Delay and Communication Tradeoffs for Blockchain Systems With Lightweight IoT Clients

Danzi, P.; Kalør, A. E.; Stefanovi, .; Popovski, P.

Published in:
IEEE Internet of Things Journal

DOI (link to publication from Publisher):
10.1109/JIOT.2019.2906615

Publication date:
2019

Document Version
Accepted author manuscript, peer reviewed version

Delay and Communication Tradeoffs for Blockchain Systems with Lightweight IoT Clients

Pietro Danzi, *Student Member, IEEE,* Anders E. Kalør, *Student Member, IEEE,* Čedomir Stefanović, *Senior Member, IEEE,* Petar Popovski, *Fellow, IEEE*

***Abstract*—The emerging blockchain protocols provide a decentralized architecture that is suitable of supporting Internet of Things (IoT) interactions. However, keeping a local copy of the blockchain ledger is infeasible for low-power and memory-constrained devices. For this reason, they are equipped with *lightweight* software implementations that only download the useful data structures, e.g. state of accounts, from the blockchain network, when they are updated. In this paper, we consider and analyze a novel scheme, implemented by the nodes of the blockchain network, which aggregates the blockchain data in periodic updates and further reduces the communication cost of the connected IoT devices. We show that the aggregation period should be selected based on the channel quality, the offered rate, and the statistics of updates of the useful data structures. The results, obtained for the Ethereum protocol, illustrate the benefits of the aggregation scheme in terms of a reduced duty cycle of the device, particularly for low signal-to-noise ratios, and the overall reduction of the amount of information transmitted in downlink from the wireless base station to the IoT device. A potential application of the proposed scheme is to let the IoT device request more information than actually needed, hence increasing its privacy, while keeping the communication cost constant. In conclusion, our work is the first to provide rigorous guidelines for the design of lightweight blockchain protocols with wireless connectivity.**

***Index Terms*—Internet of Things, data structures, blockchain.**

I. Introduction

TABLE I
Abbreviations used in this paper

Abbreviation	Description
BN	Blockchain node
BS	Base station
dApp	Decentralized application
PoI	Proof of inclusion
PoMI	Proof of multiple inclusions
RPL	Recursive Length Prefix
SPV	Simplified Payment Verification

SINCE the advent of the Bitcoin protocol in 2008 [1], a large wave of blockchain protocols has emerged, aiming to support the implementation of decentralized applications, or *dApps*, that reduce the need of a central authority to supervise the interactions in autonomous systems, including Smart Grids [2]–[5] and Internet of Things (IoT) [6]–[9].

Authors are with the Department of Electronic Systems, Aalborg University, Denmark, Email: {pid, aek, cs, petarp}@es.aau.dk.

The work was supported in part by the European Research Council (ERC Consolidator Grant no. 648382 WILLOW) within the Horizon 2020 Program.

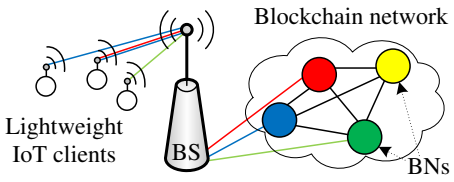

Fig. 1. Communication architecture for the interaction of IoT devices with a set of blockchain nodes (BNs) via wireless links provided by a base station (BS).

The IoT devices normally reside at the edge of the blockchain network, to which are connected through a set of regular blockchain nodes (BNs), e.g. via a wireless base station, see Fig. 1. The information exchanged with the BNs largely depends on the IoT application [10], that we divide into two types. In applications such as wireless sensor networks, devices are simple data sources and are often unattended. As such, they are prone to malicious attacks and malfunctioning, that should be accounted and detected remotely [11]. This can be implemented by a blockchain network with very limited feedback from the BNs to the IoT devices. On the other hand, there are applications that need to frequently send data back to the devices. For instance, a dApp may support key and spectrum management in the upcoming femtocells networks [12], or coordinate devices in smart grids [2]–[5].

This paper only considers the applications of the second class, in which the IoT devices need to retrieve authenticated information that is stored in the blockchain. However, storing the entire blockchain and processing every transaction require a remarkable amount of storage memory and computations. This is not feasible for IoT devices, as they are often constrained with respect to memory, computation, communication and power. Instead, the IoT devices may act as *lightweight* clients, which only store a subset of the blockchain data and eventually generate transactions to be included into the blockchain. Such devices are frequently synchronizing with the BNs [13], receiving a minimal amount of information, namely the *block headers*. In addition, when certain events that are of interest to a specific device occur, e.g. modification of specific accounts' state or transactions involving these accounts, the BNs transmit the updates to the device, including the proof of their inclusion (PoIs) in the blockchain.

While the architecture with lightweight clients reduces the processing and memory requirements, there is still a need

for a remarkable amount of downlink traffic in order to maintain synchronization to the global blockchain [13]. This type of operation challenges the common assumption that IoT devices mostly generate uplink traffic [14], [15], urging the investigation in accurate models for blockchain traffic. Without them, the industry lacks tools to determine which technologies, in the vast landscape of wireless IoT [12], are capable of supporting the blockchain traffic at the minimum viable cost.

Schemes that reduce the amount of traffic exchanged between the lightweight clients and the BNs have previously been proposed, either by modifying the block structure [16], [17] or by leveraging on the characteristics of account-based blockchains, like Ethereum [18]. A completely different approach is to remove the need of continuous synchronization by backing the authenticity of the information, transmitted by BNs, with a deposit of credit [19], [20].

This work is motivated by the observation that the blockchain synchronization process can be tailored to the actual requirements of timely information updates to the IoT devices. That is, the ultimate target is not to keep the devices always synchronized, but to synchronize them according to the needs of the underlying dApp. Hence, we replace the legacy scheme, in which the BNs transmits the information to the devices whenever available, with a novel approach in which the information is accumulated and pushed only when needed by the end IoT devices. Among the multitude of blockchain protocols, we focus on the Ethereum specification [21], but the overall principle of aggregation can be applied to other "account-based" blockchains.

The contributions of this work can be summarized as follows:

1) We propose and analyze an aggregation scheme, implemented at the BNs, that reduces the duty cycle of the device and the amount of transferred data, at the cost of an increased information delay at the IoT device. The reduction is achieved when events of interests to the device occur multiple times within an aggregation period, and is mainly caused by avoiding transmission of temporary states, but also because the size of the proof of inclusion increases sublinearly with the number of events.
2) We extend our previous model [13] by including the possibility for the IoT device to observe multiple accounts, and for the base station to select the transmission rate. The result is a model for lightweight clients that is rich, but simple to analyze. We show its potential application by constructing a set of observed accounts that increases the privacy of the IoT device, while keeping the communication cost low.
3) We study the cost of transmitting the proof of inclusion for the updated data, namely the Merkle-Patricia tree data structures, and provide experimental results obtained for Ethereum protocol.

The remainder of the paper is organized as follows. Section II provides an introduction to blockchain protocols, focusing on Ethereum, and describes the lightweight protocol variants. The system model is introduced in Section III and analyzed in Section IV. Section V presents the evaluation and Section VI a discussion of possible extensions and applications. Section VII concludes the paper.

II. Blockchain protocol

This section introduces the main components of the Ethereum protocol [21], that is a popular choice for blockchain systems tailored for IoT applications [2], [8], [22]. The common trait of Ethereum with other blockchain protocols can be found in [23].

The Ethereum blockchain is a database that records of history of the states of *accounts* in a chain of blocks. An account is a data structure that contains an amount of credit and a general-purpose memory block. The account may also contain a set of predefined procedures that can read and write to the memory; in this case, the account is called *smart contract*. The state of an account can be changed by *transactions*, either directly or through the invocation of a procedure in a smart contract. We shall refer to these modifications of accounts as *events*.

Transactions are signed by devices using an asymmetric cipher, and identified by their hash values[1], as in the Bitcoin specification [1]. The transactions are organized in a chain of blocks. Besides a set of transactions, each block contains cryptographic signatures of the current states of the accounts and a pointer to the preceding block in the chain, which defines a causal relationship between blocks. When a block is appended to the blockchain, the transactions that it includes are considered valid.

The Ethereum database is replicated at multiple nodes that are interconnected by a communication network. Every time a node appends a new block to its copy of the blockchain, it propagates the block to the rest of the network, to keep the database replications consistent.

