data are stored in HDFS. Once data are stored with the Sensor ID and timestamp primary keys, a smart contract is triggered to create a query (e.g., query via Hive), providing access to those data. The smart contract sends this query, the sensor ID and the timestamp to the "Hashmap control system". This control system stores that data in a hashmap within the blockchain. This minimizes the time required to search through large amounts of data. Although it is not the main objective of this manuscript, we have considered it appropriate to include this system for faster data search. In this way, IoT nodes monitoring and control is ore effective.

3.3. Big data layer

As described above, the blockchain layer can use the big data layer in real time with bidirectional data streaming. Although the big data layer is essential for the overall functioning of the system, it is not viewed as an important part of this article as it does not represent the main novelty of the work. For this reason, it will not be explained in great detail, although it is necessary to understand that four different blocks can be found in the big data layer: i) Event processing, which is in charge of providing in real time the computing capacity required to respond to the different requests received by the layer; ii) Operational analysis; iii) Data storage, which will generally follow NoSQL models for better performance, although other classic models can also be used; iv) Historical analysis, to extract knowledge from data stored up to the moment of the analysis request by using machine learning algorithms in combination with the business intelligence (BI) of the particular use case.

4. Simulation results

In this section, two simulation cases are considered to illustrate the effectiveness of the novel adaptive control algorithm and the faster data search using hashmap.

4.1. Simulation 1: Queuing adaptive control algorithm

The example given in this subsection shows the functioning of the network with the queuing model and the adaptive control algorithm proposed in this manuscript. The model is an open network of self-adaptive Jackson, in which each of the queuing block queues is a $M|M|1$ queue that sends its block to the block queuing system.

Let's assume the value of some of the parameters that we will use in this example:

- Let's assume that, on average, the value of $C_k = 30\ C_h$. The value of these parameters was calculated from the mean value of the experimental data.
- The time unit t is 1 h.
- Number of block queuing nodes = 4 and one block queuing system.
- $\gamma_i \in$ random value between $\{1, \cdots, 10\}$ where $i \in$ {set of the block queuing nodes}.

Step 1

We randomly create the following values:

- γ_i, which are: $\gamma_1 = 8$, $\gamma_2 = 4$, $\gamma_3 = 6$ and $\gamma_4 = 2$.
- $\mu_i = 10 \quad \forall i \in \{1, \cdots, 4\}$ and $\mu_5 = 30$

Thus, the average value of the blocks that enter the block queuing node is:

$$\lambda = \sum_i^4 \gamma_i = 8 + 4 + 6 + 2 = 20 \tag{39}$$

Hence:

$$\rho_{miners} = \frac{\lambda}{\mu_c} = \frac{20}{30} = \frac{2}{3} \tag{40}$$

Next we have simulated the first step of the algorithm:

1. Reference input of the set points are: $\lambda = 20$, $\mu_c = 30$, C_h, $C_k = 16C_h$
2. Compute the value of the parameter Q^*:

$$Q^* = \sqrt{\frac{2C_k\lambda(1-\rho)}{C_h}} = 20 \tag{41}$$

3. Compute the value of u:

$$u(z(\lambda, \rho, Q^*)) = 0 \tag{42}$$

4. Output values:
   (a) $\mu_{controller} = 20$
   (b) $w(\rho, Q^*, \lambda, \mu_c) = \mu_{reference} = 24$
   (c) $\mu_{adaptive} = 22$
5. Hence, the new input value of $\mu_{controller} = 22$ for the next step.

Step 2

We randomly create the following values:

- γ_i, which are: $\gamma_1 = 6$, $\gamma_2 = 9$, $\gamma_3 = 10$ and $\gamma_4 = 7$
- $\mu_i = 10 \quad \forall i \in \{1, \cdots, 4\}$ and $\mu_c = 22$

Thus, the average value of the blocks that enter the block queuing node is:

$$\lambda = \sum_i^4 \gamma_i = 6 + 9 + 10 + 7 = 32 \tag{43}$$

hence:

$$\rho_{miners} = \frac{\lambda}{\mu_c} = \frac{32}{22} \tag{44}$$

Next, we simulated the second step of the algorithm:

1. Reference input of the set points are: $\lambda = 32$, $\mu_5 = 25$, C_h, $C_k = 16C_h$.
2. Compute the value of the parameter Q^*, but $\rho = \frac{32}{22} > 1$, then $Q^* = 0$.
3. Compute the value of u:

$$u(z(\lambda, \rho, Q^*)) = -1 \tag{45}$$

4. Output values:
   (a) $\mu_{controller} = 22$
   (b) $w(\rho, Q^*, \lambda, \mu_c) = \mu_{reference} = 29$
   (c) $\mu_{adaptive} = \theta(\rho, \mu_{controller}) = 33$
5. Hence, the new input value of $\mu_{controller} = 33$ for the next step.

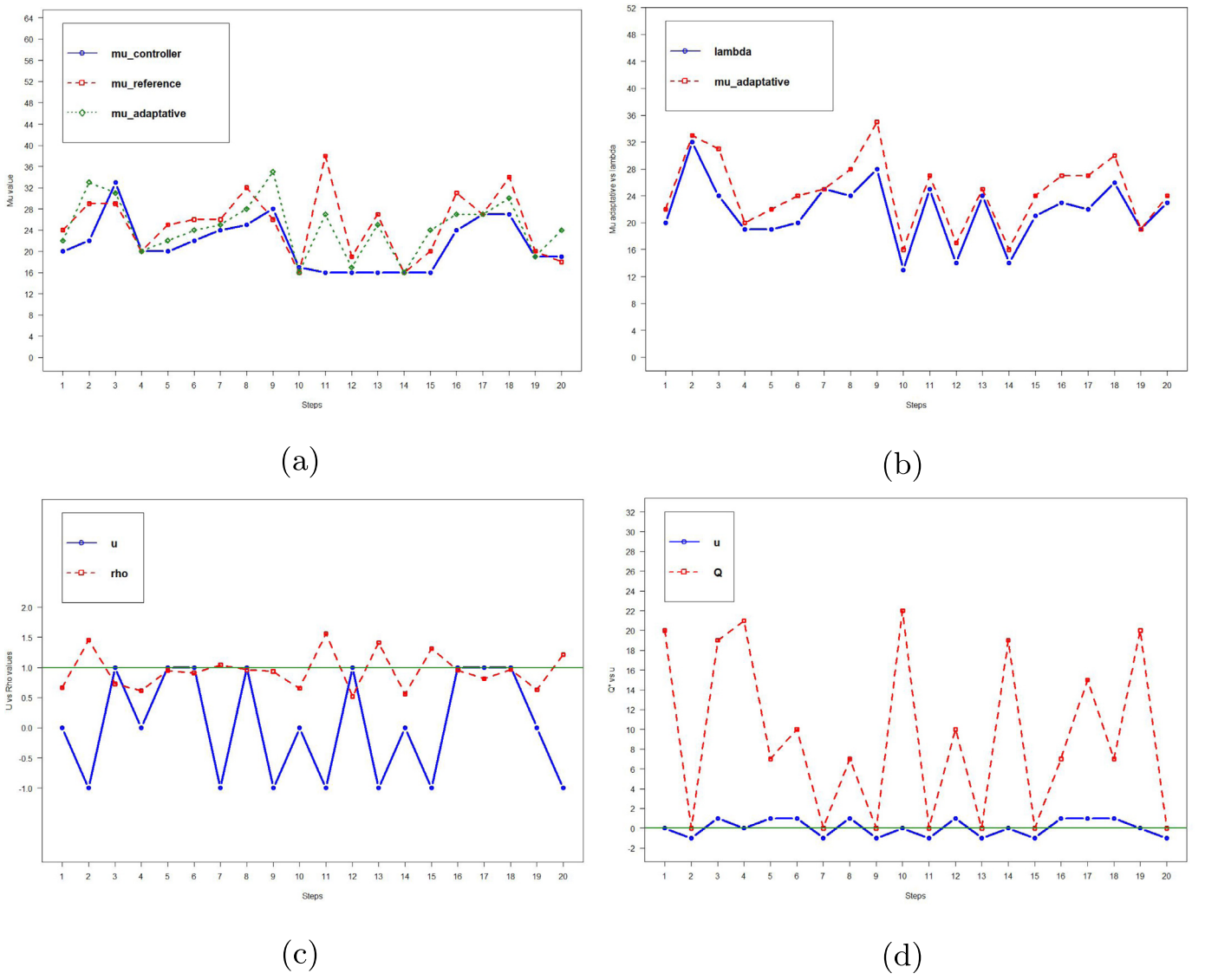

Fig. 6. Adaptive control algorithm measurements. (a) The $\mu_{controller}$ (solid blue line), the $\mu_{reference}$ (dashed red line) and the $\mu_{adaptive}$ (dashed green line) generated by simulation 1. In this figure we can find the $\mu_{controller}$ and $\mu_{reference}$ errors and how the $\mu_{adaptive}$ self-correct these errors. (b) The $\lambda_{miners'\ network}$ (solid blue line) and the $\mu_{adaptive}$ (dashed red line) generated by simulation 1. In the figure we can find how the $\mu_{adaptive} \geq \lambda_{miners'\ network}$ in all the steps of the control adaptive algorithm. (c) The controller function u values (solid blue line) and the $\rho_{miners'\ network}$ (dashed red line) generated by simulation 1. In this figure we can find a relationship between u and $\rho_{miners'\ network}$. In this way, once $\rho_{miners'\ network} \geq 1$ then $u = -1$. The