Wallance, an Alternative to Blockchain for IoT

Loïc Dalmasso, Florent Bruguier, Achraf Lamlih, Pascal Benoit

Wallance, an Alternative to Blockchain for IoT

Loïc Dalmasso, Florent Bruguier, Achraf Lamlih, Pascal Benoit
LIRMM
University of Montpellier, CNRS
Montpellier, France
firstname.name@lirmm.fr

***Abstract*—Since the expansion of the Internet of Things (IoT), connected devices became smart and autonomous. Their exponentially increasing number and their use in many application domains result in a huge potential of cybersecurity threats. Taking into account the evolution of the IoT, security and interoperability are the main challenges, to ensure the reliability of the information. The blockchain technology provides a new approach to handle the trust in a decentralized network. However, current blockchain implementations cannot be used in the IoT domain because of their huge need for computing power and storage utilization. This paper provides a lightweight distributed ledger protocol dedicated to the IoT application, reducing the computing power and storage utilization, handling the scalability and ensuring the reliability of the information.**

Keywords—Blockchain, Distributed Ledger, Internet of Things, Decentralized network, Fog computing

I. Introduction

With the so-called “Internet of Things” (IoT), connected devices overwhelm our society in many areas such as smart cities, healthcare, automotive, environment, industries, etc [1-4]. Each node is a device such as a sensor or an actuator, able to digitalize and to interact with its environment and other nodes. A further extent of this concept is called the "Internet of Everything" (IoE), which means anything can be connected to the global network. Even if IoT/IoE can bring revolutionizing applications in many domains, there are still huge challenges to solve to fully turn it into reality. While the main challenges are interoperability, data storage, scalability, energy, and quality of service, security remains one of the most important. Since the aim of an IoT infrastructure is to connect everything on the network, this hyper-connectivity and the omnipresence of connected devices potentially represent a huge privacy and security threats.

IoT devices have limited memory capacity, computing power, and energy source compared to computers, servers, etc. Ensuring security in these very constrained devices without compromising performances is a big challenge. To overcome the IoT limitations, Cloud systems have hugely been deployed. However, it represents a single point of failure regarding security, privacy, and availability. Besides, real-time and energy efficiency are important features of IoT devices. With the expansion of the latter, Cloud systems have to handle the exponential increase of communication and volume of data [1], [3].

Nowadays, complex operations can be performed in so-called "Fog computing" such as a gateway, improving efficiency, latency, and scalability of the network. In this ecosystem, everything is connected and cooperates thanks to a decentralized Machine-to-Machine (M2M) communication protocol. A security mechanism has to ensure the trust between devices, securing all interactions without a central authority like Cloud. In this context, the blockchain system brings many benefits in terms of reliability and security. It avoids a single point of failure because of its data replication on multiple peers. There is no third party to trust and all validation processes are performed by the peers according to the consensus protocol.

Blockchain is a very promising mechanism to secure a decentralized network. Even though it has shown itself to be suitable for some application domains such as cryptocurrency [5-8], there are many constraints to adapt it to the IoT. By definition, blockchain needs a continually growing storage space, which is very limited in connected devices. Besides, blockchain cannot handle a huge transaction throughput, as expected with billions of IoT devices. In this paper, we propose a lightweight alternative to blockchain to secure interactions in an IoT infrastructure. Taking into account all its constraints as memory, computing power and energy, our proof of concept promotes the sharing of data and the utilization of services as a real IoT network, securely. Moreover, we set up a prototyping platform to perform evaluations in real conditions, demonstrating the relevance of our approach.

The remainder of this paper is organized as follows: Section II is dedicated to related works; then we provide general assumptions of this study in the third part. Section IV presents our *Wallance* protocol and its implementation is described in the next section. Section VI is dedicated to the experimental results.

II. Related Works

An IoT network aims to connect everything to share data and create services. To enable this infrastructure, the main challenge is to ensure the trust in this decentralized system. According to the state of the art, blockchain is the reference solution. Many surveys [1-4] describe it in detail and provide directives for its integration with IoT. In this paper, we only introduce the main characteristics of blockchain to emphasize challenges for an IoT adaptation.