A. The block data structure

A block is composed of a header and a body, see Fig. 2. The block header has a fixed size, while the rest of the block contains the actual transactions and has a variable size. When the number of transactions in a block is high, the variable-size part takes a dominant portion of the total block size. The information specified in the header includes: the block hash value, an incremental counter, the cryptographic signature of the node that generated it, the proof that the block is valid, e.g., Proof of Work solution, and one or more hash values that represent roots of PoI trees. In this work, we mainly consider the transactions tree and the state tree. The transactions tree, which uniquely binds the modifications of accounts certified by a block with the block header, can be used to prove that specific transactions are included in the block. The state tree, depicted on the right-hand side of Fig. 2, provides a snapshot of the entire collection of account states.[2]

B. Proof of inclusion (PoI) via Merkle-Patricia trees

The Ethereum protocol provides PoIs using Merkle-Patricia trees [21], [24], [25]. A Merkle-Patricia tree has three types

[1]The hash value of some input data x is the output of a hash function defined by the blockchain protocol, and is indicated as $h(x)$.

[2]The state tree is a characteristic of the Ethereum specification.

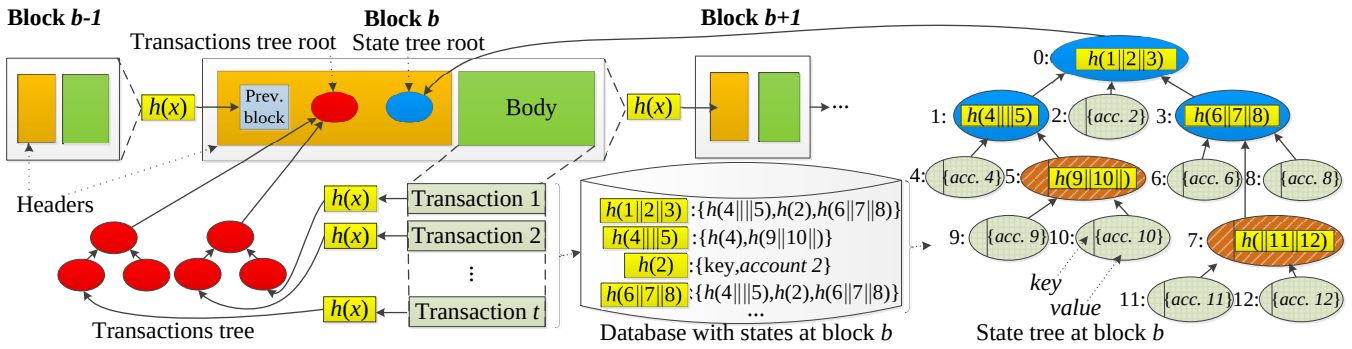

Fig. 2. Example structure of a blockchain. $h(x)$ is the hash value of node x and || is the concatenation operation. The t transactions included in block b apply modifications to the accounts' states, which are stored in a database. The state tree depicted on the righthand side, ternary in this example, is build from this database, and its root included in the block header. Branch nodes are colored in blue and extension nodes in brown. Leaf nodes (there are eight of them in the example) are composed by key and value, and colored in green.

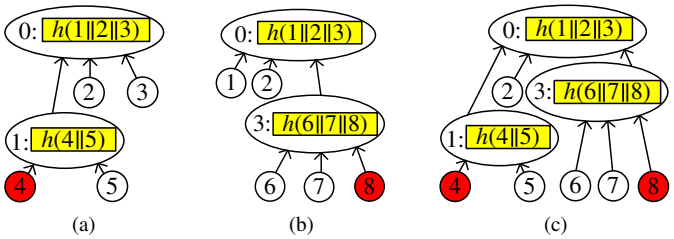

Fig. 3. Representation of the PoIs (a) of node 4, (b) of node 8 of Fig. 2, and (c) PoMI of 4 and 8. $h(x)$ is the hash value of x and || is the concatenation operation.

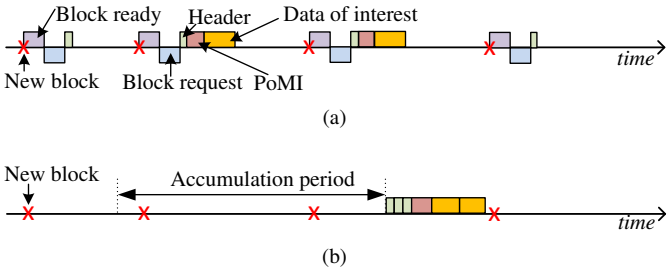

Fig. 4. Information exchanged between a client and a BN using a lightweight protocol during four block periods, (a) without aggregation scheme and (b) with aggregation. Downlink/uplink messages are depicted above/below the time arrow. "Data of interest" includes the accounts' data and relative PoMIs.

of nodes; leaf, extension and branch nodes, see Fig. 2, as in standard Patricia trees [26], and is used to efficiently store and retrieve data structures associated with strings. In the blockchain context, the string is the hash value of the address of an account or transaction, and the data structure to be retrieved is the account/transaction itself. The branch nodes only store the hash value of the list of its child nodes, see Fig. 2. Leaf and extension nodes, also illustrated in Fig. 2, store a key, that is the hash value of the common path shared by all child nodes, and a value. The value stored by extension nodes is the hash value of the list of child nodes, and the one of the leaf nodes is the hash value of the data that is to be authenticated (e.g. an account or transaction). The use of a hash function to index the addresses provides equal length of the strings, which are equiprobable.

The presence of a specific node in the Merkle-Patricia tree is proven by constructing its PoI. A PoI is a collection of node values that enables generation of the hash value, contained by the root node of the tree, e.g. the node labeled 0 in Fig. 2, starting from the specific node to prove. By comparing the generated root hash value with the value stored in the block header, the inclusion in the blockchain of the data structure, associated with the specific node, can be verified [1]. In practice, the PoI is used to verify that a particular leaf node, i.e. an account or transaction, is present in a state tree. Specifically, a PoI is created by starting from the root of the Merkle-Patricia tree, and descending to the specific node. At each level, all nodes, that are siblings to the node on the path from the root to the specific node, are collected, as illustrated in Fig. 3 ((a) and (b)). Notice that a PoI, in general, contains much fewer nodes than the complete Merkle-Patricia tree since most of the branches are not collected during the descent from the root [25].

A single proof can be constructed to prove multiple data structures by collecting the union of the nodes required to prove each of the data structures. We shall refer to such a proof as a Proof of Multiple Inclusions (PoMI)[3]. Since the nodes required to prove each data structure are likely to intersect, a PoMI is typically much smaller than if each data structure is to be proven by an individual PoI. Fig. 3(c) provides an example of this reduction, for the proof of both nodes 4 and 8. If two individual proofs are build, the PoI of node 4 contains nodes $\{5, 2, 3\}$ and the PoI of node 8 contains $\{1, 2, 6, 7\}$, such that seven nodes are needed in total. However, if the proofs are sent together in a PoMI, only nodes $\{2, 5, 6, 7\}$ are needed, motivating the advantage of using this data structure.

C. Synchronization protocols

A blockchain client is updated on modifications of the blockchain database, observed by BNs, by means of a synchronization protocol. In [13] we have presented two possible protocols that can be adopted for this purpose, denoted by P1 and P2. With P1, the client itself stores the entire blockchain,

[3]In contrast with prior literature [24], we use the terms PoI and PoMI to differentiate the proof from the blockchain-specific data structure that provides it, e.g. Merkle tree (in Bitcoin) or Merkle-Patricia tree (in Ethereum).

and locally checks the correctness of the transactions. This configuration is not envisaged for IoT devices due to the requirements of storage memory and processing, and will not be considered in this work. In P2, that includes protocols such as Bitcoin's Simplified Payment Verification (SPV) [1] and the Ethereum Light Client [27], the BNs are notified about the account updates that the client is interested in receiving. Hence, the client receives the block headers from the BNs, by default, and the accounts of interest, only when they are modified. This scheme, referred to as a *lightweight protocol*, reduces the amount of data communicated in the downlink, as well as the amount of local processing. In fact, the client only verifies that the information sent by BNs is consistent, delegating to them the auditing of the actual validity of the transactions [25]. It follows that, with P2, the client must be connected to at least one honest BN to be able to detect the presence of false information. Finally, there is a large class of IoT devices that is incapable of synchronizing with a global blockchain. These are devices equipped with low-rate wireless interfaces, e.g. LoRaWAN, or that have limited energy provision. A device of this class connects to a proxy node that only sends to the device the useful information, without providing any proof that is included in the blockchain. This class of protocols, hereby identified as P3, is also not considered in this work, as it requires the device to fully trust the proxy node, which is not in line with the envisioned trustless decentralized architecture.

