Blockchain adapted to IoT via green mining and fractal Proof of Work

Philippe Jacquet

Blockchain adapted to IoT via green mining and fractal Proof of Work

Philippe Jacquet
IEEE Fellow
Inria
Saclay, France
Email: philippe.jacquet@inria.fr

***Abstract*—Blockchain applications continue to grow in popularity, but their energy costs are clearly becoming unsustainable. In most cases, the primary cost comes from the amount of energy required for proof-of-work (PoW). Here we study the application of blockchains to the IoT, where most devices are underpowered and would not support the energy cost of proof of work. PoW was originally intended for two main uses: block moderation and protecting the blockchain from tampering. For IoT we propose to replace the expensive moderation of PoW with the proposed energy-efficient green mining [6]. The blockchain will be protected by a fractal difficulty PoW. One of the results of this paper is the proof that the average mining time in fractal PoW actually depends only on the average difficulty. This crucial property will allow low-difficulty PoWs to be reserved for devices with low computational capacity, while higher difficulties will be reserved for devices with the highest computational power. The consequence is to give equal opportunity to objects with low computational power compared to objects with high computational power.**

I. Introduction

The concept of blockchain was popularized by the crypto-currency Bitcoin. However, one of Bitcoin's drawbacks is that moderating the rate of block mining and protecting the ledger from tampering relies on proof-of-work (PoW). Block mining is subject to fierce competition. To prevent too many simultaneous blocks from being mined and offered for confirmation, Bitcoin requires proof of work by forcing block miners to perform a large number of CPU cycles via iterated hash calculations before submitting their block. Proof-of-work ensures rarity and uniqueness, helping to reduce the mining rate of individual blocks.

In addition, the PoW protects the ledger against forgery by forcing the forger to calculate a PoW for each forged block in order to be accepted into the blockchain. This entails an unsustainable cost, since the forgery must be completed to the end of the blockchain, and this must be achieved in a very short time.

However, average PoW in Bitcoin currently costs more than 10^{22} computing cycles and mobilises a total instantaneous computing power of 3.10^{20} calculated hash values per second which would be excessive if used for IoT (e.g., [1], [2], [5], [7]). Our contribution is to show how to modify the PoW to make it suitable for IoT and make substancial energy savings.

II. Blockchain for IoT, contribution of this paper

In Bitcoin each block contains the references of several transactions (up to one thousand) which are collected in a shared pool. Mining a block is rewarded in Bitcoins by the protocol. Applied to IoT, the transaction pool is superfluous, since there is no need of incentive to mine blocks other than that the miners want to share their transactions without delay. We can therefore expect smaller, and therefore much more frequent, blocks.

Our contribution will focus on the block mining process, irrespective of how and where blocks are constructed. However, we will keep in mind that the sources of the blocks, the transactions and the miners, will be very diverse and will have very different computing powers.

We assume a network composed of IoT devices whose purpose is to maintain the digital twin of a large complex system such as an urban area, a large factory, a battlefield or a threatened natural area. We consider that such a network will give rise to three sources of data mined in the blockchain:

- the real time data originated from the monitored area to be reflected in the digital twin;
- the software used to handle these data;
- the protocol updates of the network connecting the IoT.

Our two contributions are:

- on relieving the PoW of the task of moderating the flow of mining blocks;
- using the PoW with fractal difficulty adapted to the computational capabilities of the miners.

The problem of moderating block mining arises when the number of blocks competing for extraction is highly dynamic. In this situation, the network and the ledger may be subject to more or less temporary congestion. More specifically, if millions of miners were to compete for access to the blockchain, this would generate collisions and huge conflicts, with a risk of collapse in a network made up of low-speed wireless components, which were probably not designed to carry millions of access requests. Under these conditions the moderation via PoW is the main source of energy waste and in passing makes it unpratical for device with low computing power.

A. The green leader election

Our proposition described in section III is to replace the block moderation by proof of work by a moderation by green leader election [8]. The algorithm is described in [6]. The action which consists into among n contenders to select a winner or a small set of leaders is called "leader election". A classic algorithm uses a fixed parameter p, called the "decimation probability". An election proceeds in different rounds. At the first round, the n contenders try to access the network. If their number is too high for a selection, *i.e.* leads to a fruitless collision, then they decimate themselves by a random selection with probability p, and the survivors access the network. If their number is still too high they proceed to a third round and so forth until the survivors are in an enough small number to select. Since the average number of survivors at round ℓ is $p^\ell n$ which decays like a downward stairway (see figure 1, left), then one would expect that the election process will take $\frac{\log n}{\log(1/p)}$ to converge. The problem is that meanwhile the network would have supported the pressure of $n + pn + \ldots + p^\ell n$ individual accesses in a short interval of time, which would be too much if we suppose large values for n (say one million).