A blockchain is a distributed ledger, replicated among a network of peers. This ledger is composed of blocks containing transactions made in the network. Blocks are linked together by a reference to the previous one, forming an immutable chain. The classic structure of the blockchain is a single main chain, like Bitcoin [5] and Ethereum [6] where blocks are added one after the other to the chain. To speed up the block generation, some blockchains as IOTA [7] are based on Directed Acyclic Graph (DAG), where blocks have several

parents. Therefore, it is possible to add several blocks at a time. Another blockchain structure improving the scalability is the block-lattice, used by Nano [8]. The idea is to create a sub-chain for each participant. Each block depends only on blocks on its sub-chain, resulting in a fast and asynchronous update of the chain. In order to replicate exactly the same blockchain on every peer, a consensus protocol is needed to select unanimously the next block. With a probabilistic protocol such as Proof of Work (PoW) or Proof of Stake (PoS), a node is selected to propose the new block according to its hash power (PoW) or the amount of its stake (PoS). However, because of their probabilistic approach, several nodes can propose different new blocks at the same time, resulting in different versions of the blockchain, so-called "fork". The resolution of the latter implies a latency to ensure the confirmation of blocks and transactions (60 min in Bitcoin). Nevertheless, probabilistic consensus handles a huge number of peers, ideal for a trustless environment, like IoT. At the opposite, with a voting-based consensus protocol, like Practical Byzantine Fault Tolerance (PBFT) [9] and Algorand [10], a new block is added only after a majority vote. These consensuses are more efficient in terms of energy efficiency, avoid the fork and the transaction throughput is maximum. However, their scalability in terms of participants is low because of all voting communications, which increases quadratically with the number of peers. Hybrid consensus such as Proof of Authority (PoA) [4] based on PoS with assigned time slot or a voting system, uses a set of predefined nodes as leaders which are the only ones to produce blocks. Nonetheless, the main drawback of such a scheme is the fixed list of leaders, reducing the distributed approach and strongly assuming that these nodes are trustworthy. Consequently, PoA and voting-based consensus are preferable in private blockchain, where there are few nodes, contrary to an IoT network. Another way to improve the scalability is the off-chain solution such as the lightning network [11]. The idea is to create a channel between two parts allowing them to transact without involving the blockchain, except for the opening and the closing of the channel. However, this approach does not entirely solve the problem of scalability due to the consensus algorithm.

Today, the community agrees on some real issues, challenging the blockchain's adoption for the IoT [1-4], such as computing power, scalability, and storage utilization. Indeed, the latter is a major constraint of blockchain because the full history of transactions is stored to ensure all the verifications from the very first block. At the time of writing this paper, the size has exceeded 306 GB and 255 GB [12] for Bitcoin and Ethereum respectively. IOTA could store 1 million transactions with 1.6 GB but it is a theoretical value, because of the lack of online explorer. According to a Nano node [13], 25.5 GB are used to store almost 49 million blocks. The size of these blockchains depends on the number of transactions and blocks, made by peers. Currently, there are less than 45 million user wallets [14] in Bitcoin, 72 million in Ethereum [15], about 400 000 in IOTA [16] and less than 400 000 in Nano [17]. However, in the IoT context with tens of billions of devices, the storage of their transactions implies a huge size of the blockchain.

In the state of the art, most of the papers provide overviews and challenges for an IoT blockchain. An important research area is the access control to resources by IoT devices and privacy preferences. Authors in [18] propose a framework to store on the Ethereum blockchain only the hash of encrypted data while data itself is stored in the trusted execution environment (TEE) such as Intel SGX. However, this leads to a not generic approach, which is not suitable for IoT. Furthermore, this implies the total confidence in the manufacturer, which is against the blockchain's policy. In [19] and [20] authors respectively propose a connected gateway to adapt and secure user privacy preferences of devices and to secure authorized access to IoT resources with Ethereum. They use Raspberry boards as IoT devices, but the blockchain is running on a desktop with Intel Core i7. Finally, [21] set up an entity called "management hub" used to connect IoT devices to the Ethereum blockchain as a gateway. Instead of including IoT devices in the blockchain, authors claim to create an interoperability method between IoT and blockchain.