In this paper, we consider IoT devices with low memory and communication capabilities but still capable of supporting lightweight protocols, i.e. of type P2. Fig. 4 shows the messages exchanged between a client and a BN using such protocols during three block periods. The red crosses represent the instants at which new blocks are generated. In the basic lightweight protocol, Fig. 4(a), the information is pushed in the downlink from BS as it becomes available. In the first block period there is no information of interest, and only the block header is sent, while in the second and third periods there are events of interest and the respective data are sent with their PoMI.

The presence of both transactions and state tree roots in the Ethereum block headers, see Fig. 2, permits to adopt two different approaches to update the local copy of the account states, as illustrated with the following example. Suppose that an account is updated multiple times during several block periods. The BN can send the last version of the account data, with the corresponding partition of *state* tree at the last block. In this case, the IoT device just replaces the local data if the PoI root matches the one included in the last block header, otherwise refuses it, cf. [18], [24]. Alternatively, the BNs send the whole sequence of transactions that modified the account, along the block periods, together with the collection of their PoIs build from the *transactions* tree. The sequence of transactions is applied, by the device, to its local version of the account state, to finally reconstruct the updated state. In this paper, we only consider the first approach, and we remark the extension to the second one in Sec. VI-B.

III. System Model

The scheme proposed in this paper aggregates the information in order to reduce the communication cost, see Fig. 4(b). Given that the application run by the client can tolerate a delay, information is accumulated at the BN and then periodically released at the subsequent aggregation point. The approach followed by the BN is to always send a proof by means of the state tree, triggering the replacement of the local copy of the client. This permits to send only the *latest* version of accounts that are modified multiple times during the accumulation, and merge the PoIs of accounts modified in *different* blocks, in a unique PoMI. The scheme is investigated in detail in the rest of the paper.

A. Blockchain network and IoT device

We consider a blockchain network in which new blocks are generated at exponentially distributed intervals with (network-wide) rate λ. A single IoT device is connected to a set of N BNs via a wireless link through a base station, see Fig. 1. The blockchain traffic on the wireless link is generated by two different processes: (i) the transmission of transactions, to be included in the blockchain, from the device to the BNs, and (ii) the exchange of messages as part of the synchronization protocol. However, we note that (i) only involves the transmission of the transaction meta-data (mainly the signature of the device), which has a deterministic size. For this reason, this process is not treated in the rest of the paper. Regarding process (ii), the device subscribes to block headers for all generated blocks as well as state updates for a set $\mathcal{A}$ of accounts, which are a subset of the existing accounts. The generic account, indexed as $j \in \mathbb{N}$, is updated independently in a block with probability (or relative frequency) p_j. We consider the case where the device is not interested in the full state history of the observed accounts, but merely in their most recent state. That is, the device needs to be informed about only the most recent state of the observed accounts, as well as receive the PoMI that proves the inclusion of the specific account states in the blockchain. The case is representative of a class of problems in which the *age* of the information, i.e. data freshness, is more valuable than tracking all state changes, and includes environmental monitoring applications and power grid stabilization systems [28].

To simplify the presentation, we assume that a block header and updated accounts' states take up a fixed number of l_{H} and l_{a} bits, respectively. In contrast with this, the size of the PoMI, with length l_{PoMI} bits, is random as a result of the PoMI tree data structure. Specifically, as described in Section II, the size of the PoMI is sublinear in the number of accounts.

B. Aggregation protocol

The block headers and the updated observed accounts are aggregated at a BN, termed *aggregation BN*, selected by the device, and transmitted to the device periodically every T seconds. The value of T depends on the information delay, tolerated by the application, from the instant at which the account is modified, to the instant at which the update is delivered

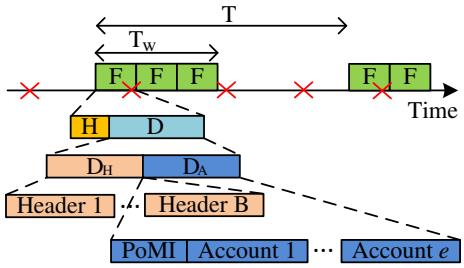

Fig. 5. The periodic release of information. Red crosses represent block generations. In the first release, there are two retransmissions of the frame (F), due to failure, in the second release, only one retransmission.

to the device. Upon successful reception of the transmission, the IoT device acknowledges the packet. We assume that the device selects the sequence of aggregation BNs, over different aggregation period, as part of the initial network association procedure, e.g. by means of a seed sequence. Consequently, the execution of the protocol only requires downlink messages, because all the information, needed by BNs, is sent by the IoT device in the initialization phase. When no transmission is ongoing, the device is assumed to be in power-saving mode.

C. Wireless link

The wireless downlink from the base station to the IoT device is assumed to be a block Rayleigh-fading channel with constant channel gain over the duration of a transmission and independent channel gains across transmissions. This occurs, for example, in system based on per-packet frequency hopping (FH). Due to the power constraints of the IoT device, we assume that the base station has no information about the channel and hence performs no power or rate adaptation. As a result, a transmission may fail with probability [29]

$$p_{\text{out}} = 1 - \exp\left(-\frac{2^{\frac{R}{W}} - 1}{\gamma}\right), \tag{1}$$

where γ is the average received signal-to-noise ratio (SNR), R is the transmission rate in bits/s, W is the bandwidth of the channel in Hz. The downlink packet is retransmitted until it has been received successfully by the device.

In contrast to the downlink transmissions, we assume that the transmission of the acknowledgment packet in the uplink happens instantaneously and is always received reliably, thanks to power control, performed at the IoT device side, based on the received transmission.

D. Frame structure

The downlink frame, represented in Fig. 5, consists of F bits, and is divided into a fixed number H of header bits, representing the standard communication protocol overhead, and a variable number D of payload bits, corresponding the blockchain information, i.e. $F = H + D$. Its duration is related to the transmission rate R as

$$T_w = \frac{kF}{R} \qquad [\text{s}],$$

where $k \geq 1$ is the number of transmissions, including retransmissions due to outage.

If the transmission of the frame takes longer than the transmission period, i.e. $T_w > T$, due to retransmissions, it is halted and considered failed. In this case, that has been analyzed in [13], since the block headers are required in order for the IoT device to stay synchronized to the blockchain, the next frame should include the block headers accumulated in the current frame. In this work, we consider the channel and block generation parameters that provide a negligible probability that the frame cannot be received in the current transmission period, so that the phenomenon can be ignored.[4] The D payload bits are divided into D_{H} bits for block headers and D_{A} bits for account updates, i.e. $D = D_{\text{H}} + D_{\text{A}}$. D_{H} and D_{A} are random as they depend on the number of generated blocks and account updates.

IV. Analysis

In this section, we present an analysis of the aggregation scheme. To this end, we first obtain the distribution of the frame size and frame transmission duration, and then use this result to evaluate the communication cost and latency.

A. Frame size distribution

Recall that the frame is divided into H header bits, D_{H} bits for block headers and D_{A} bits for account states. The H header bits are fixed, while D_{H} depends on the number of generated blocks during the aggregation period T, indicated as B. Similarly, D_{A} depends on the number of generated blocks, as it impacts the number of observed accounts that are updated. We indicate the probability distributions of D_{H} and D_{A}, conditioned on the number of generated blocks, respectively as $p_{D_{\text{H}}|B}$ and $p_{D_{\text{A}}|B}$. As a result, we may factorize the distribution of the total frame size F as

$$p_F(f) = \sum_{b=0}^{\infty} p_B(b) \sum_{i=0}^{f} p_{D_{\text{H}}|B}(i|b) p_{D_{\text{A}}|B}(f-i|b), \tag{2}$$

where we have used the fact that $f = D_{\text{H}} + D_{\text{A}}$. The possible sizes are conditioned on the event that b blocks are generated during T given by

$$p_B(b) = \frac{(\lambda T)^b \exp(-\lambda T)}{b!}. \tag{3}$$

The formula directly follows from the assumption of exponential waiting time between blocks, which has been shown to hold for blockchains based on Proof of Work [30].