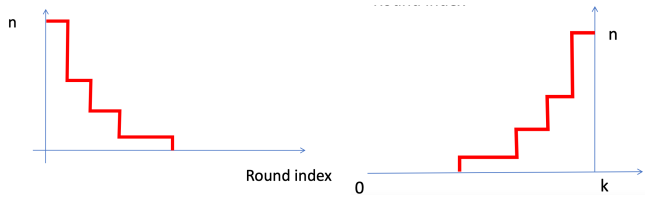

Fig. 1: Classic leader election via downward stairway (left), green leader election via upward stairway (right)

In [8] is described an inverted leader election, called a green leader election, which consists of inverting the direct leader election described above. We assume that the maximum number of rounds is less than a given integer k, which leads us to assume that $n \ll p^{-k}$. In the first round (let's say round 0), candidates access the network with probability p^k, if none access it, a new round takes place where candidates access it with probability p^{k-1}, and so on. The probability increases geometrically with each new round until some candidates actually access the network. Figure 1, on the right, shows the average number of accesses to the network as a function of round, resembling an ascending staircase. In the ℓ round, the probability of access is $p^{k-\ell}$ and if any winners access the network, this ends the election, since the number of potential leaders is already low enough.

The property of moderation through the green election of a leader is that no access to the network takes place until a winner has been designated. There can be several simultaneous winners, but their number will be limited whatever the initial number of competing blocks. Of course, as long as this number is less than a large but arbitrary number (several billion, or billions of billions). The consequence of the green election is that no PoW calculations are performed for election purposes which considerably reduce the energy footprint of the blockchain and make it practical for low powered miners.

The green election algorithm operates in epochs, with each epoch ending with the extraction of a winning block. An epoch is divided into several rounds, each lasting a few seconds, depending on the precision of the time synchronization. All rounds are empty, except for the last one, which ends the epoch (otherwise, the epoch ends when the maximum number of rounds is reached). The protocol therefore depends on three parameters fixed by design: (i) the probability of decimation p, (ii) the maximum number k of rounds per election, (iii) the duration of the rounds. The probability of access per round will be given by an implicit target to be respected by the block's election hash value, as will be explained in section III. In [6], the following theorem is proved (albeit with some variations in notation):

Theorem 1: The average number of network accesses per green leader election for an initial contention of n blocks is upper bounded by the quantity $np^k + A$ with $A = \frac{1/p}{\log(1/p)} \sum_{k\in\mathbb{Z}} |\Gamma(1 + 2ik\pi/\log p)|$, where $\Gamma(.)$ is the Euler "Gamma" function.

We remark that if $n \ll p^{-k}$ then the quantity np^k is negligible. For $k = 16$, $p = 1/4$ we have $A \approx 2.9104$. For $k = 8$, $p = 1/16$ and $A \approx 7.0608$. In both case $p^{-k} = 2^{32}$.

B. The variable PoW with fractal difficulty

Our second contribution to the blockchain for IoT is the fractal PoW and will be analysed in section IV. The block moderation by green election above described, enables to leave the use the PoW for the blockchain protection, regardless on how blocks are moderated. In this case a PoW with variable difficulty D is interesting, since for a blockchain of length L the resilience to forgery is asymptotically equal to $E[D]L$, thus the same resilience as for a blockchain where the difficulty would be identical for all blocks and equal to $E[D]$. Meanwhile miners with low CPU will have equal chance to mine their blocks. It is not question that the miners would compete with different difficulties at the same time, since in this case the miners with low D would have a permanent undue priority. Indeed the difficulty will be fixed as a function of the last mined block so that all miners will have the same PoW difficulty in their attempts in mining their own block. The difficulty is assumed to be in seconds of a normalized CPU, but in the block format it will be expressed as a target for the final block hash value.

The concept of variable PoW difficulty is not new, but to the best of the authors' knowledge, most proposals [10] introduce protocols for smoothly adapting and adjusting block difficulty to the computational capacity of the miner population. It has never been proposed to finely modify PoW difficulty from one block to the next as part of a dynamic process, such as a "fractal" stationary process. Figure 2 shows that for a classical blockchain, each link (block) will be mined with a fixed root depth (left). With variable difficulty, some links will have a shorter or longer root, so that the average difficulty remains the same.

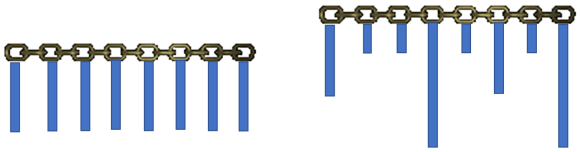

Fig. 2: Fixed difficulty mining (left), fractal difficulty mining (right)

Theorem 2: Let λ be the Poisson rate of creation of new block per second. Assuming that every miner has the same computational capacity, the average number of miners in simultaneous PoW for the next block is equal to $\lambda E[D]$. The average time a miner is in PoW before mining its own block is $E[D]$

We assume that the PoW difficulty for the next block is determined by the previous block. It can be determined by the block sequence number. But it's quite possible for a miner to experience different successive PoW difficulties before successfully mining his own block.

The consequence of this theorem is that varying difficulty has no impact on the first moment of the mining waiting time, which is a remarkable result. In the context of this theorem, we assume that all miners have the same computing power, but in real-life situations, computing power will differ between miners and, in practice, miners with low computing power will have no chance when difficulty is high, but will have "more equal" chances when difficulty is low (in fact, difficulty may be zero in some cases).