Despite many contributions and huge efforts, there are very few real implementations on IoT architectures. Besides, most of them are only application-oriented and use already available blockchain platforms, like Ethereum. There are still challenges to set up a specific blockchain infrastructure dedicated to the IoT: ensuring the scalability in terms of transaction throughput and the number of connected nodes, allowing a real-time approach, while taking into account the IoT constraints.

III. Global Assumptions

In IoT networks, potentially many kinds of sensors (temperature, motion detector, etc.) and actuators (alarm, dimmer, etc.) use several communication protocols such as LoRa, BLE or ZigBee. This is a very heterogeneous environment, but the main idea is to enable interactions between devices, autonomously. In a classic network architecture, there is the "Thing world" *i.e.* sensors/actuators, which digitalize and act on their environment and the "Cloud world", which handles the computing power. All data are centralized to make decisions, implying a high-energy expenditure. Finally, gateways are the middle point between these worlds, mainly used to convert a radio signal from sensor to the standard TCP/IP protocol for the Cloud. Since gateways embed more computing and storage capabilities, they constitute the new "Fog world". All primary decisions are made by gateways and only specific data are sent to the Cloud, improving the latency, bandwidth, interoperability and energy consumption. Also, only the necessary information is sent on the network, avoiding the exposition of all data from the sensor to the Cloud. This distribution of intelligence reduces the communication chain and is the foundation of a decentralized network architecture.

To set up a decentralized network for IoT, we chose a publish-subscribe network model, which is more appropriate than the classic client-server in this context. In the latter, client actively asks for data, increasing the power consumption and bandwidth utilization. In the publish-subscribe model, the client simply waits for data. Amongst publish-subscribe protocols, several stand out, such as Message Queuing Telemetry Transport (MQTT) and Data Distribution Service (DDS). In [22], an evaluation emphasizes that DDS outperforms MQTT by comparing latencies, which highlights its real-time capabilities. The DDS-based network is composed of topics, an application-specific data domain made by the user *e.g.* room temperature, motion detection, etc. Participants on the network subscribe to topics and automatically receive new data.

Gateways are the key point between the radio frequency and the TCP/IP worlds. Even if their resources (computing power, storage capabilities, and energy source) are very limited compared to servers, gateways can pre-compute data before sending it to the network. Thanks to the M2M communication protocol, devices interact with each other in a decentralized way, reducing the solicitation of the Cloud. However, there is no standard to secure interactions in such a system, without a confident third part and taking into account the very limited resources of devices.

IV. WALLANCE

In the IoT context, since everything is connected to share data and create services, the main challenge is to enable the trust between devices, without a third party, in a decentralized way. To set up a fair and durable ecosystem, devices are connected to have access to resources/services, but in exchange, they have to contribute to the development and security of the network. The problematic is how to estimate the contribution of each device to control its access requests, unanimously and securely. Since blockchain cannot be used for the IoT, we present our *Wallance* protocol, a lightweight alternative to blockchain.

A. Principle

We created a monetary valuation, the *DCoin* (Data Coin), on the shared data quantity: each node earns division of *DCoin* called *DCoin Rate* for its data sharing. With their coins, node gets the right to access resources/services. For example, to rent a radio protocol or computing power of another node. The network, through the consensus protocol, based on the majority vote, controls all remunerations and accesses. To encourage the security of the network, a *DCoin Reward* recompenses nodes that have correctly voted.

B. Structure

A major constraint of blockchain is the storage utilization. Indeed, blockchains store the full history of transactions from the very first block. However, this approach is not suitable in the IoT context with tens of billions of devices and limited memory size. *Wallance* is based on a lattice wallet structure as Nano [8], shown in Fig. 1. Each node has a wallet, composed of its ID, its *DCoin* balance and a *State*, which is the fingerprint of the current wallet, like the hash of the block. Each new *State* depends on its previous *State* and the node's operation, ensuring the integrity of the wallet. In addition, each wallet is completely independent and does not depend on others: that means it can be updated asynchronously. In contrast, each blockchain full-node stores the whole wallets of the network. To avoid the continuous increase of the storage space, our structure does not keep the full history of wallets, but only their current version. Therefore, there is no history but the integrity is ensured by the *State* of each wallet. The structure of *Wallance* is a trade-off between the high-level security of blockchain and storage utilization.