saturated bound (solid green line) shows the top bound for the system to not be saturated. (d) The u controller function value (solid blue line) and the Q^* value (dashed red line) generated by simulation 1. Since $u = -1$ then $Q^* = 0$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Step 3

We randomly create the following values:

- γ_i, which are: $\gamma_1 = 4$, $\gamma_2 = 8$, $\gamma_3 = 7$ and $\gamma_4 = 5$
- $\mu_i = 10 \quad \forall i \in \{1, \cdots, 4\}$ and $\mu_c = 33$

Thus, the average value of the blocks that enter the block queuing node is:

$$\lambda = \sum_{i}^{4} \gamma_i = 4 + 8 + 7 + 5 = 24 \tag{46}$$

Hence:

$$\rho_{miners} = \frac{\lambda}{\mu_c} = \frac{24}{33} \tag{47}$$

Then, we simulated the third step of the algorithm:

1. Reference input of the set points are: $\lambda = 20$, $\mu_5 = 640$, C_h, $C_k = 16C_h$
2. Compute the value of parameter Q^*:

$$Q^* = \sqrt{\frac{2C_k\lambda(1-\rho)}{C_h}} = 19 \tag{48}$$

3. Compute the value of u:

$$u(z(\lambda, \rho, Q^*)) = 1 \tag{49}$$

4. Since $u(\lambda, Q^*) = 0$ the block queuing system does not send blocks to the miners' network.
5. Output values:
   (a) $\mu_{controller} = 33$
   (b) $w(\rho, Q^*, \lambda, \mu_c) = \mu_{reference} = 29$
   (c) $\mu_{adaptive} = \theta(\rho, \mu_{controller}) = 31$
6. Hence, the new input value of $\mu_{controller} = 31$ for the next step.

Step 4

We randomly create the following values:

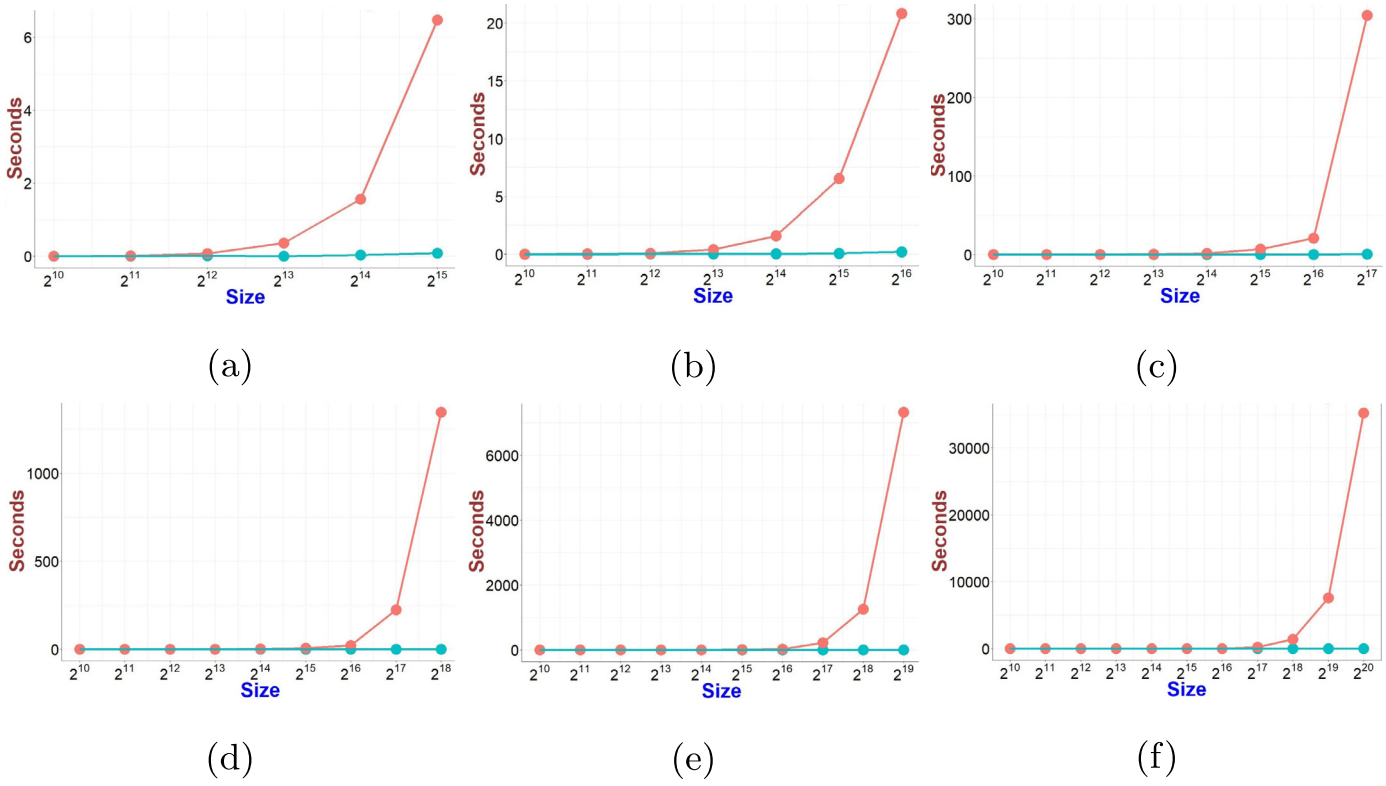

Fig. 7. Data search speed up using hashmaps. The time (in seconds) of the search process using hashmaps (solid blue line) and the time (in seconds) of the search process with big data technologies (solid red line) generated by the simulation with different size values. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- γ_i, which are: $\gamma_1 = 5$, $\gamma_2 = 4$, $\gamma_3 = 9$ and $\gamma_4 = 1$
- $\mu_i = 10 \quad \forall i \in \{1, \cdots, 4\}$ and $\mu_c = 31$

Thus, the average value of the blocks that enter the block queuing node is:

$$\lambda = \sum_{i}^{4} \gamma_i = 5 + 4 + 9 + 1 = 19 \tag{50}$$

Hence:

$$\rho_{miners} = \frac{\lambda}{\mu_c} = \frac{19}{31} \rightarrow \infty \tag{51}$$

Then, we simulated the fourth step of the algorithm:

1. Reference input of the set points are: $\lambda = 20$, $\mu_c = 31$, C_h, $C_k = 16C_h$
2. Compute the value of parameter Q^*:

$$Q^* = \sqrt{\frac{2C_k\lambda(1-\rho)}{C_h}} = 21 \tag{52}$$

3. Compute the value of u:

$$u(z(\lambda, \rho, Q^*)) = 0 \tag{53}$$

4. Output values:
   (a) $\mu_{controller} = 20$
   (b) $w(\rho, Q^*, \lambda, \mu_c) = \mu_{reference} = 20$
   (c) $\mu_{adaptive} = \theta(\rho, \mu_{controller}) = 20$
5. Hence, the new input value of $\mu_5 = \mu_{adaptive} = 20$

The simulation of the adaptive control algorithm is shown in Table 1. This table shows the results of the simulations of the adaptive control algorithm for 20 stages with 4 IoT nodes that randomly send blocks to the block queuing system.