C. The block format

The proposed block format is:

field #	value
1	previous block hash
2	ID & date & sequence number
3	next block election target
4	election hash value
5	payload
...	...
6	next block difficulty target
7	nonce
8	final block hash value

For the green leader election algorithm, the miner will calculate the election hash value (field 4) from fields 1, 2 and 3. The payload is intentionally left outside the scope of the election hash value to prevent devices with powerful CPUs from modifying their payload at will, if it is not selected from a common repository. The nonce is already excluded for the same reason (see section III). For the variable-difficulty part, a nonce will be added after the election hash value, after the payload fields and before the final hash value of the block to match the required difficulty (the hash value must be less than or equal to a required target). A block is mined when the following two conditions are simultaneously met:

- (i) its election hash value corresponds to the election target of the current round, the round index is calculated as the ratio between the time elapsed since the date of the last mined block and the round delay fixed by convention
- (ii) the second hash value fits with the current demanded PoW difficulty.

The condition (i) is not affected by a nonce value, so that miner with a powerful CPU won't take the lead over miner with slower CPU.

The election target of the next block is also predetermined by the protocol design parameters and its display could be optional. Displaying the difficulty target of the next block could also be optional, as it should be a specified function of the block sequence number (which is the number of the previous block plus 1).

III. The green election

Each mined block requires its successor to have an election hash value lower than the election target value displayed. If no successor comes forward in the current round, the target interval is updated in the next round and corresponds to the previous target interval multiplied by $1/p$ (and rounded up to the nearest integers). The election hash value does not involve a nonce field or payload, so the hash value cannot be modified by the brute force of a powerful CPU. The election target increases until a winner is designated and a successor is finally extracted. The hash interval increases at a rate of $\frac{1}{p}$, typically 4 or 16. When the hash interval reaches 2^h, where h is the length of the hash field, all pending miners are called.

The protocol creates an ascending exponential staircase to determine the winner(s) of the election. The steps are enumerated by $\ell = 0, 1, \ldots, k$. The initial call corresponds to the smallest election target which is $\lfloor 2^{h+1} p^k \rfloor - 1$, h being the number of bits in the hash value (*i.e.* $h = 256$). The probability of a random hash value matching the initial election target is p^k. As long as the staircase levels increase, the target intervals evolve as $\lfloor 2^{h+1} p^{k-\ell} \rfloor - 1$ and the probability of a hash value falling within the interval is $p^{k-\ell}$.

Note that when $\ell = k$, the target of the election is the entire hash interval $2^{h+1} - 1$, so all values are called. We call this last round the *terminal* round. By convention, if no blocks are extracted during a terminal round, because no new blocks were ready, we assume that the terminal round is extended indefinitely and that we remain in the unrestricted call interval $[0, 2^{h+1} - 1]$, until a new block is extracted.

For $h = 256$, $k = 8$, $p = \frac{1}{16}$, the election target sequence is:

round number $0 \leq \ell \leq k = 8$	Access probabilities $p^{k-\ell}$	election target
0 (initial)	2^{-32}	$2^{224} - 1$
1	2^{-28}	$2^{228} - 1$
2	2^{-24}	$2^{232} - 1$
3	2^{-20}	$2^{236} - 1$
4	2^{-16}	$2^{240} - 1$
5	2^{-12}	$2^{244} - 1$
6	2^{-8}	$2^{248} - 1$
7	2^{-4}	$2^{232} - 1$
8	1	$2^{256} - 1$

As the value of the election target increases, the original [6] protocol has a central entity that produces empty blocks that update the target value. But the protocol also has the option of implicit calls without empty blocks, and the value of the call interval is determined by the date of the new block relative to the date of the previous one. Assuming that the round delay (say 10 seconds) and the call sequence are fixed by the protocol, we think this is the best solution. Of course, one could imagine tricking these implicit calls by dating the blocks in advance to give a miner more chances, but this would be easily thwarted by the clock consensus. Especially if priority is given to the smallest dates.

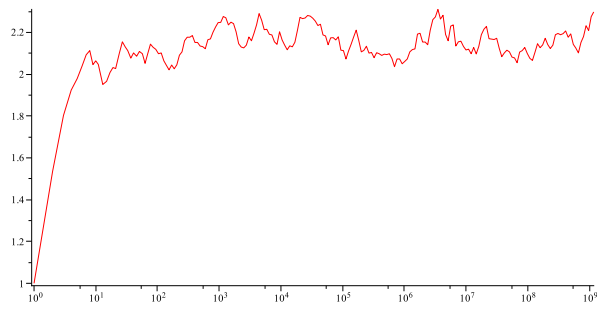

Fig. 3: Histogram of the simulations, $p = 1/4$, $k = 16$, borrowed from [6].

The figure 3 is borrowed from [6], it displays the average number of winners for various values of n for $k = 16$ and $p = 1/4$. Each point is supposed to have been simulated 1,000 times. We notice that the values are well below the theoretical upper-bound given by theorem 1 which is 2.9103.