1) Distribution of D_H: Since we assumed that the block headers are always received within the current transmission period, i.e. $T_w < T$, the size of D_{H}, when B blocks are generated, can be approximated with the fixed quantity $B \cdot l_{\text{H}}$ bits, yielding $p_{D_{\text{H}}|B}$ in (2).

If this is not the case, the number of block headers that needs to be transmitted should be modeled as a bulk queue, where blocks arrive according to a Poisson distribution with

[4]Scenarios for which this assumption does not hold can be observed when the probability of outage is rather high, and the transmission rate is low.

rate λ, and are served in bulks of up to $\left\lceil \frac{D}{l_{\mathrm{H}}} \right\rceil$, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. However, this makes the accurate analysis intractable and is outside of the scope for this work.

2) Distribution of D_A: In order to obtain the size of the account updates D_{A}, we first need to characterize the number of accounts U that are updated during an aggregation period T. The probability that account j, characterized by relative frequency p_j[5], is updated at least once in b blocks accumulated during the aggregation period is given by $q_j = 1-(1-p_j)^b$. We denote by U the total number of accounts that are modified at least once in B blocks. Since each of the accounts is updated independently conditioned on B, U follows a Poisson binomial distribution parameterized by the account update probabilities $q_1, q_2, \ldots, q_{|\mathcal{A}|}$:

$$p_{U|B}(u|b) = \sum_{\mathcal{B}\in\mathcal{F}_u} \prod_{j\in\mathcal{B}} q_j \prod_{l\in\mathcal{F}_u\setminus\mathcal{B}} (1-q_l)\,. \tag{4}$$

$\mathcal{F}_u$ is the set of all subsets of $\{1, 2, \ldots, |\mathcal{A}|\}$ with cardinality u, $\mathcal{B}$ is an element of $\mathcal{F}_u$ and $\mathcal{F}_u \setminus \mathcal{B}$ is the complement of $\mathcal{B}$. This distribution is used to find the distribution of D_{A}, conditioned on B:

$$p_{D_{\mathrm{A}}|B}(a|b) = \sum_{u=0}^{|\mathcal{A}|} p_{D_{\mathrm{A}}|U,B}(a|u,b)\cdot p_{U|B}(u|b). \tag{5}$$

Notice that $p_{D_{\mathrm{A}}|U,B}(a|u,b)$ only depends on the realization of the number of modified accounts u. This permits us to write

$$\begin{aligned} p_{D_{\mathrm{A}}|U,B}(a|u,b) &= p_{D_{\mathrm{A}}|U}(a|u) \\ &= \sum_{i=0}^{a} p_{l_{\mathrm{PoMI}}|U}(i|u)\, p_{l_{\mathrm{acc}}|U}(a-i|u). \end{aligned} \tag{6}$$

To complete the analysis, we need to characterize the size of PoMI and accounts' information. The size of accounts' information varies only with the number U of modified accounts,; as in our model they have fixed size of l_{a}, this size is simply $l_{\mathrm{acc}} = U \cdot l_{\mathrm{a}}$ bit. On the other hand, the PoMI length l_{PoMI} does not only depend on the number of observed accounts, but also on their position in the state tree. For simplicity, we assume that the tree is perfectly balanced at all levels, and that the location of the observed accounts at the last level is uniformly distributed. The approximation is supported by the fact that Patricia trees are generally well-balanced [31].

However, the fact that the number of proofs that each node can be part of is bounded by the number of descendant leaf nodes, makes the problem of obtaining the distribution of the number of nodes in the PoMI a hard combinatorial problem; even the expected value of the number of nodes is computationally intractable to obtain. Instead, we approximate the number of nodes by relaxing this condition. The resulting approximation captures the characteristics of the PoMI size as the number of modified accounts grows, and is accurate as long as the number of modified accounts is much lower than the total number of leaf nodes in the tree. This is typically the case, as the set of observed accounts is small. Specifically, relaxation results in the following recursive approximation of the expected number of *nodes* in a PoMI for u accounts, when the tree has height η:

$$\bar{N}_\eta(u) = \sum_{h=1}^{\eta} L\bar{N}_{h-1}(u)\left(1 - \frac{1}{L\bar{N}_{h-1}(u)}\right)^u, \tag{7}$$

with $\bar{N}_0(u) = 1$. The derivation is given in Appendix A.

To obtain the expected number of *bits* for a PoMI, we assume that the tree does not contain extension nodes, as they have variable size, see [26]. Hence, with this approximation, the internal nodes are only branch nodes. Indicated the size of the output of the hash function with l_{s}, each internal node has fixed size of l_{s} bits. Instead, the leaf nodes are composed by a key and a value, see Sec. II, both containing hash values, resulting in a fixed size of $2 \cdot l_{\mathrm{s}}$ bits. In conclusion, we obtain the expected number of bits required for a PoMI of u accounts:

$$\bar{l}_{\mathrm{PoMI}}(u) = l_{\mathrm{s}}\bar{N}_\eta(u) + u(2\cdot l_{\mathrm{s}}). \tag{8}$$

B. Transmission duration

The total transmission duration T_w depends on F and the number of (re)transmissions that is needed before the packet is successfully received by the IoT device.

A frame is transmitted successfully with probability $1-p_{\mathrm{out}}$, independent of the size of the frame, and hence the number of transmissions is geometrically distributed with the probability mass function

$$\Pr(k \text{ transmissions}) = p_{\mathrm{out}}^{k-1}(1-p_{\mathrm{out}}). \tag{9}$$

Since the rate remains fixed across (re)transmissions, it follows that the probability density function of T_w is

$$p_{T_w}(t) = \sum_{k=1}^{\infty} p_F\left(\frac{t}{k}R\right)\Pr(k \text{ transmissions}) \tag{10}$$

$$= (1-p_{\mathrm{out}})\sum_{k=1}^{\infty} p_F\left(\frac{t}{k}R\right)p_{\mathrm{out}}^{k-1}. \tag{11}$$

C. Data savings of the aggregation protocol

Since the states of accounts can be verified from the state tree root contained in the most recent block header, the aggregation scheme provides data savings by (i) sending a unique PoMI that certifies only the *latest* state of the modified accounts, (ii) sending only the most updated copy of the account data structure, and (iii) reducing the amount of frame overhead, H. We consider protocol P2 from [13] introduced in Sec. II-C as the benchmark. Recall that P2 requires the device to download the following information at each block period: the frame overhead H, a notification of the new block from each peer, the block header, the PoMI and the account data structures. In addition, with P2, the device receives a notification of new block, indicated as "Block ready" in Fig. 4(a), from each BN, and consequently selects a BN with uplink message, indicated as "Block request" in the same figure. However, to establish a fair comparison with the aggregation protocol, we assume that the BN, in charge of sending the update, is pre-selected via random seed also in P2, removing the need of these messages.

[5]This quantity is not assumed but experimentally estimated in Sec. V.

TABLE II
SYSTEM PARAMETERS

Blockchain					
λ	0.1 s^{-1}	H	1200	l_H	4046 bit
l_a	320 kb	l_s	256 bit	L	16
η	5				
Communication channel					
R	250 kbit/s	W	180 kHz	γ	30 dB

The expected amount of bits downloaded with protocol P2 during a block period is

$$\mathbb{E}\left[F^{(\mathrm{P2})}\right] = H + P = H + l_{\mathrm{H}} + \sum_{a=0}^{\infty} a \cdot p_{D_{\mathrm{A}}|B}(a|1). \tag{12}$$

The expression is based on (5), and on the fact that exactly one block is generated during a block period.

