A Random Network Model for the Analysis of Blockchain Designs with Communication Delay

Carlos Pinzón, Camilo Rocha, and Jorge Finke

Pontificia Universidad Javeriana, Cali

Abstract This paper proposes a random network model for blockchains, a distributed hierarchical data structure of blocks that has found several applications in various industries. The model is parametric on two probability distribution functions governing block production and communication delay, which are key to capture the complexity of the mechanism used to synchronize the many distributed local copies of a blockchain. The proposed model is equipped with simulation algorithms for both bounded and unbounded number of distributed copies of the blockchain. They are used to study *fast* blockchain systems, i.e., blockchains in which the average time of block production can match the average time of message broadcasting used for blockchain synchronization. In particular, the model and the algorithms are useful to understand *efficiency* criteria associated with fast blockchains for identifying, e.g., when increasing the block production will have negative impact on the stability of the distributed data structure given the network's broadcast delay.

1 Introduction

A blockchain is a distributed hierarchical data structure of blocks that cannot be altered retroactively without alteration of all subsequent blocks, which requires consensus of the network majority. It was invented to serve as the public transaction ledger of the cryptocurrency Bitcoin in 2008 [13], in which the need for a trusted third party is avoided: instead, this digital currency is based on the concept of 'proof of work' allowing users to execute payments by digitally signing their transactions using hashes through a distributed time-stamping service [15]. Because of its resistance to modifications, decentralized consensus, and proved robustness for supporting cryptocurrency transactions, this technology is seen to have great potential for new uses in other domains, including financial services [7, 17], distributed data models [3], markets [16], government systems [9, 14], healthcare [8, 1, 11], IoT [10], and video games [12]. As the blockchain technology matures, it is expected to change economics, business, and society [2] in the years to come.

Technically, a blockchain is a distributed append-only data structure comprising a linear collection of blocks, shared among several *workers* (or, *miners*; i.e., computational nodes responsible for working on the blockchain with the goal to extend it further with new blocks). Since the blockchain is decentralized, each worker possesses a local copy of the blockchain. This means, e.g., that workers

can be working at the same time on unsynchronized local copies of the blockchain structure. In the typical peer-to-peer network implementation of blockchain technology, workers adhere to a consensus protocol for inter-node communication and validation of new blocks: to work on top of the largest blockchain and, in case of ties, on the one whose last produced block was seen earliest. This protocol guarantees an effective synchronization mechanism as long as the task of producing new blocks is hard to achieve, which is known in the literature as 'proof of work' blockchains, in comparison to the time it takes for inter-node communication. The idea is that, if several workers extend different versions of the blockchain, the consensus mechanism allows the network to eventually select only one of them, while the other blocks are discarded (including the data they carry) when local copies are synchronized. This process carries on upon the creation of new blocks.

The scenario of discarding blocks massively, which can be seen as an efficiency issue in a blockchain implementation, is rarely present in "slow" block-producing blockchains. This is because the time it takes to produce a new block in such implementations is long enough for workers to synchronize their local copy of the blockchain. This prevents, up to some extent, workers from wasting resources and time in producing blocks that will likely be discharged during a future synchronization stage. This is the case, e.g., of the Bitcoin network in which it takes, in average, 10 minutes to mine a block and 12.6 seconds to communicate [6]; it was estimated that the theoretical fork-rate of its blockchain was approximately 1.78% in 2013 [6]. However, as the blockchain technology finds new uses, it is being argued that block production needs to be faster in order to have more versatile and attractive applications [4, 5]. This means that precisely understanding how block production speed-ups can negatively impact blockchains in terms of the amount of blocks discharged due to race conditions among the workers is of great practical importance for designing efficient blockchains in the near future.

This paper presents a random network model for blockchains. It is parametric on the number of workers under consideration (possibly infinite), a probability distribution describing the time for producing new blocks, and a probability distribution describing the communication delay between any pair of random workers in the network. The model is equipped with probabilistic algorithms that can be used to mathematically simulate and analyze blockchains having a fixed or unbounded number of workers producing blocks at different rates over a network with communication delays. In this paper, the model and the algorithms are used as means to study the continuous process of block production in *fast* blockchains: highly distributed networks of workers rapidly producing blocks and in which inter-node communication delays can be crucial for efficient block production. As explained above, one of the main consequences of having faster block production is that blocks tend to be discharged at a higher rate, yielding a speed-efficiency tradeoff in fast blockchains. In this work, experiments are presented to understand how such a tradeoff can be analyzed for many scenarios to showcase the proposed approach. In the broader picture of the years to come in which the use of fast blockchain systems is likely to spread, the

model, algorithms, and approach contributed in this work can be seen as useful mathematical tools for specifying, simulating, and analyzing such designs.

This paper is organized as follows. Section 2 summarizes basic notions of proof-of-work blockchains. Sections 3 and 4 introduce the proposed network model and simulation algorithms. Section 5 presents experimental results in the analysis of fast blockchains. Section 6 summarizes related work and concludes the paper.