IV. The fractal Proof-of-Work difficulty

In this section, we describe our main contribution, namely the apparent insensitivity of PoW performance to variable difficulty, as expressed in Theorem 2. The difficulty may simply be a predetermined function of the block index number; for example, indexes that are multiples of ten, hundred or thousand might be of greater difficulty. For the sake of completeness, we'll assume that the difficulty of the next block is recalled in the previous block, as described in the block format.

The difficulty display follows the same format as the election target, *i.e.* is an integer of the form $\lfloor 2^{h+1} \frac{1}{C} \rfloor - 1$ where C is the average number of hash values, calculated on a standardized CPU, that are required to reach the difficulty D (expressed in seconds). C is more or less proportional to D (except that when D is small, we limit C to 1).

The nonce is a field of h bits that is determined by the miner in such a way that the final hash value of the block is less than the difficulty target. It is assumed that there is no other way than brute force.

We present a theoretical analysis for fractal difficulty PoW for the first two moments of the number of simultaneous miners. We make a simplifying assumption that decouples the green election from the PoW. The coupling of the green election and the PoW should not overly affect the result of theorem 2 and will be the subject of a further work.

We do not comment on the strategy to be adopted by low-capacity devices in the event of variable PoW. For example, they could suspend their PoW in the terminal rounds, since they would no longer stand a chance against powerful CPUs (but be ready to resume if no mining takes place). To clarify, we assume that all miners have the same computing capacity, the case with different classes of computation powers and how it impacts the energy cost will be addressed in a forthcoming work. We assume that the green election is doing well its job and also, that only one winner is selected, which is not fully correct but is good enough for the present work.

A. Proof of theorem 2 with fixed difficulty D

Proof: The proof is quite simple. We assume that the green election and extraction of the winning block are instantaneous and that only one winner completes his PoW at any given time. The fact that the miner has had to modify certain fields of his own block several times during the PoW, because other blocks have been mined in the meantime, does not change the probability that the miner will complete his PoW within a given time interval $[t, t+dt]$, the probability being $\frac{dt}{D}$.

Thus, the average waiting time between the creation of a block and its extraction is D, and according to Little's formula, the average number of miners simultaneously in proof of work is λD. ■

The proof is rather trivial, but unfortunately it cannot be generalized to the case where the difficulty changes according to the last blocks mined. In this case, the average miner will alternate between different difficulties during his PoW, which will complicate the expression of his probability of unconditional termination in an arbitrary time interval $[t, t+dt]$.

We will introduce an alternative proof that can be extended to conditions of variable difficulty. The analysis is done at the epoch level. We assume that an epoch begins at time $t = 0$ after the last block has been extracted. Let $S_n(t)$ be the probability that at time t the epoch is still running and has exactly n blocks in PoW phase.

We assume that the rate at which new blocks enter PoW mode is a Poisson process with parameter λ per second. For convenience, we ignore the entanglement with the green election of leaders, as we'll assume that blocks are extracted during a terminal cycle and that extraction after PoW takes zero delay, as we explained in the introduction to the section. We also ignore the issues of time synchronization.

We have the following time evolution:

$$S_n(t+dt) = (1 - \lambda dt - \frac{n}{D}dt)S_n(t) + \lambda dt S_{n-1}(t).$$

which leads to the continuous markovian transition equation:

$$\partial_t S_n(t) = (S_{n-1}(t) - S_n(t))\lambda - \frac{n}{D}S_n(t).$$

where ∂_t indicates the derivative with respect to variable t. Let $S(z,t) = \sum_n S_n(t)z^n$, we have:

$$\partial_t S(z,t) = \lambda(z-1)S(z,t) - \frac{z}{D}\partial_z S(z,t).$$

Let $\sigma_0(z)$ the probability generating function of the initial queue length of the epoch. In other words $\sigma_0(z) = S(z,0)$ and the p.g.f. is naturally unitary, *i.e.* $\sigma_0(1) = 1$. We notice that $S(1,t)$ strictly decays; this is because the epoch may have stopped before t, and indeed $S(1,t)$ is the probability that the epoch lasts more than t seconds. In passing the average epoch duration is $\int_0^\infty S(1,t)dt$.

Let $S^*(z) = \int_0^\infty S(z,t)dt$, we have the identity (by integration by part):

$$-\sigma_0(z) = \lambda(z-1)S^*(z) - \frac{z}{D}\partial_z S^*(z).$$

When the epoch ends, a block is mined and the remaining blocks stop their current proof-of-work and start a new one, as they form the initial queue for the next epoch. Let $\sigma_1(z)$ be the p.g.f. of the next epoch's queue. When we have the identity $\sigma_0(z) = \sigma_1(z)$, the epoch process is stationary and we'll see that the unconditional average number of blocks in simultaneous PoW, whatever the epoch, is exactly $\frac{\partial_z S^*(1)}{S^*(1)}$.