The expected number of bits per block period downloaded using the aggregation protocol proposed in this paper is given by averaging (2):

$$\mathbb{E}[F] = \frac{1}{\lambda \cdot T} \sum_{f=0}^{\infty} f \cdot p_F(f), \tag{13}$$

where $\lambda \cdot T$ is the expected number of blocks within the aggregation period. We can now express the savings of the aggregation protocol as

$$\Gamma = 1 - \frac{\mathbb{E}[F]}{\mathbb{E}\left[F^{(\mathrm{P2})}\right]} = \tag{14}$$

$$= 1 - \frac{\sum_{f=0}^{\infty} f \cdot p_F(f)}{\lambda \cdot T \cdot \left(H + l_{\mathrm{H}} + \sum_{a=0}^{\infty} a \cdot p_{D_{\mathrm{A}}|B}(a|1)\right)}. \tag{15}$$

V. EVALUATION

To validate our model and show the performance of the aggregation scheme, we have modified the Python implementation of Ethereum protocol, PyEthereum [32]. The system, parametrized as listed in Table II, includes a randomly generated blockchain. This is obtained by generating accounts that contain random information, with size l_a bits, and inserting them in a newly initialized blockchain database. For the statistical characterization of accounts updates, we take as reference the Ethereum main network as described in the following section.

A. Statistical characterization of accounts updates

The statistics of account updates plays a fundamental role in the design and evaluation of blockchain protocols. We base our evaluation on the Ethereum main network dataset [33], by analyzing the activity during blocks numbered from 5.1 to 6.4 million. Fig. 6 shows the frequency of updates of the 10^4 most updated accounts, indexed in descending order of their updates frequencies. To extract this metric, we do not distinguish between transactions from/to the accounts, or consider if there are multiple transaction involving one account in the same block. We model the relative frequency of updates of account j according to the broken power-law:

$$p_j = \begin{cases} \alpha_1 j^{\alpha_2} & \text{if } j \le \alpha_3, \\ \alpha_3^{\alpha_2-\alpha_4} \alpha_1 j^{\alpha_4} & \text{otherwise.} \end{cases}$$

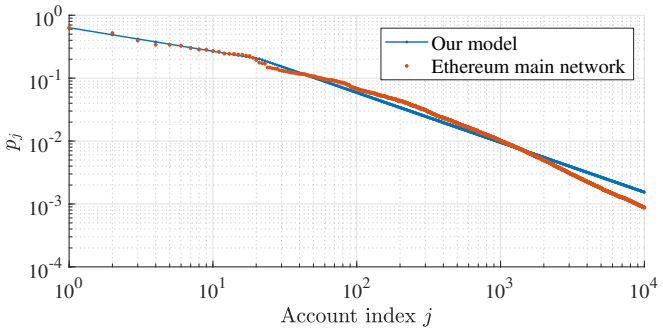

Fig. 6. Relative frequency of updates for the most active accounts, in log-log scale.

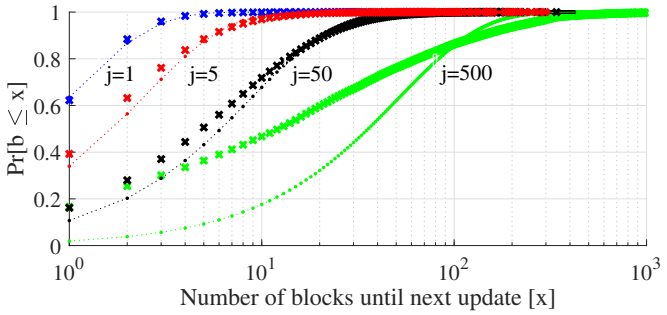

Fig. 7. Comparison of empirical CDF of accounts (represented with crosses), with index j, with the CDF of geometrical distribution (represented with dots).

We opt for this function, instead of the plain power-law, adopted e.g. in [34], [35], because the most frequent accounts are associated to web services that provide currency exchanges and are updated at similar rates. The least squares fit gives $\alpha_1 = 0.63$, $\alpha_2 = -0.37$, $\alpha_3 = 21$, $\alpha_4 = -0.79$. This function, also shown in Fig. 6, is used to generate the relative frequencies for our "synthetic" blockchain in the evaluation.

In addition to obtaining the account update probabilities, we inspect the accuracy of modelling the number of blocks between two account updates as a geometric distribution as assumed in our model. We compare the empirical cumulative density function (CDF) of an account, j, to the CDF of a geometric distribution with parameter p_j. Fig. 7 shows the results obtained for some accounts of the data set. It results that the assumption only holds for frequently updated accounts. This behaviour should be taken into account in future works.

B. Validation of Merkle-Patricia proofs length

Since the analysis of the length of a Merkle-Patricia proof is based on the assumption that the tree is perfectly balanced, the analysis is validated by comparing analytical results both to numerical results obtained from a perfectly balanced tree, and to measurements obtained from the Merkle-Patricia tree implementation in PyEthereum, which is in general unbalanced, see Fig. 8. In particular, Fig. 8(a) compares the analytical expression for the average number of nodes that compose a PoMI with the experimental data obtained from PyEthereum and with numerical results. The results show that the analytical expression fits the numerical results obtained for a balanced tree; at the same time, it overestimates the average number

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2906615, IEEE Internet of Things Journal

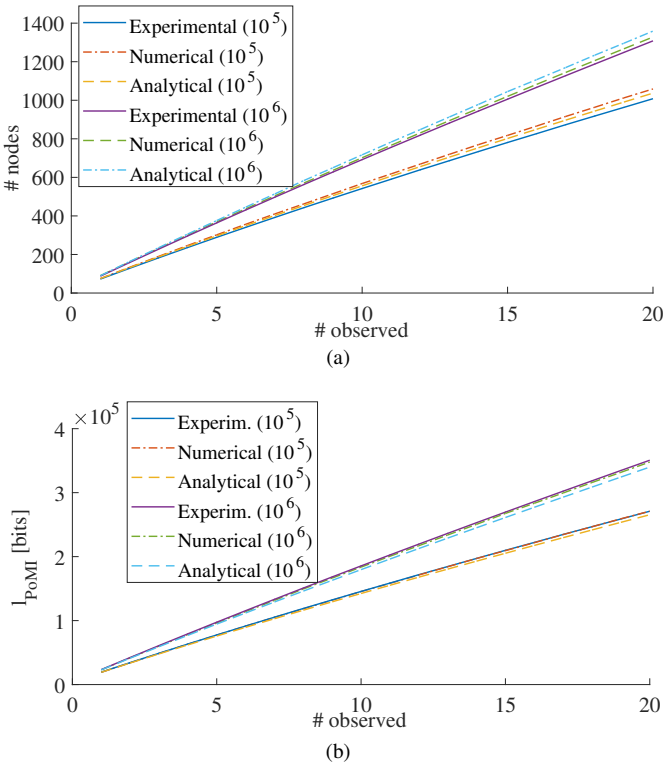

Fig. 8. Comparison of analytical approximation, numerical and experimental results for the (a) number of nodes needed in the PoMI and (b) its length.

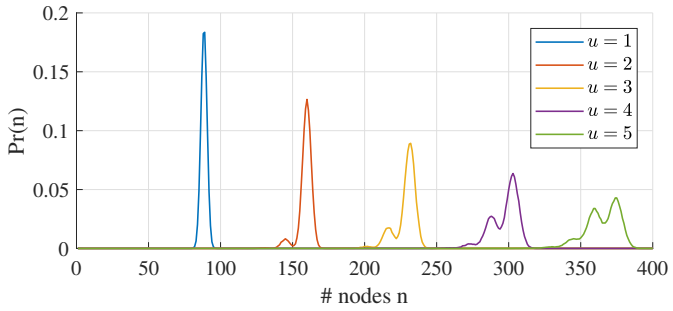

Fig. 9. Probability distribution of the number of nodes in a PoMI, for different number of observed events u, obtained via experiment.

of nodes, needed for a PoMI, in the Ethereum system. This follows from the fact that the average depth of leaves in a Patricia tree is greater than in a balanced tree [31], implying that some internal levels might not be completely populated, hence the slight reduction in number of nodes needed for the PoMI. Fig. 8(b) compares the length of the PoMI. For the numerical and analytical results, each node is represented by the corresponding hash value, while for the experimental data the PoMI data structure is represented with Recursive Length Prefix (RLP) [21]. The RLP also contains information about the structure of the tree, therefore introducing a small overhead. For this reason, the length for the experimental data is slightly larger than the ones of the numerical and analytical results.