C. Transactions

Nodes earn the *DCoin Rate*, for sharing their data through a *Sensor Transaction*. Our protocol does not impose a particular structure of this type of transaction but at least the ID of the node, the name of the topic and the sensor value must be specified as shown in Fig. 2 (a). The data sharing represents more than 80% of the total network activity. Consequently,

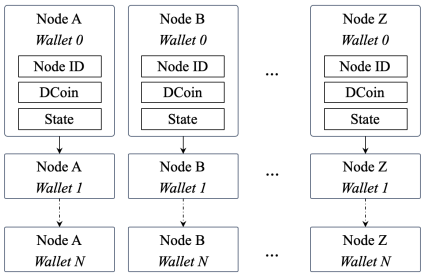

Fig. 1. Wallet Structure

the remuneration process has to be simple, without consensus. Since all nodes store the wallet of all the others, the remuneration is made on each node, locally increasing their version of the Sender's *DCoin* balance. Finally, nodes have to send a *Request Transaction* to access a resource/service as in Fig. 2 (b). To optimize the communication protocol, this transaction has to be as small as possible while ensuring the security of the protocol. *Request Transaction* indicates which node (*Requester*) wants to access which resource/service (*Service*), at what price (*Price*) and at what time (*Time*). To prevent the double-spending attack, node has to include the current *State* of its wallet. If several *Request Transactions* refer to the same *State*, that means node tempts to access to several resources at the same time as double spending scenario. Therefore, the *Wallance* protocol authorizes only one request per *State*. Even if that occurs, the majority will validate one of these requests, invalidating the second because of the wrong *State*. Finally, nodes have to include the solution of a cryptographic puzzle in the *Request Transaction*. As PoW in Bitcoin but with very low difficulty, we set up a Lightweight Proof of Work (LWPoW). The aim is to compute the fingerprint of the *Request Transaction*, including a random nonce. The hash has to begin with some nibbles to '0', according to the difficulty level. The only way to find a correct nonce is to try several values. This forces each participant to spend some computing power and time to have a valid transaction, to avoid network spamming. All transactions with a wrong nonce are not considered by nodes.

D. Consensus

To select unanimously the next block, full-nodes on blockchain execute the consensus algorithm. Bitcoin and Ethereum use the PoW, which is clearly unsuitable for the IoT because of its huge computing power and energy consumption. Besides, the consensus process has to respect

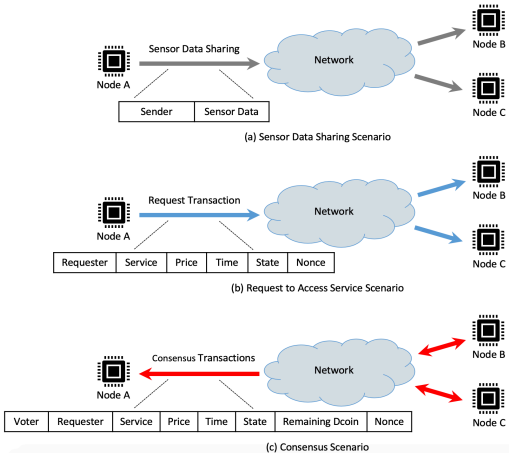

Fig. 2. *Wallance* Interaction Scenarios

the real-time characteristic of the IoT context. Therefore, the confirmation time of the transaction has to be quick, which is not allowed in probabilistic consensus algorithms, as PoW or PoS because of their fork probability. To meet these constraints and avoid forks, *Wallance* uses a voting-based consensus model, where a transaction is validated only if the majority agrees. Once the majority is reached, the transaction is definitively validated without additional time. We set the majority threshold to 2/3 of the network, the smallest theoretical number of correct participants that a Byzantine Fault Tolerance (BFT) system can handle [23]. As previously mentioned, the main constraint in such a consensus model is the scalability in terms of the number of participants. Indeed, the bigger the network, the more votes required. To handle a large number of them, we designed a small structure of the vote (in bytes) to reduce the latency impact and to improve the validation time. However, they have to include enough information to be resilient against major blockchain attacks such as double-spending.