The probability that the previous epoch ends in the interval $[t, t+dt]$ and that the next epoch begins with n blocks is $\frac{n+1}{D}S_{n+1}(t)dt$. Taking the contributions of all the time intervals, we obtain the p.g.f :

$$\sigma_1(z) = \int_0^\infty \frac{1}{D}\partial_z S(z,t)dt = \frac{1}{D}\partial_z S^*(z).$$

If $\sigma_0(z)$ is the stationary state then we should have $\sigma_1(z) = \sigma_0(z)$ and simplifying by the factor $(z-1)$ the simple identity:

$$\sigma_0(z) = \lambda S^*(z).$$

With the identity $\sigma_0(1) = 1$ we get $S^*(1) = \frac{1}{\lambda}$ which comes with no surprise since we should have $\int_0^\infty S(1,t)dt = \frac{1}{\lambda}$: the average time between two mining should be equal to the average time between two arrivals in stationary situation.

Since block creation follows a Poisson process and epochs are renewal periods, the cumulative average number of blocks waiting during an epoch is $\frac{\partial_z S^*(1)}{S^*(1)}$, i.e. λD, since $\partial_z S^*(1) = D\sigma_1(1) = D$. Therefore, thanks to Little's formula, the average waiting time is D. This is an alternative proof of the theorem 2.

Theorem 3: The unconditional second moment of the number of blocks in simultaneous PoW when the difficulty is constant and equal to D is $(\lambda D)^2 + \lambda D$

Proof: We can go further within this framework by finding the second moment of the number of blocks in PoW, which is $\frac{\partial_z^2 S^*(1) + \partial_z S^*(1)}{S^*(1)}$. From the identity $\sigma_0(z) = \lambda S^*(z)$ which is equivalent to the identity $\frac{1}{D}\partial_z S^*(z) = \lambda S^*(z)$, we naturally have $\frac{1}{D}\partial_z^2 S^*(1) = \lambda \partial_z S^*(1) = \lambda D$, thus the second moment of the number of blocks in PoW is $\frac{\partial_z^2 S^*(1) + \partial_z S^*(1)}{S^*(1)} = (\lambda D)^2 + \lambda D$. ■

B. Proof of theorem 2 with Variable difficulty

Proof: For the purposes of theoretical analysis, we assume that after each epoch, the difficulty of the next extraction is given by a random process (it could be predetermined) without memory. The sequence of epochs forms a Markov chain. We denote $S^{\langle D\rangle}(z,t)$ the p.g.f. of the queue length at time t obtained with difficulty D. Calling $\sigma_0(z)$ the p.g.f. of the initial queue, we obtain the new equation :

$$-\sigma_0(z) = \lambda(z-1)S^{\langle D\rangle *}(z) - \frac{z}{D}\partial_z S^{\langle D\rangle *}(z)$$

with $S^{\langle D\rangle *}(z) = \int_0^\infty S^{\langle D\rangle}(z,t)dt$.

In fact, this is the same identity as in the previous section, except that we keep an exponent $\langle D\rangle$ to remind us that the transition is made in difficulty D. In this case, we denote $\sigma_1^{\langle D\rangle}(z)$ the p.g.f of the

initial queue of the next epoch and we have the identity $\sigma_1^{\langle D\rangle}(z) = \frac{1}{D}\partial_z S^{\langle D\rangle *}(z)$. But even if the stationary state is $\sigma_0(z)$, there's no reason why $\sigma_1^{\langle D\rangle}(z) = \sigma_0(z)$. In fact, we have the identity $\sigma_0(z) = E_D[\sigma_1^{\langle D\rangle}(z)]$ by averaging over all difficulties (see figure 4).

We therefore get in passing the identity

$$\sigma_0(z) = \lambda E_D[S^{\langle D\rangle *}(z)]$$

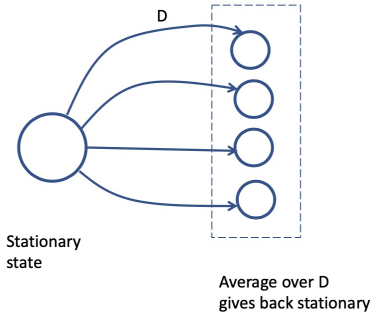

Fig. 4: Transition with variable D

The average number of waiting blocks per time unit is $\frac{\partial_z E_D[S^{\langle D\rangle *}(1)]}{E_D[S^{\langle D\rangle *}(1)]}$ which is $\lambda E[D]$. Indeed $E_D[S^{\langle D\rangle *}(1)] = \frac{1}{\lambda}$ because (i) $1 = \lambda E_D[S^{\langle D\rangle *}(1)]$ and (ii) $E_D[\partial_z S^{\langle D\rangle *}(1)] = E_D[D\sigma^{\langle D\rangle}(1)]$ and $\sigma_1^{\langle D\rangle}(1) = 1$. ∎

Theorem 4: The unconditional second moment of the number of blocks in simultaneous PoW is

$$\left(E[D^2] + E_D[DS^{\langle D\rangle *}(1)]\right)\lambda^2.$$

Proof: The second moment is $\frac{E_D[\partial_z^2 S^{\langle D\rangle *}(1)] + E_D[\partial_z S^{\langle D\rangle *}(1)]}{E_D[S^{\langle D\rangle *}(1)]}$ which is equal to $\lambda E_D[\partial_z^2 S^{\langle D\rangle *}(1)] + \lambda E[D]$, since $E_D[S^{\langle D\rangle *}(1)] = \frac{1}{\lambda}$ and $E_D[\partial_z S^{\langle D\rangle *}(1)] = D$.