The experimental setup also permits to characterize the distribution of the number of nodes in a PoMI, see Fig. 9,

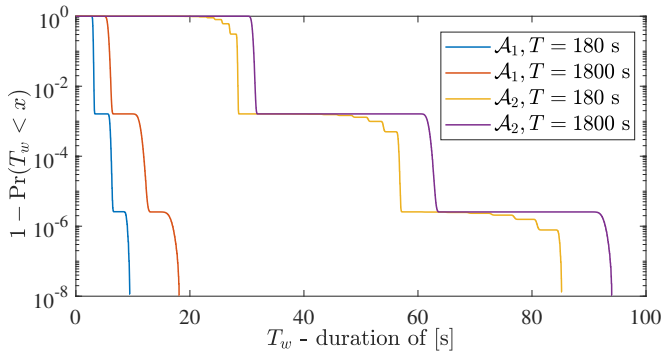

Fig. 10. Complementary CDF of T_w for two different intervals.

in which we show the results obtained for a blockchain with $\eta = 6$ levels, completely filled, therefore containing $L^6 = 16^6$ accounts, where L is the maximum number of children of a node. The relative position of the accounts in the tree clearly impacts the length of their PoMI and, hence, the communication cost of transmitting them. In addition, the results provide insights on the consequence of using the expected length of PoMI, instead of its distribution, in (6). As the variance of the PoMI distribution remarkably increases with the number of included accounts, u, the precision of the approximation is decreased. On the other hand, its contribution to the total length of the payload, D_A, is counterbalanced by the weight of accounts' data structure, which becomes predominant. This is shown in details in the following text.

C. Performance of the aggregation protocol

We consider a scenario in which the device is connected to BNs via a communication link parameterized as in Table II. We remark that if the device is solely interested in observing accounts that are updated sporadically, the aggregation protocol only provides reduction of the communication overhead (the frame headers). Therefore, we focus the evaluation on the case where the device observes *active* accounts. An account, j, is considered active if it is updated at least once every T seconds, with probability P_{A}, i.e.

$$1 - \left(1 - p_j\right)^{\lceil T/T_{\text{B}} \rceil} \geq P_{\text{A}} \tag{16}$$

In the rest of the work, we set $P_{\text{A}} = 0.9$. By requesting updates about more active accounts, than those of actual interest, the device can increase its privacy, at the cost of downloading unnecessary information. The application is further discussed in Sec. VI-C.

1) Duty cycle trade-offs: Fig. 10 reports the complementary CDF of the duration of the transmission, T_w, for two deterministic sets of observed accounts: $\mathcal{A}_1 = \{1, 2\}$, that contains the two most frequently updated ones, and $\mathcal{A}_2 = \{j | 20 < j \leq 41\}$, containing the 20 less active accounts when $T = 180$ s.[6] The sets are formed to illustrate two interesting limit scenarios. The probability of channel outage, derived from R and γ of Table II, is $1.6 \cdot 10^{-3}$. The number of observed accounts, and

[6]According to our definition, see (16), there are 41 active accounts for $T = 180$ s and 695 for $T = 1800$ s.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2906615, IEEE Internet of Things Journal

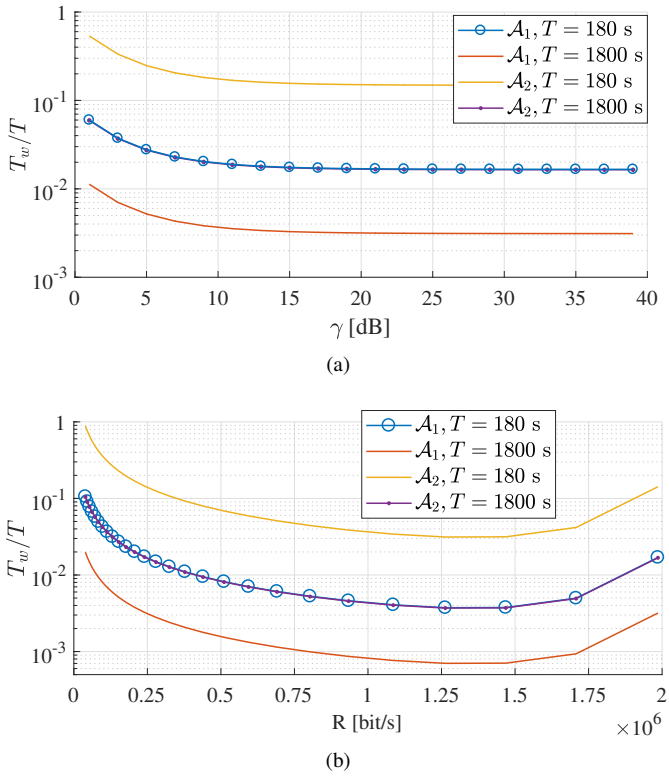

Fig. 11. Duty cycle of the device (a) for different values of SNR and (b) for different rates.

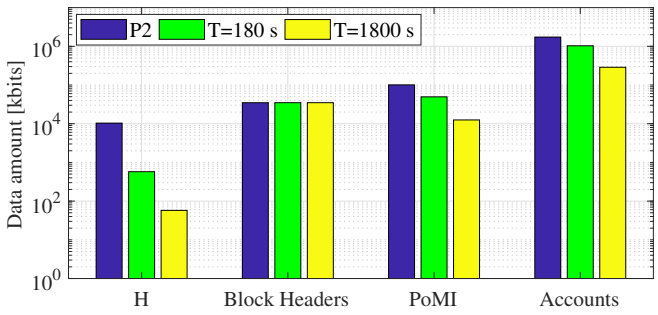

Fig. 12. Amount of information downloaded for $\mathcal{A}_3$, during 24 hours, for protocol P2 and for protocol with aggregation with different values of T.

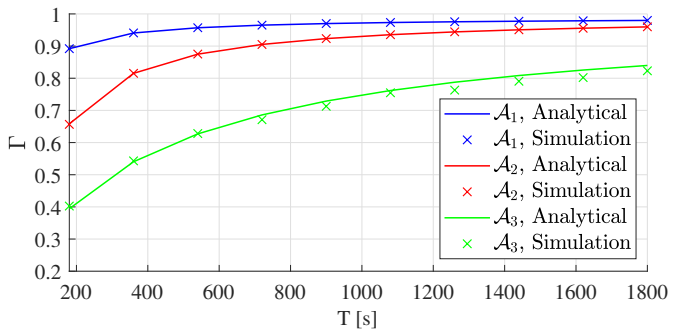

Fig. 13. Gain of the aggregation protocol, analytical expression and simulation.

their statistics, clearly plays a central role in shaping the CDF, as the size of the account data structure is much bigger than the size of a block header, see Table II.

The study of the duty cycle of the device, reported in Fig. 11, covers several fundamental trade-offs. Fig. 11(a) shows that the duty cycle decreases when the channel quality (SNR) increases, due to the reduced number of retransmissions, and saturates for high values of SNR, as retransmissions are not likely to happen. A possible strategy that can be adopted by the IoT device, to reduce its duty cycle, is to update less frequently, i.e. increase T, or reduce the set of observed accounts. Fig. 11(b) reports the duty cycle as function of the transmission rate of the wireless link. At low rates, the duty cycle of the device is drastically increased. On the other hand, selecting a high rate causes transmission failures and therefore retransmissions which become dominant when the rate reaches a certain level.

2) Communication cost: The rest of the results focuses on how the different parts of the frame contribute to its total length and on the aggregation gain, defined in Sec. IV-C. We construct a set $\mathcal{A}_3$ containing $|\mathcal{A}_3| = 20$ accounts, by randomly selecting among those that are active during $T = 1800$ s. It should be noted that $\mathcal{A}_2$ is a possible realization of $\mathcal{A}_3$. Fig. 12 shows the amount of information, downloaded during 24 hours of execution, for different values of T and different realizations of accounts in $\mathcal{A}_3$. In this scenario, there are no retransmissions; their effect would be a proportional increase in all the quantities. The figure shows that most of communication cost is due to the size of the account data structures, which is an order of magnitude higher w.r.t. the size of PoMI, and two order of magnitudes higher than the size of the block headers and communication protocol headers.