As discussed in Section IV.A, the consensus aims to control the remuneration and the service access requests, unanimously. To limit the communication and computation overheads, the consensus process is only executed when an access request is received. At the same time, all previous remuneration operations can be certified. It is designed to avoid the multi-round steps and the timing dependence of most of the BFT algorithms. Once a *Request Transaction* has been validated (correct nonce, valid *State*, enough *DCoin*, etc.) by the node, a unique vote called *Consensus Transaction* is generated, without timeout constraint. Algorithm 1 describes the process of the vote's generation. As shown in Fig. 2 (c), all information on the *Request Transaction* is included, such as *Service*, *State*, etc. Also, *Remaining DCoin* is the amount of *DCoin* still available to the *Requester*, after the purchase of the service. To prevent multiple votes *e.g.* sending the same vote several times to force the majority, *Consensus Transaction* includes the *Voter*'s ID. Each node can generate only one vote per *Request Transaction*. Furthermore, *Consensus Transaction* requires a valid nonce, depending on the *Voter*'s ID. Each node has to find its own nonce, which avoids the replay vote by another node. Finally, the *Consensus Transaction* is sent to the network and the used *Request Transaction* is removed. Since there is no multi-round, once a vote is sent, it is too late for an attacker to stop the propagation of this vote and so prevent the consensus finality.

The *Request Transaction* is in pending mode until it has received enough votes from other nodes, without timing constraint. Periodically, nodes parse all received *Consensus Transactions* to find a majority. This is the consensus process, described in Algorithm 2. The aim is to find a group of identical *Consensus Transactions*, with enough voters (2/3 of the network). To form a group, transactions must have exactly the same information: *Requester*, *Service*, *Price*, *Time*, *State* and *Remaining DCoin*. In the same way as the verification of *Request Transaction*, only *Consensus Transaction* with a known *Voter*'s ID and the correct *State* of the *Requester*'s wallet is considered. Once a group reaches the majority, the wallet of the *Requester* is updated. First, the new *DCoin* balance is set by the *Remaining DCoin* value, estimated by the majority. As previously discussed, this represents the certification of all previous remuneration operations, such as the sharing sensor data. Thanks to that, the entire network resynchronizes on the same balance and the new *Requester* wallet's *State* is computed. Then, each node locally rewards correct voters by incrementing their wallets by the *DCoin Reward*. The synchronization of their balances is done during the consensus process initiated by these voters, thanks to the *Remaining DCoin* value. Finally, the used *Consensus Transactions* are removed and the *Requester* can access to the purchased service, through a Smart Contract, a piece of code executed only after the majority authorization.

Algorithm 1: Consensus Transaction Generation

```
Input: RequestTX
Output: ConsensusTX
if nonce not valid or (Service, Price) not valid then
    Remove(RequestTX)
else
    if State == Requester.Wallet.State then
        if Requester.Wallet.DCoin ≥ Price then
            ConsensusTX = GenerateConsensusTX(RequestTX)
            LWPoW(ConsensusTX)
        end
    end
end
```

Algorithm 1. Consensus Transaction Generation

Algorithm 2: Consensus Process

```
Input: ConsensusTXs
Output: Requester.Wallet
Group = ParseConsensusTXs(MajorityThreshold, ConsensusTXs)
if Group.State == Requester.Wallet.State then
    Requester.Wallet.DCoin = RemainingDCoin
    Requester.Wallet.State = Hash(State, Group)
    RewardParticipants(DCoinReward)
    if Node == Requester then
        SmartContract(Service)
    end
    Remove(Group)
end
```

Algorithm 2. Consensus Process

V. Implementation

According to Section III, gateways are the middle point between wireless and TCP/IP networks, with limited but sufficient resources to pre-compute data. To have a fair representation of gateways in IoT infrastructure *i.e.* computing power, storage capability, and energy consumption, we selected Raspberry Pi3 Model B+, composed of 64-bit quad-core ARM Cortex-A53 at 1.4 GHz. In this section, we propose a C/C++ implementation of our *Wallance* protocol. All source files are available on GitHub at *https://github.com/WallanceProject*.