We are specifically looking at $E_D[\partial_z^2 S^{\langle D\rangle *}(1)]$. If we average over D the identity $-\sigma_0(z) + \frac{z}{D}\partial_z S^{\langle D\rangle *}(z) = \lambda(z-1)S^{\langle D\rangle *}(z)$ we get

$$\sigma_0(z) = \lambda E_D[S^{\langle D\rangle *}(z)],$$

since $\frac{1}{D}\partial_z S^{\langle D\rangle *}(z) = \sigma_1^{\langle D\rangle}(z)$ and $E_D[\sigma_1^{\langle D\rangle}(z)] = \sigma_0(z)$. From this identity at $z = 1$ we get $E_D[S^{\langle D\rangle *}(1)] = \frac{1}{\lambda}$, something we already know.

If we let $z = 1$ in the identity $-\sigma_0(z) + \frac{z}{D}\partial_z S^{\langle D\rangle *}(z) = \lambda(z-1)S^{\langle D\rangle *}(z)$ and let $z = 1$ we get $-1 + \frac{1}{D}\partial_z S^{\langle D\rangle *}(1) = 0$, thus

$$\partial_z S^{\langle D\rangle *}(1) = D.$$

If we derive the identity $\sigma_0(z) = \lambda E_D[S^{\langle D\rangle *}(z)]$ and make $z = 1$ we get $\partial_z \sigma_0 = \lambda E_D[\partial_z S^{\langle D\rangle *}(1)] = \lambda E[D]$.

If we derive the identity $-\sigma_0(z) + \frac{z}{D}\partial_z S^{\langle D\rangle *}(z) = \lambda(z-1)S^{\langle D\rangle *}(z)$ with respect to the variable z and make $z = 1$ we obtain

$$-\partial_z \sigma_0(1) + \frac{1}{D}\partial_z^2 S^{\langle D\rangle *}(1) + \frac{1}{D}\partial_z S^{\langle D\rangle *}(1) = \lambda S^{\langle D\rangle *}(1).$$

Thus the previously obtained identities lead to

$$\partial_z^2 S^{\langle D\rangle *}(1) = \lambda D S^{\langle D\rangle *}(1) - D + \lambda D E[D]$$

and finally

$$E_D[\partial_z^2 S^{\langle D\rangle *}(1)] = \left(E_D[DS^{\langle D\rangle *}(1)] + E[D]^2\right)\lambda - E[D].$$

∎

Unfortunately, we can't conclude here as we don't have an expression for $E_D[DS^{\langle D\rangle *}(1)]$ in the general case.

C. Block waiting time distribution

The distribution of miners' waiting time is more difficult to analyze, as the waiting time per block may straddle several epochs. So we'll just give an overview of how it can be analyzed theoretically.

Consider a queue of n blocks at time t. Let T be the time already spent in the queue by an arbitrary miner. Let $T_1, \ldots, T_n$ be the respective waiting times of the n blocks, including their waiting times in previous epochs. Let $P_n(t, T_1, \ldots, T_n)$ be the probability that the current epoch started t seconds ago and that the respective waiting times (including previous epochs) of the blocks are n-triple $(T_1, \ldots, T_n)$. Let ω be a complex number, and we define

$$P_n(\omega, t) = \frac{1}{n}\sum_{T_1,\ldots,T_n} P_n(t, T_1, \ldots, T_n)\left(e^{\omega T_1} + \cdots + e^{\omega T_n}\right).$$

We notice that $P_n(0, t) = S_n(t)$. During the evolution from time t to time $t + dt$, the waiting time increases of dt (thus the block $e^{\omega T_1} + \cdots e^{\omega T_n}$ is multiplied by $e^{\omega dt}$). With probability $\frac{n}{D}dt$ the epoch terminates, with probability λdt a new block is added with a waiting time $T_{n+1} = 0$. It consists into replacing the Laplace function $e^{\omega T_1} + \cdots + e^{\omega T_n}$ by the new Laplace expression $e^{\omega T_1} + \cdots + e^{\omega T_n} + 1$. Said differently, the block $\frac{e^{\omega T_1}+\cdots+e^{\omega T_n}}{n}$ by the block $\frac{n}{n+1}\frac{e^{\omega T_1}+\cdots+e^{\omega T_n}}{n} + \frac{1}{n+1}\frac{e^{0 T_1}+\cdots+e^{0 T_n}}{n}$.