Values taken by the three different $\mu_{controller}$, $\mu_{reference}$ and $\mu_{adtative}$ can be found in Fig. 6a. The $\mu_{controller}$ parameter is the signal that the controller sends to the block queuing system (process), while the $\mu_{reference}$ parameter receives the signal that comes out of the process and sends it as feedback to the adaptive control. In this way it can self-correct the error in these parameters. In this figure we can find the fact that the parameters μ that represent the capacity of miners are always equal or greater than the entry of blocks in the queue (i.e., λ). This is important because it is proof that the control algorithm prevents the queue from collapsing, and if this happens, the algorithm makes the queue work correctly again in the next step of the algorithm. Fig. 6b verifies that the $\mu_{adaptive}$ is lower bounded by the value of λ. So, the μ of the process will always be higher than the λ. Our findings suggest that the $\mu_{process} = \mu_{adaptive}$ self-adapts to the $\mu_{process}$ allowing the block queuing system to recover from saturated conditions, and otherwise, the $\mu_{process}$ self-adapts to the $\mu_{process}$ in order to ensure that $\rho < 1$. This way we check that the control algorithm works correctly. This is because the capacity of the server is always greater than the number of blocks that enter the queue.

Fig. 6c shows the parameters ρ and the value of the controller function u. This figure shows that both variables are closely related. So, when the variable $\rho \geq 1$, $u = 1$ (i.e., if the dashed red line is over the green line, the system is saturated). In other words, the controller informs the system that once the block queuing system is saturated, it has to auto-adapt the system's ρ to overcome this state. This is achieved by increasing the value of $\mu_{process}$, as $\rho = \frac{\lambda}{\mu}$. Thus, if $\mu > \lambda$ the system is no longer saturated and returns to its stationary state. This graph shows how the control algorithm reacts when the tail collapses. This prevents the miners' network from collapsing and avoids delays in the monitoring and control of the IoT network. In addition, Fig. 6d shows the behavior of the Q^* parameter in relation to u. Since u is directly related to ρ as presented in Fig. 6c, Q^* is also directly related to ρ. Eleven $u = -1$ ($\rho \geq 1$), the controller detects that the system is saturated and therefore the optimal value of the blocks that have to be on hold for the miners' network to mine blocks is zero (i.e., $Q^* = 0$).

Table 1
Result of the 20 step numerical simulation demonstrating how the adaptive control algorithm works.

Step No	$\gamma_{1,2,3,4}$	λ_5	ρ	Q^*	$u(\lambda, \mu, Q^*)$	$\mu_{controller}$	$\mu_{reference}$	$\mu_{adaptive}$
1	{8, 4, 6, 2}	20	$\frac{2}{3}$	20	0	20	24	22
2	{6, 9, 10, 7}	32	$\frac{32}{22}$	0	−1	22	29	33
3	{4, 8, 7, 5}	24	$\frac{24}{19}$	19	1	33	29	31
4	{5, 4, 9, 1}	19	$\frac{19}{21}$	21	0	20	20	20
5	{7, 5, 3, 4}	19	$\frac{19}{7}$	7	1	20	25	22
6	{5, 4, 8, 3}	20	$\frac{20}{10}$	10	1	22	26	24
7	{8, 6, 9, 2}	25	$\frac{25}{0}$	0	−1	24	26	25
8	{10, 7, 1, 6}	24	$\frac{24}{7}$	7	1	25	32	28
9	{9, 8, 6, 5}	28	$\frac{28}{0}$	0	−1	28	26	35
10	{6, 4, 1, 2}	13	$\frac{13}{22}$	22	0	17	16	16
11	{5, 9, 1, 10}	25	$\frac{25}{0}$	0	−1	16	38	27
12	{2, 5, 3, 4}	14	$\frac{14}{10}$	10	1	16	19	17
13	{6, 3, 5, 10}	24	$\frac{24}{0}$	0	−1	16	27	25
14	{5, 6, 2, 1}	14	$\frac{14}{19}$	19	0	16	16	16
15	{8, 5, 7, 1}	21	$\frac{21}{0}$	0	−1	16	20	24
16	{7, 5, 10, 1}	23	$\frac{23}{7}$	7	1	24	31	27
17	{5, 4, 3, 10}	22	$\frac{22}{15}$	15	1	27	27	27
18	{4, 7, 6, 9}	26	$\frac{26}{7}$	7	1	27	34	30
19	{3, 4, 2, 10}	19	$\frac{19}{20}$	20	0	19	20	19
20	{9, 6, 5, 3}	23	$\frac{23}{0}$	0	−1	19	18	24