A. Software Architecture

Raspberry embeds a Linux distribution to support our software architecture, described in Fig. 3. The processing unit receives data from a *Virtual Sensor*, which is only used for the prototyping platform version. These data are pre-computed by the *Processing Unit* and then transmitted to the network. We implement the publish-subscribe network model by

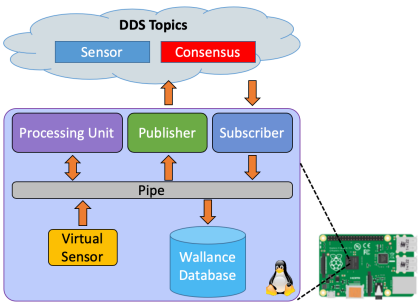

Fig. 3. Software Architecture

OpenDDS [24], an open-source implementation of DDS to keep our project generic. Currently, we have one data topic named *Sensor*, used to send and receive new data by *Publisher* and *Subscriber* processes respectively. Another topic, *Consensus*, is used to send all resource/service access requests and to perform the consensus process. In order to store data in a generic, compact and flexible manner, the *Wallance* database is implemented by the SQLite library [25] to build a relational one-file database, composed of three tables: *Wallets*, *Request Transactions*, and *Consensus Transactions*.

B. *Wallance Configuration*

To provide a flexible protocol, the designer can customize several parameters, listed in Table I. In our implementation, we set a majority threshold to 2/3 of the network as classic BFT protocols. We use the SHA-256 hash function for our LWPoW. The difficulty level requires only one nibble to '0' to limit the computing overhead. We set the *DCoin Rate* to 5, which means each *Sensor Transactions* worth 0.2 *DCoin*. To emphasize the incentive of consensus participation, the *DCoin Reward* is set to 2 *DCoin*. Finally, we fix the consensus period at 2 seconds for the real-time constraint of our IoT application. Since our consensus is based on the majority, the network size is a critical element to secure the system. In our first prototype, we set a static size of 8 Raspberry nodes, so with a majority threshold of 2/3, at least 5 *Consensus Transactions* are needed to reach the majority. Currently, each node has the same voting weight and no additional node can be added. Also, we set the ID of each node and we assume that is unforgeable.

C. *Wallance Interface*

As shown in Fig. 4, we provide a user-friendly interface, based on Grafana [26], to visualize all nodes' wallets and to interact with our platform in real-time. We can purchase services, by generating a *Request Transaction* on behalf of selected node. However, the interface does not participate in the consensus process *i.e.* it does not send vote. As a demonstrator, we set up an espresso service to prepare coffee through a connected machine.

VI. Results

Thanks to our implementation, we can evaluate in real conditions our *Wallance* protocol. The *Consensus Transaction* has to be as small as possible not to overload the bandwidth of the network while ensuring security. Indeed, the delay to reach the consensus depends on the propagation time of the vote on the network, which depends on the size of the vote. With a 4-byte integer (*Price*, *Time*, *Nonce*), a 4-byte float (*Remaining DCoin*) and a 32-byte string for identifiers (*Voter*, *Requester*, *Service*, *State*) our *Consensus Transaction* size is up to only 144 bytes. To characterize the latency, the protocol

TABLE I. List of Parameters

Parameters	Description	Impact on the system
Majority Threshold	Minimum number of votes to reach consensus	Security level to control the majority
Difficulty	Number of nibbles to '0' at the beginning of the hash	Computing overhead Energy consumption Vote latency
DCoin Rate	Number of data shares to earn 1 *DCoin*	Incentive of the sharing
DCoin Reward	Reward for the participation in the consensus process	Incentive of consensus participant (security)