The aggregation gain, Γ, is shown in Fig. 13 for several values of the aggregation period, T, and compared with a simulation of the system. The figure shows a good match between the simulation and the analytics, and that the gain is remarkable, even for small values of T. As $T \to \infty$ the observed accounts are updated almost surely during a period, and will be downloaded in the next transmission. This causes the gain to increase linearly when T is large.

VI. Discussion

In this section, we discuss limitations and extensions of the aggregation protocol. We also present examples of potential applications.

A. Limitations of the aggregation protocol

The period of the aggregation protocol inherently defines the maximum delay after which the IoT device is informed about the new state of an account. However, it should be considered that, due to the possibility of blockchain forks [1], the transactions included in blockchains achieve finality (i.e., they can be considered immutable) only after a certain delay. For instance, in Ethereum main network it is common to wait more than ten block periods before making use of the information contained in validated transactions [36], and the block period is on average 14 seconds. In this context, there is no difference between retaining the information at the BNs

for $T = 140$ s, or delivering it to the IoT device that waits the same time before making use of it.

A second limitation is given by the relative frequency with which accounts are updated in the network. If the accounts observed by the device are rarely updated, the aggregation protocol provides limited advantage over the legacy protocol P2. In fact, the information that is downloaded is restrained to block headers, that are generated by the blockchain network at a constant rate. The aggregation provides a reduction of the communication system overhead, but it is not remarkable, see Fig. 12. Better approaches to deal with such scenario are the reduction of block header information sent to the device [16], [27] or the use of protocols of class P3, see Sec. II-C.

B. Remarks on possibilities to further reduce of the communication cost

We briefly discuss possible directions for a further reduction of the communication cost for IoT lightweight clients, based on the insights provided by the evaluation of the protocol. The size of the accounts' data structures has shown a prominent impact on the amount of transmitted data. This can be reduced with several approaches, for example (i) by keeping their size as small as possible, avoiding the storage of unnecessary information; (ii) by compressing the account information before sending it; (iii) by only sending the portion of account structure that has changed.[7]

A completely different approach is to send the updates by means of the transactions tree, when the size of a transaction is lower than the one of the account's state. In addition, while the state tree grows with the number of accounts, the transactions tree size is limited by the block size. This option, mentioned in Sec. II-C, has not been considered in this paper, as it does not provide aggregation gain. Future works can consider this extension by (i) including in the system model the statistics of the number of transactions, that modify an account, in a single block; (ii) considering also the contribution of the transactions tree to the size of P_{D_A} in (5); (iii) finding a strategy to decide if sending the update under the form of updated state, or as collection of transactions.

C. Example applications

1) Support of periodic schemes: The aggregation protocol finds a direct application to schemes that run infrequently but periodically. This is the case in distributed tertiary controllers in Microgrids [2] and accounting of costs in large-scale power systems [5], which are typically executed every 15-30 minutes.

2) Privacy of IoT device: We conclude this section by providing a practical application of the aggregation protocol to improve the privacy of a device. Consider a scenario in which the IoT device is solely interested in observing one (active) account, indexed as $j^\star$. However, to keep $j^\star$ private, it requests updates about additional accounts, in which it is not interested, from BNs (privacy by obfuscation) [20]. A malicious BN is aware that the IoT device is interested in one account and applies an outlier detection technique to find it. In the presented model, the only feature available to the BN is the relative frequency of update of accounts. For both sides (device and BN), it is reasonable to assume that the set of observed accounts, indicated as $\mathcal{A}_4$, only contains active accounts, since non active accounts would be excluded by the outlier detection.

[7]The latter approach is already included in Ethereum protocol, by representing the state of the account itself with a Merkle-Patricia tree [21].

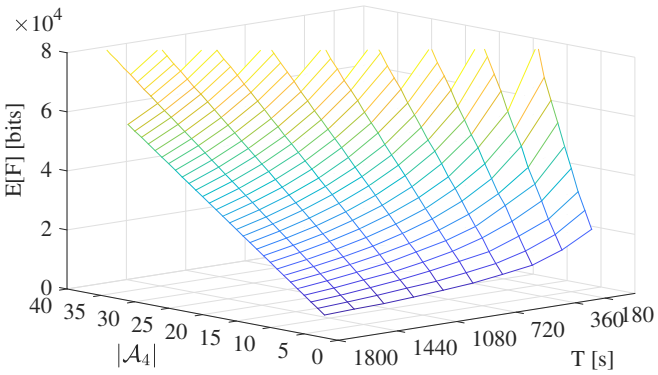

Fig. 14. Communication cost as function of $|\mathcal{A}_4|$ and T.

Based on these considerations, the IoT device constructs $\mathcal{A}_4$ by adding $j^\star$ and other random active accounts. The construction starts with $\mathcal{A}_4 = \{j^\star\}$, then $|\mathcal{A}_4|$ is iteratively incremented. At each iteration, the set of active accounts is split in $|\mathcal{A}_4|$ segments and one account is randomly picked from each segment ($j^\star$ is always picked among the accounts in its segment). The iteration is repeated until the tolerated communication cost, expressed by (13), is reached. Finally, $\mathcal{A}_4$ is sent to the BN. There is a trade-off between the delay, given by the aggregation protocol, and privacy, i.e. $|\mathcal{A}_4|$. This trade-off is shown in Fig. 14, for different tolerated communication costs $\mathbb{E}[F]$, and $j^\star = 41$ (that is an active account). A further improvement, not implemented in this paper, is impose that accounts in $\mathcal{A}_4$ should be located in proximity of each other in the state tree. In fact, this provides shorter PoMI and therefore lower communication cost, see Fig. 9.

VII. Conclusion

In this paper, we have investigated what is the communication cost of sending blockchain information to Ethereum-like lightweight clients. A novel aggregation scheme has been proposed that has the potential to obtain a lower communication cost, at the expense of higher information delay, or availability of information, at the application layer. The analysis of the scheme shows the probability distributions of the data structures exchanged over the wireless link, and their impact on the total downlink budget.

Finally, the results show that, if the statistics of account updates and the channel state are known, the lightweight clients can construct a list of events of interest that provides a predictable average communication cost. The example application illustrates how to apply our findings to improve the privacy of IoT devices. The guidelines presented in this paper can be applied to design more advanced blockchain lightweight protocols.

APPENDIX A
DERIVATION OF THE EXPECTED NUMBER OF NODES IN A POMI

Under the relaxation described in Sec. IV, the probability that an arbitrary node at level h is ancestor to one of u modified leaf nodes (denoted by the binary random variable X_h) is $\Pr(X_h = 1|N_{h-1} = n_{h-1}, U = u) = (1 - 1/(Ln_{h-1}))^u$, where N_{h-1} is the number of nodes at level $h-1$ that are ancestors to a modified leaf node and L is the branching factor of the tree. Since X_h is a binary random variable, $\mathbb{E}_{X_h}[X_h|N_{h-1} = n_{h-1}, U = u] = \Pr(X_h = 1|N_{h-1} = n_{h-1}, U = u)$, and the expected total number of ancestor nodes at level h is $\mathbb{E}_{N_h}[N_h|N_{h-1} = n_{h-1}, U = u] = Ln_{h-1} \cdot \mathbb{E}_{X_h}[X_h|N_{h-1} = n_{h-1}, U = u]$. By the law of total expectation,

$$\begin{aligned}\mathbb{E}_{N_h}[N_h|U=u] &= \mathbb{E}_{N_{h-1}}\left[\mathbb{E}_{N_h}[N_h|N_{h-1}, U=u]\,|U=u\right] \\ &= \mathbb{E}_{N_{h-1}}\left[LN_{h-1}\cdot\mathbb{E}_{X_h}[X_h|N_{h-1}, U=u]\,|U=u\right] \\ &= \mathbb{E}_{N_{h-1}}\left[LN_{h-1}\cdot\left(1-\frac{1}{LN_{h-1}}\right)^u \,\middle|\, U=u\right].\end{aligned}$$

By applying a first-order Taylor expansion at $\mathbb{E}_{N_{h-1}}[N_{h-1}|U = u]$ we obtain

$$\mathbb{E}_{N_h}[N_h|U=u] \approx L\,\mathbb{E}_{N_{h-1}}[n_{h-1}|U=u]\left(1-\frac{1}{L\mathbb{E}_{N_{h-1}}[n_{h-1}|U=u]}\right)^u. \quad (17)$$

Denoting $\bar{N}_h(u) = \mathbb{E}_{N_h}[N_h|U = u]$ and using the fact that the PoMI will always contain a single root to define $\bar{N}_0(u) = 1$, (17) can be obtained by recursion. The approximated number of nodes for a complete PoMI, in a tree of height η, is the sum of the nodes required at each level which yields (7):

$$\bar{N}_\eta(u) = \sum_{h=1}^{\eta} \bar{N}_h(u).$$

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *[Online]. Available: https://bitcoin.org/bitcoin.pdf*, 2008, accessed: 2019-02-22.