4.2. Simulation 2: Hashmap search

This simulation has been done using HDFS to store the data of the IoT nodes. The objective of this simulation is to validate that it takes less time to search the data using a hashmap than it is in HDFS. For the HDFS simulation, the data of IoT nodes were randomly generated and stored in HDFS. While for the hashmap simulation, the sensor ID, timestamp and query were stored in the hashmap for subsequent data search in the HDFS. In this way, it is easy to monitor and control IoT nodes, by locating the data through their sensor ID field, or their timestamp field. The search via hashmap is very fast (its computational complexity is $O(1)$). The different simulations that have been generated are shown in Fig. 7. The size of the data has been increased to see how both search systems behave. In all the simulations, search via hashmap proved to be faster.

5. Conclusion and future work

This paper has addressed the block control flow problem between IoT devices and blockchain. By using queuing theory, adaptive controller is developed to achieve the optimal block number to improve the mining process efficiency. In this way, stability of the miners' network is improved since the adaptive controller ensures that the network is not saturated. On the other hand, a new model to speed up the searches by using hashmaps is proposed in this paper. Thus, a new architecture to merge these new contributions is proposed to improve IoT platforms. The effectiveness of the proposed approach is supported by simulations. The extensions to systems with more general structure are an interesting topic for future research.

Acknowledgment

This work was developed as part of "Virtual-Ledgers-Tecnologías DLT/Blockchain y Cripto-IOT sobre organizaciones virtuales de agentes ligeros y su aplicación en la eficiencia en el transporte de última milla", ID SA267P18, project cofinanced by Junta Castilla y León, Consejería de Educación, and FEDER funds.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.inffus.2018.12.007