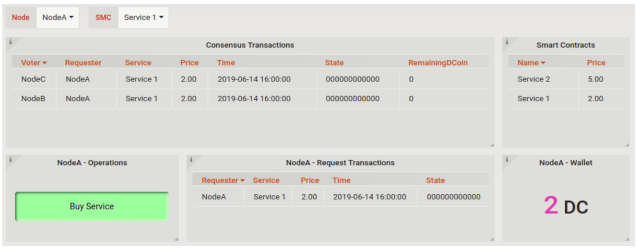

Fig. 4. *Wallance* Interface based on Grafana

is as follows: a Raspberry node is connected to a TP-LINK Archer C50 router over 5 GHz WiFi and sends a transaction on the *Consensus* topic. Once received, the node measures the time between the sending and receiving operation, then another transaction is sent. 10 000 samples have been used to perform this evaluation, shown in Table II. Several sizes of *Consensus Transaction* have been used, by modifying the identifier length. Because the transaction remains small, there is no significant impact on the latency, which is on average less than 2ms. Note that the minimum/maximum values are respectively the lowest and highest latency obtained during the test. The aim is to bound the transaction latency and these values depend on the router capability to broadcast transactions over the network at the time of the test. Besides, the delay to reach consensus depends on the number of nodes who have to send their vote. Therefore, the expected number of required *Consensus Transactions* proportionally increases with the network size according to the majority threshold. In our platform, with 8 nodes, the consensus is reached in 300 ms and less than 1s with 48 nodes. According to these results, we estimate that *Wallance* reaches the consensus in 2s and less than 20s with 1 000 and 10 000 voting nodes respectively. It is important to note that these estimations are made taking into account the adaptation of the network infrastructure (addition of routers and switches) according to the number of connected nodes.

To avoid spams, each node has to compute a LWPoW to have a valid transaction. The aim is to find a nonce, creating a hash of the transaction beginning with some '0' according to the difficulty. Table III shows the computation overhead of the

TABLE II. Evaluation of the Consensus Transaction Latency

ID Size (bytes)	Total Size (bytes)	Average Latency (ms)	Min Latency (ms)	Max Latency (ms)
8	72	1.66	1.40	5.98
16	96	1.53	1.38	5.82
32	144	1.66	1.41	4.94

TABLE III. Evaluation of the Difficulty of the LWPoW Computation

Difficulty	Average # of Nonce	Min	Max	Average Computation Time
0	1	1	1	46 us
1	15.96	1	189	740 us
2	256.02	1	2847	12 ms
3	4104.01	1	42373	190 ms
18 (Bitcoin)	-	-	-	~ 7 billion years

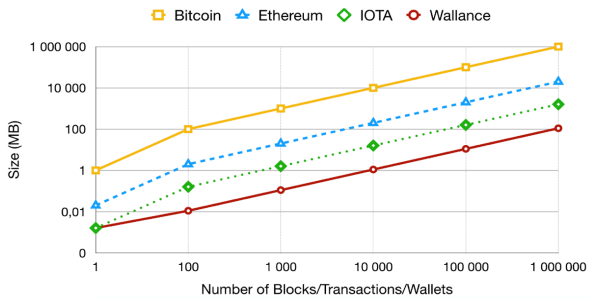

Fig. 5. Storage Utilization Comparison with Classic Blockchains

LWPoW using several difficulties. First, the hash operation without difficulty takes 46us. For each additional nibble to '0', the computing time is multiplied by 16, increasing the energy consumption. For comparison purpose, at the time of writing, Bitcoin requires 18 nibbles to '0' that represents on average 7 billion years for a Raspberry Pi3 to generate only one transaction. This emphasizes the inappropriate properties of blockchain such as Bitcoin for the IoT context.

Another important constraint is the memory utilization. Fig. 5 compares our work with well-known blockchains in terms of storage utilization. From Bitcoin with a 1 MB block, Ethereum decreases the size to 20 kB and IOTA, which uses transactions instead of blocks, reduces the size to only 1.6 kB. At this time, Bitcoin has exceeded 306 GB with more than 616 000 blocks and 255 GB for Ethereum with more than 9 000 000 blocks. Theoretically, the size of IOTA with 1 million of transactions is about 1.6 GB, reducing by 12.5 the size compared to Ethereum. It is important to note that these blockchains are used for cryptocurrency applications and require high-security level *e.g.* store the full history of transactions. However, in the context of *Wallance*, the security level may be lower, to the benefit of the memory utilization. According to the empirical evaluation of our protocol, the size of a wallet is about 110 bytes, almost 10^4 times smaller than the Bitcoin block. Furthermore, the most important property is that our protocol does not depend on the number of transactions or blocks like classic blockchains, but only on the number of wallets, *i.e.* the number of participants on the network. An inherent characteristic of *Wallance* is its tailored storage utilization: node can store only part of the whole wallets in the network, according to its memory capacity. A limited node such as a smart sensor can store only 100 wallets (50 kB), while gateways, as a Raspberry board, can easily embed hundreds of GB and store 10 million wallets (1 GB).