This naturally leads to the following time evolution, since $P_n(0, t) = S_n(t)$:

$$\begin{aligned} P_n(\omega, t+dt) &= e^{\omega dt}\left(1 - \lambda dt - \frac{n}{D}dt\right)P_n(\omega, t) \\ &\quad + \lambda dt\left(P_{n-1}(\omega, t)\frac{n-1}{n} + \frac{1}{n}S_{n-1}(t)\right) \end{aligned}$$

The last term indicates that when a new block arrives in a queue of length $n-1$ its starts with a waiting time $T = 0$, the other blocks share a fraction $\frac{n-1}{n}$ of the previous distribution of the waiting times.

Thus defining the p.g.f: $P(z, \omega, t) = \sum_n P_n(\omega, t) z^n$ and letting $dt \to 0$:

$$\begin{aligned} \partial_t P(z, \omega, t) &= (\omega + \lambda(z-1))P(z, \omega, t) + \lambda \mathcal{I} S(z, t) \\ &\quad - \lambda \mathcal{I} P(z, \omega, t) - \frac{z}{D}\partial_z P(z, \omega, t) \end{aligned}$$

where $\mathcal{I}$ is the "primitivation" operator: $\mathcal{I}f(z) = \int_0^z f(x)dx$. Let $P^*(z, \omega) = \int_0^\infty P(z, \omega, t)dt$, and $P(z, \omega, 0) = P_0(z, \omega)$:

$$\begin{aligned} -\pi_0(z, \omega) &= (\omega + \lambda(z-1))P^*(z, \omega) - \lambda \mathcal{I} P^*(z, \omega) \\ &\quad + \lambda \mathcal{I} S^*(z) - \frac{z}{D}\partial_z P^*(z, \omega). \end{aligned}$$

where $\pi_0(z, \omega) = P(z, \omega, 0)$.

What remains to establish is the connection with the next epoch p.g.f. Assume that an epoch ends with n blocks with respective waiting times $T_1, \ldots, T_n$, which correspond to a Laplace function $e^{\omega T_1} + \cdots + e^{\omega T_n}$. When the epoch ends, one of the waiting blocks selected at random will be mined and the other blocks will remain in the queue. All possibilities added up, the mean of the new Laplace function is after normalization is $\frac{1}{n}(e^{\omega T_1} + \cdot + e^{\omega T_n})$. In fact, we can keep only the mean value, as the subsequent evolution of the Markov chain that makes up the sequence of epochs depends only on the number of remaining blocks $n-1$ and not on the individual waiting times. In other words, the average Laplace function is unchanged

when a block leaves the queue. Therefore the initial p.g.f. of the next epoch satisfies

$$\pi_1(z,\omega) = \int_0^\infty \frac{1}{D}\partial_z P(z,\omega,t)dt = \frac{1}{D}\partial_z P^*(z,\omega).$$

In stationary situation we would have to average on all D introducing $P^{\langle D\rangle}(z,\omega,t)$ and $P^{\langle D\rangle *}(z,\omega)$, the epoch starting on the stationary initial distribution $\pi_0(z,\omega)$ and get $\pi_1(z,\omega) = \pi_0(z,\omega)$.

V. Practical simulations

We have simulated block mining, first with a fixed proof of work, and second with a variable proof of work. We assume that all devices have same computing power, the analysis with different computing powers will be done in another work. We place ourselves under the conditions of the proof where the block creation process is Poisson with a rate $\lambda = 0.5$, the block deposit time after PoW is zero (*i.e.* takes a negligible time), PoW is a continuous process that stops in the interval $[t, t+dt]$ with probability $\frac{dt}{D}$ where D is the current difficulty. We ignore the fine-grained interaction with the green election of the leader. We simulated a block mining process, in a short time interval of 200 seconds, and tracked the evolution of the number of blocks which are simultaneously in PoW during this time interval. The model is rather simple, the simulations are done with the formal language Maple.

Figure 5 shows the histogram of the number of blocks in simultaneous PoW, when difficulty D is set to 5. Figure 6 shows the same thing, but with difficulty D variable over values $(2, 4, 8, 16)$ with respective probabilities $(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$, so $E[D] = 5$. In both figures, we add, below the time axis and in negative, the difficulty values as established at each epoch change.

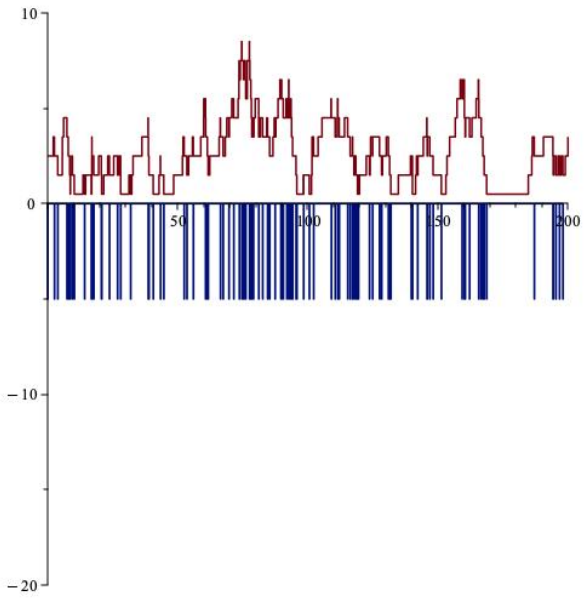

Fig. 5: number of blocks in PoW versus time, fixed D in brown, below in negative and in blue the fixed difficulty D determined at each epoch starting point

Although this may not be obvious over a short period, both graphs should show the same average value over a very long period. To be convinced that these quantities are indeed the same in both figures, we simulate the long-term average for $t = 10,000$ in figure 7 and compare it with the theoretical value $\lambda E[D] = 2.5$.