References

[1] X.-M. Zhang, Q.-L. Han, X. Yu, Survey on recent advances in networked control systems, IEEE Trans. Ind. Inf. 12 (5) (2016) 1740–1752.
[2] X. Ge, F. Yang, Q.-L. Han, Distributed networked control systems: a brief overview, Inf Sci 380 (2017) 117–131.
[3] J. Qiu, H. Gao, M.-Y. Chow, Networked control and industrial applications [special section introduction], IEEE Trans. Ind. Electron. 63 (2) (2016) 1203–1206.
[4] J.P. Hespanha, P. Naghshtabrizi, Y. Xu, A survey of recent results in networked control systems, Proc. IEEE 95 (1) (2007) 138–162.
[5] W. Sun, Z. Zeng, C. Luo, S.-K. Nguang, Robust-gain codesign of networked control systems, Int. J. Robust Nonlinear Control (2018) 1–15.
[6] K. Liu, A. Seuret, E. Fridman, Y. Xia, Improved stability conditions for discrete-time systems under dynamic network protocols, Int. J. Robust Nonlinear Control (2018) 1–21.
[7] R. Casado-Vara, Z. Vale, J. Prieto, J. Corchado, Fault-tolerant temperature control algorithm for iot networks in smart buildings, Energies 11 (12) (2018) 3430.
[8] C. Tan, L. Li, H. Zhang, Stabilization of networked control systems with both network-induced delay and packet dropout, Automatica 59 (2015) 194–199.
[9] Z. Zhou, H. Zhang, S. Li, X. Du, Hermes: a privacy-preserving approximate search framework for big data, IEEE Access 6 (2018) 20009–20020.
[10] Y. Cao, H. Qi, W. Zhou, J. Kato, K. Li, X. Liu, J. Gui, Binary hashing for approximate nearest neighbor search on big data: a survey, IEEE Access 6 (2018) 2039–2054.
[11] J. Bremer, S. Lehnhoff, Decentralized coalition formation with agent-based combinatorial heuristics, ADCAIJ 6 (3) (2017) 29–44.
[12] S. Nakamoto, A peer-to-peer electronic cash system, 2008, (https://bitcoin.org/bitcoin.pdf). 2018-08-30.
[13] B. Wiki, Irreversible transactions, 2012, (https://en.bitcoin.it/wiki/Irreversible_Transactions). 2018-08-30.
[14] A.M. Antonopoulos, Mastering Bitcoin: Unlocking Digital Cryptocurrencies, " O'Reilly Media, Inc", 2014.
[15] R.C. Cardoso, R.H. Bordini, A multi-agent extension of a hierarchical task network planning formalism, ADCAIJ 6 (2) (2017) 5–17.
[16] K.-H.N. Bui, J.J. Jung, D. Camacho, Consensual negotiation-based decision making for connected appliances in smart home management systems, Sensors (Basel) 18 (7) (2018).
[17] G. Capellari, E. Chatzi, S. Mariani, Cost–benefit optimization of structural health monitoring sensor networks, Sensors (Basel) 18 (7) (2018).
[18] R. Casado-Vara, J.M. Corchado, Blockchain for democratic voting: how blockchain could cast off voter fraud, Orient. J. Comp. Sci. and Technol 11 (1) (2018).
[19] A. Monteriù, M.R. Prist, E. Frontoni, S. Longhi, F. Pietroni, S. Casaccia, L. Scalise, A. Cenci, L. Romeo, R. Berta, et al., A smart sensing architecture for domestic monitoring: methodological approach and experimental validation, Sensors (Basel) 18 (7) (2018).
[20] B. Wiki, Hashcash, 2013, (https://en.bitcoin.it/wiki/Hashcash). 2018-08-30.
[21] C. Pop, T. Cioara, M. Antal, I. Anghel, I. Salomie, M. Bertoncini, Blockchain based decentralized management of demand response programs in smart energy grids, Sensors 18 (1) (2018) 162.
[22] P.A.R. Marín, N. Duque, D. Ovalle, Multi-agent system for knowledge-based recommendation of learning objects, ADCAIJ 4 (1) (2015) 80–89.
[23] L. Becerra-Bonache, M.D.J. López, Linguistic models at the crossroads of agents, learning and formal languages, ADCAIJ 3 (4) (2014) 67–87.
[24] J. Medhi, Stochastic models in queueing theory, Elsevier, 2002.