[2] P. Danzi, M. Angjelichinoski, Č. Stefanović, and P. Popovski, "Distributed proportional-fairness control in microgrids via blockchain smart contracts," in *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct 2017, pp. 45–51.

[3] J. Horta, D. Kofman, D. Menga, and A. Silva, "Novel market approach for locally balancing renewable energy production and flexible demand," in *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct 2017, pp. 533–539.

[4] E. Münsing, J. Mather, and S. Moura, "Blockchains for decentralized optimization of energy resources in microgrid networks," in *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE, 2017, pp. 2164–2171.

[5] P. Danzi *et al.*, "Blockchain-based and multi-layered electricity imbalance settlement architecture," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Oct. 2018, pp. 1–7.

[6] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[7] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2018.

[8] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2018.

[9] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, 2019.

[10] P. Danzi and P. Popovski, "Towards blockchain networks tailored to iot devices," *IEEE Blockchain Technical Briefs [Online]. Available: https://blockchain.ieee.org/technicalbriefs/january-2019/towards-blockchain-networks-tailored-to-iot-devices*, accessed: 2019-02-22.

[11] Y. Sun, H. Luo, and S. K. Das, "A trust-based framework for fault-tolerant data aggregation in wireless multimedia sensor networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 785–797, 2012.

[12] F. Al-Turjman, E. Ever, and H. Zahmatkesh, "Small cells in the forthcoming 5g/iot: Traffic modelling and deployment overview," *IEEE Communications Surveys & Tutorials*, 2018.

[13] P. Danzi, A. E. Kalor, C. Stefanovic, and P. Popovski, "Analysis of the communication traffic for blockchain synchronization of iot devices," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[14] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "A first look at cellular machine-to-machine traffic: large scale measurement and characterization," *ACM SIGMETRICS performance evaluation review*, vol. 40, no. 1, pp. 65–76, 2012.

[15] C. Bockelmann, N. Pratas, H. Nikopour, K. Au, T. Svensson, C. Stefanovic, P. Popovski, and A. Dekorsy, "Massive machine-type communications in 5g: Physical and mac-layer solutions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 59–65, 2016.

[16] A. Kiayias, A. Miller, and D. Zindros, "Non-interactive proofs of proof-of-work," *Cryptology ePrint Archive, Report 2017/963, 2017. Accessed: 2019-02-22*, 2017.

[17] A. Palai, M. Vora, and A. Shah, "Empowering light nodes in blockchains with block summarization," in *New Technologies, Mobility and Security (NTMS), 2018 9th IFIP International Conference on*. IEEE, 2018, pp. 1–5.

[18] A. Schoedon, "The ethereum-blockchain size will not exceed 1tb anytime soon," *[Online]. Available: https://dev.to/5chdn/the-ethereum-blockchain-size-will-not-exceed-1tb-anytime-soon-58a*, accessed: 2019-02-22.

[19] "Slock.it incubed client," *[Online]. Available: https://slock.it/incubed.html*, accessed: 2019-02-22.

[20] D. Gruber *et al.*, "Unifying lightweight blockchain client implementations," *NDSS 2018 - Workshop on Decentralized IoT Security and Standards*, 2018.

[21] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *[Online]. Available: http://gavwood.com/paper.pdf*, 2014, accessed: 2019-02-22.

[22] O. Alphand *et al.*, "Iotchain: A blockchain security architecture for the internet of things," in *Wireless Communications and Networking Conference (WCNC), 2018 IEEE*. IEEE, 2018, pp. 1–6.

[23] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[24] V. Buterin, "Merkling in ethereum," *[Online]. Available: https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/*, accessed: 2019-02-22.

[25] M. Al-Bassam *et al.*, "Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities," *ArXiv preprint arXiv:1809.09044v1*, 2018.

[26] D. R. Morrison, "Patricia - practical algorithm to retrieve information coded in alphanumeric," *Journal of the ACM (JACM)*, vol. 15, no. 4, pp. 514–534, 1968.

[27] F. Zsolt, "Client side flow control model for the les protocol," *[Online]. Available: https://github.com/zsfelfoldi/go-ethereum/wiki/Client-Side-Flow-Control-model-for-the-LES-protocol*, 2016, accessed: 2019-02-22.

[28] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7492–7508, Nov 2017.

[29] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[30] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10.

[31] B. Rais, P. Jacquet, and W. Szpankowski, "Limiting distribution for the depth in patricia tries," *SIAM Journal on Discrete Mathematics*, vol. 6, no. 2, pp. 197–213, 1993.

[32] "Pyethereum," *[Online]. Available: https://github.com/ethereum/pyethereum*, accessed: 2019-02-22.

[33] "Google cloud blog," *[Online]. Available: https://cloud.google.com/blog/products/data-analytics/ethereum-bigquery-public-dataset-smart-contract-analytics*, accessed: 2019-02-22.
[34] M. Lischke and B. Fabian, "Analyzing the bitcoin network: The first four years," *Future Internet*, vol. 8, no. 1, p. 7, 2016.
[35] S. Somin, G. Gordon, and Y. Altshuler, "Social Signals in the Ethereum Trading Network," *Preprint available: https://arxiv.org/abs/1805.12097*, 2018.
[36] R. Yasaweerasinghelage, M. Staples, and I. Weber, "Predicting latency of blockchain-based systems using architectural modelling and simulation," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 253–256.

Pietro Danzi (S'16) is a doctoral student in Wireless Communications at Aalborg University, Denmark, where he received a Marie Skłodowska-Curie fellowship as Early Stage Researcher. Previously, he obtained a M.Sc degree in Telecommunication Engineering from Università degli Studi di Padova, Italy. His current interests include machine-type communication protocols, blockchain protocols and cybersecurity for smart grids.

Anders E. Kalør (S'17) received the B.Sc. degree in computer engineering and the M.Sc. degree in networks and distributed systems from Aalborg University, Denmark, in 2015 and 2017, respectively. He is currently pursuing a Ph.D. degree in the area of wireless communications and networking at Aalborg University. His research interests include communication theory, MAC layer design for wireless systems, and networking.

Čedomir Stefanović (S'04–M'11–SM'17) received his Dipl.Ing., Mr.Ing., and Ph.D. degrees in electrical engineering from the University of Novi Sad, Serbia. He is currently an Associate Professor with the Department of Electronic Systems, Aalborg University, Denmark. He is involved in several national and European Union projects related to the Internet of Things and fifth-generation communications. His research interests include communication theory, wireless and smart grid communications.

Petar Popovski (S'97–A'98–M'04–SM'10–F'16) is a Professor of Wireless Communications with Aalborg University. He received his Dipl. Ing and Magister Ing. degrees in communication engineering from the University of Sts. Cyril and Methodius in Skopje and the Ph.D. degree from Aalborg University in 2005. He has over 300 publications in journals, conference proceedings, and edited books and he is featured in the list of Highly Cited Researchers 2018, compiled by Web of Science. He holds over 30 patents and patent applications. He received an ERC Consolidator Grant (2015), the Danish Elite Researcher award (2016), IEEE Fred W. Ellersick prize (2016) and IEEE Stephen O. Rice prize (2018). He is currently a Steering Committee Member of IEEE SmartGridComm and IEEE Transactions on Green Communications and Networking. He previously served as a Steering Committee Member of the IEEE INTERNET OF THINGS JOURNAL. He is currently an Area Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. Prof. Popovski is the General Chair for IEEE SmartGridComm 2018 and IEEE Communication Theory Workshop 2019. From 2019, he is also a Member-at-Large of the Board of Governors of the IEEE Communications Society. His research interests are in the area of wireless communication and communication theory.