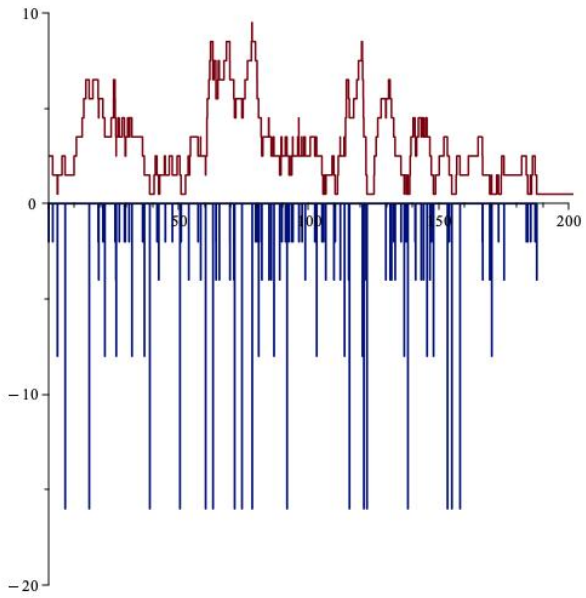

Fig. 6: number of blocks in PoW versus time, fixed D in brown, below in negative and in blue the fractal difficulty D determined at each epoch starting time

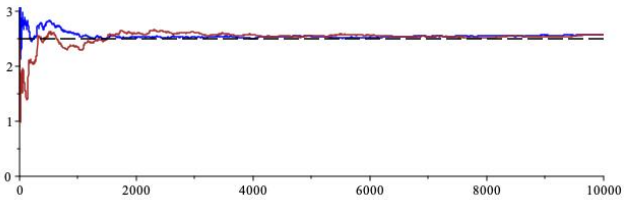

Fig. 7: number of blocks in PoW versus time, for fixed D (blue) and fractal D (brown), dashed the theoretical value $\lambda E[D]$

VI. Conclusion

We have shown that merging green leader election and fractal difficulty PoW is a powerful tool for reducing the energy footprint of blockchains for IoT when the computing capacity of devices is too diverse and makes the energy cost of PoW unsustainable.

The surprising result of this study is that the average performance of a PoW with fractal difficulty seems to depend only on the average difficulty. The surprise comes from the fact that the average performance of a queuing system generally depends on the *second* moment of service times, so that a small variation in the service time statistic has a greater impact on the queuing system. But this is not the case with variable difficulty PoW.

The theoretical analysis presented in this paper is limited to devices with same computing power. The case with different power classes will be the subject of a further work which will show that fractal difficulties can indeed reduce the computing burden on low powered class of devices.

References

[1] Elie Bouri, Naji Jalkh, Peter Molnár and David Roubaud, *Bitcoin for energy commodities before and after the December 2013 crash: diversifier, hedge or safe haven?*, Applied Economics, 49(50):5063-5073, 2017.

[2] Debojyoti Das and Anupam Dutta, *Bitcoin's energy consumption: Is it the Achilles heel to miner's revenue?*, Economics Letters, Available online July 2019, In Press.

[3] Cynthia Dwork and Moni Naor, *Pricing via processing or combatting junkmail*, in proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO), pp. 139–147, London, UK, 1993.

[4] P.J. Grabner and H. Prodinger, *Sorting algorithms for broadcast communications: Mathematical analysis*, Theoretical computer science, 289(1), pp. 51-67, 2002.

[5] Adam S. Hayes, *Cryptocurrency value formation: An empirical study leading to a cost of production model for valuing bitcoin*, Telematics and Informatics, 34(7), pp. 1308-1321, 2017.

[6] Jacquet, Philippe, and Bernard Mans. "Green mining: Toward a less energetic impact of cryptocurrencies." IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2019.

[7] Qiang Ji, Elie Bouri, David Roubaud, and Ladislav Kristoufek, *Information interdependence among energy, cryptocurrency and major commodity markets*, Energy Economics, 81, pp. 1042-1055, 2019.

[8] Cichon, J., Kapelko, R., & Markiewicz, D. (2016). *On Leader Green Election*. In proceedings of the 27th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA), Kraków, Poland, July 4–8, 2016, arXiv preprint arXiv:1605.00137.

[9] K.J. Dwyer and D. Malone, *Bitcoin mining and its energy footprint*. In proceedings of the 25th IET Irish Signals Syst. Conf. (ISSC 14), Jun. 2014, pp. 280-285.

[10] Garay, Juan, Aggelos Kiayias, and Nikos Leonardos. "The bitcoin backbone protocol with chains of variable difficulty." Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part I 37. Springer International Publishing, 2017.