VII. Conclusion and Perspectives

Standard centralized architectures such as Cloud have to handle the expansion of the connected devices and the volume of generated data. The recent blockchain development brings renewal in a decentralized and trustless environment. However, the computing power and storage requirements of blockchain are not suitable for the IoT. In this context, we developed *Wallance*, a proof of concept of an alternative to blockchain dedicated to IoT. According to this study, our protocol is a trade-off between security and resource utilization. *Wallance* does not require intensive computing power, compared to the classic blockchains. With only a few memory resources, *i.e.* of the order of a hundred MB, our system can easily handle millions of peers. Our protocol is fully customizable and reaches finality in the order of second until 1 000 nodes, without fork. Future works will focus on the extension of our platform with more nodes to characterize performances and the robustness under several attack scenarios. Furthermore, network sharding is an interesting approach to improve scalability in terms of memory utilization and consensus latency.

References

[1] T. M. Fernandez-Carames and P. Fraga-Lamas, "A Review on the Use of Blockchain for the Internet of Things," *IEEE Access*, vol. 6, pp. 32979–33001, 2018.

[2] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain Technologies for the Internet of Things: Research Issues and Challenges," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2188–2204, Apr. 2019.

[3] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, Nov. 2018.

[4] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in IoT: The challenges, and a way forward," *Journal of Network and Computer Applications*, vol. 125, pp. 251–279, Jan. 2019.

[5] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9.

[6] V. Buterin, "A Next Generation Smart Contract & Decentralized Application Platform," p. 36.

[7] S. Popov, "Tangle.pdf." [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf. [Accessed: 25-Jul-2019].

[8] C. LeMahieu, "Nano: A Feeless Distributed Cryptocurrency Network," p. 8.

[9] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," p. 14.

[10] J. Chen and S. Micali, "Algorand," *arXiv:1607.01341 [cs]*, Jul. 2016.

[11] J. Poon and T. Dryja, "The Bitcoin Lightning Network:," p. 59.

[12] "Bitcoin, Litecoin, Namecoin, Dogecoin, Peercoin, Ethereum stats," *BitInfoCharts*. [Online]. Available: https://bitinfocharts.com/. [Accessed: 25-Jul-2019].

[13] "NanoCrawler Node Status." [Online]. Available: https://nanocrawler.cc/status. [Accessed: 13-Feb-2020].

[14] "Blockchain Wallet Users," *Blockchain.com*. [Online]. Available: https://www.blockchain.com/charts/my-wallet-n-users. [Accessed: 25-Jul-2019].

[15] "Ethereum Unique Address Growth Chart." [Online]. Available: https://etherscan.io/chart/address. [Accessed: 25-Jul-2019].

[16] "Statistics about tokens distribution." [Online]. Available: https://thetangle.org/statistics/tokens-distribution. [Accessed: 25-Jul-2019].

[17] "NanoCrawler All Accounts." [Online]. Available: https://nanocrawler.cc/explorer/accounts/1. [Accessed: 13-Feb-2020].

[18] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, "Decentralized IoT Data Management Using BlockChain and Trusted Execution Environment," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, Salt Lake City, UT, 2018, pp. 15–22.

[19] S.-C. Cha, J.-F. Chen, C. Su, and K.-H. Yeh, "A Blockchain Connected Gateway for BLE-Based Devices in the Internet of Things," *IEEE Access*, vol. 6, pp. 24639–24649, 2018.

[20] O. Alphand *et al.*, "IoTChain: A blockchain security architecture for the Internet of Things," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, 2018, pp. 1–6.

[21] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.

[22] J. Laurent, P. Benoit, L. Dalmasso, and T. Gil, "Computing in the Fog with Reconfigurable Gateways," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, Florence, 2018, pp. 1–4.

[23] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[24] "OpenDDS." [Online]. Available: http://opendds.org/. [Accessed: 25-Jul-2019].

[25] "SQLite Home Page." [Online]. Available: https://www.sqlite.org/index.html. [Accessed: 25-Jul-2019].

[26] "Grafana Labs," Grafana Labs Blog. [Online]. Available: https://grafana.com/. [Accessed: 25-Jul-2019].