[25] S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, L. Veltri, A scalable and self-configuring architecture for service discovery in the internet of things, IEEE Internet Things J. 1 (5) (2014) 508–521.
[26] P. Chamoso, A. González-Briones, S. Rodríguez, J.M. Corchado, Tendencies of technologies and platforms in smart cities: a state-of-the-art review, Wireless Commun. Mobile Comput. 2018 (2018).
[27] R. Mahmoud, T. Yousuf, F. Aloul, I. Zualkernan, Internet of things (iot) security: current status, challenges and prospective measures, in: Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for, IEEE, 2015, pp. 336–341.
[28] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, N. Venkatasubramanian, A software defined networking architecture for the internet-of-things, in: Network Operations and Management Symposium (NOMS), 2014 IEEE, IEEE, 2014, pp. 1–9.
[29] R. Casado-Vara, F. Prieto-Castrillo, J.M. Corchado, A game theory approach for cooperative control to improve data quality and false data detection in wsn, Int. J. Robust Nonlinear Control (2018), doi:10.1002/rnc.4306.
[30] D.P. Abreu, K. Velasquez, M. Curado, E. Monteiro, A resilient internet of things architecture for smart cities, Ann. Telecommun. 72 (1–2) (2017) 19–30.
[31] J. Matias, J. Garay, N. Toledo, J. Unzilla, E. Jacob, Toward an sdn-enabled nfv architecture, IEEE Commun. Mag. 53 (4) (2015) 187–193.
[32] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[33] J. Zhou, T. Leppanen, E. Harjula, M. Ylianttila, T. Ojala, C. Yu, H. Jin, L.T. Yang, Cloudthings: a common architecture for integrating the internet of things with cloud computing, in: Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2013, pp. 651–657, doi:10.1109/CSCWD.2013.6581037.
[34] H. Yue, L. Guo, R. Li, H. Asaeda, Y. Fang, Dataclouds: enabling community-based data-centric services over the internet of things, IEEE Internet Things J. 1 (5) (2014) 472–482, doi:10.1109/JIOT.2014.2353629.
[35] A. Panarello, N. Tapas, G. Merlino, F. Longo, A. Puliafito, Blockchain and IoT integration: a systematic survey, Sensors 18 (8) (2018) 2575.
[36] R. Casado-Vara, A. González-Briones, J. Prieto, J.M. Corchado, Smart contract for monitoring and control of logistics activities: Pharmaceutical utilities case study, in: The 13th International Conference on Soft Computing Models in Industrial and Environmental Applications, Springer, 2018, pp. 509–517.
[37] M. Conoscenti, A. Vetrò, J.C.D. Martin, 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), 2017, pp. 288–290, doi:10.1109/ICSE-C.2017.60.
[38] J. Herbert, A. Litchfield, A novel method for decentralised peer-to-peer software license validation using cryptocurrency blockchain technology, in: Proceedings of the 38th Australasian Computer Science Conference (ACSC 2015), 27, 2015, p. 30.
[39] C. Fromknecht, D. Velicanu, S. Yakoubov, Certcoin: a namecoin based decentralized authentication system, Technical Report, Massachusetts Institute of Technology, MA, USA. 6.857 Class Project, 2014.
[40] G. Caronni, Walking the web of trust, in: Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000), 2000, pp. 153–158, doi:10.1109/ENABL.2000.883720.
[41] D. Miller, Blockchain and the internet of things in the industrial sector, IT Prof. 20 (3) (2018) 15–18, doi:10.1109/MITP.2018.032501742.
[42] O. Novo, Blockchain meets iot: an architecture for scalable access management in iot, IEEE Internet Things J. (2018).
[43] B. Gnedenko, I. Kovalenko, Introduction to Queuing Theory. Mathematical Modeling, Birkhaeuser Boston, Boston, 1989.
[44] L. Kleinrock, Queueing Systems, volume 2: Computer Applications, 66, wiley New York, 1976.
[45] C.R. Rao, C.R. Rao, M. Statistiker, C.R. Rao, C.R. Rao, Linear Statistical Inference and Its Applications, 2, Wiley New York, 1973.
[46] B.S. Blanchard, W.J. Fabrycky, W.J. Fabrycky, Systems Engineering and Analysis, 4, Prentice Hall Englewood Cliffs, NJ, 1990.
[47] S. Fomundam, J.W. Herrmann, A survey of queuing theory applications in healthcare, Technical Report, 2007.
[48] G. Giambene, Queuing Theory and Telecommunications, Springer US, 2005.
[49] R. Srivastava, Exploration of in-memory computing for big data analytics using queuing theory, in: Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications, ACM, 2018, pp. 11–16.
[50] C.-S. Choi, F. Baccelli, G. de Veciana, Densification leveraging mobility: an iot architecture based on mesh networking and vehicles, in: Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2018, pp. 71–80.
[51] D. Zhai, R. Zhang, L. Cai, B. Li, Y. Jiang, Energy-efficient user scheduling and power allocation for noma based wireless networks with massive iot devices, IEEE Internet Things J. (2018).
[52] D.A. Chekired, L. Khoukhi, H.T. Mouftah, Industrial iot data scheduling based on hierarchical fog computing: a key for enabling smart factory, IEEE Trans. Ind. Inf. (2018).