2   An Overview of Proof-of-work Blockchains

This section presents an overview of proof-of-work distributed blockchain systems, including basic definitions and an example.

A *blockchain* is a distributed hierarchical data structure of blocks that cannot be altered retroactively without alteration of all subsequent blocks, which requires consensus of the network majority. The nodes in the network, called *workers*, use their computational power to generate *blocks* with the goal of extending the blockchain. The adjective 'proof-of-work' comes from the fact that producing a single block for the blockchain tends to be a hard computational task for the workers (i.e., one requiring extensive computation, e.g., a partial hash inversion).

Definition 1. *A* block *is a digital document containing: (i) a digital signature stating the worker who produced it; (ii) an easy to verify proof-of-work witness in the form of a nonce; and (iii) a hash pointer to the previous block in the sequence (except for the first block, called the* origin*, that has no previous block and is unique).*

A technical definition of blockchain as a data structure has been proposed by different authors (see, e.g., [18]), although most of them coincide on it being an immutable, transparent, and decentralized data structure shared by all the workers in the network. For the purpose of this paper, it is important to clearly distinguish between the *local* copy of the blockchain each worker has and the abstract *global* blockchain shared by all workers. It is the latter one that holds the complete history of the blockchain.

Definition 2. *The* local blockchain *of a worker w is a non-empty sequence of blocks stored in the local memory of w. The* global blockchain *(or,* blockchain*) is the minimal rooted tree containing all workers' local blockchains as branches.*

Note that by the assumption that the root node is unique (Definition 1), the (global) blockchain is well-defined no matter how many workers are part of the network and is non-empty if there is at least one worker. Definition 2 allows for local blockchains of workers to be either synchronized or unsynchronized, the latter being more common in systems with long propagation times or in the presence anomalous situations (e.g., if there is an attacker intentionally trying to hold a fork). This is a good reason why the global blockchain cannot simply be

defined as a unique sequence of blocks, but rather as a distributed data structure to which workers can be assumed to be partly synchronized.

Figure 1 presents an example of a blockchain with five workers, where blocks are represented by natural numbers. On the left, the local blockchains are depicted as linked lists; on the right, the corresponding global blockchain is depicted as a rooted tree. Some of the blocks in the rooted tree representation in Figure 1 are labeled with the identifier of a worker: it indicates the position of each worker in the global blockchain. For modeling purposes, the rooted tree representation of a blockchain is preferred. This is because, on the one hand, it can reduce the amount of memory needed to store it and, on the other hand, it visually simplifies the structure analysis of the data structure. More specifically, storing a global blockchain with m workers containing n unique blocks as a collection of lists requires $O(mn)$ memory in the worst case (i.e., with perfect synchronization). In contrast, the rooted tree representation of the same blockchain with m workers and n unique blocks requires $O(n)$ memory for the rooted tree (e.g., using parent pointers) and an $O(m)$ map for assigning each worker its position in the tree, totaling $O(n + m)$ memory.

$$w_0 : 0 \leftarrow 1 \leftarrow 5 \qquad 0 \leftarrow 1 \leftarrow 5^{w_0}$$
$$w_1 : 0 \leftarrow 2 \leftarrow 3 \leftarrow 6 \qquad 2 \leftarrow 3^{w_3} \leftarrow 6^{w_1, w_4}$$
$$w_2 : 0 \leftarrow 2 \leftarrow 4 \qquad 4^{w_2}$$
$$w_3 : 0 \leftarrow 2 \leftarrow 3$$
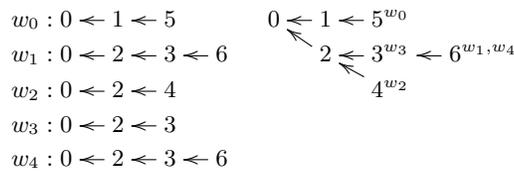$$w_4 : 0 \leftarrow 2 \leftarrow 3 \leftarrow 6$$

Figure 1: A blockchain network of five workers with their local blockchains (left) and the corresponding global blockchain (right); blocks are represented by natural numbers. Workers w_0, w_2, and w_3 are not yet synchronized with the longest sequence of blocks.

A blockchain tends to achieve synchronization among the workers, mainly, because of three reasons. First, workers follow a *standard protocol* in which they are constantly trying to produce new blocks and broadcasting their achievements to the entire network. In the case of cryptocurrency networks, for instance, this behavior is motivated by paying cryptocurrency rewards. Second, workers can easily verify (i.e., with a fast algorithm) the authenticity of any block. If a malicious worker changes the information in one block, it is forced to repeat the extensive proof-of-work process for that block and all its subsequent blocks in the blockchain in order for its malicious modification to be made part of the global blockchain. Since this requires to spend enough electricity, time, and/or machine rental, such a situation is hardly ever expected to happen. Third, the standard protocol forces any malicious worker to confront the computational power of the whole network, assumed to have mostly honest nodes, into a race that is very hard to win [15].

Algorithm 1 presents a definition of the above-mentioned standard protocol assumed to be followed for each worker in a blockchain system. It can be summarized as follows. When a worker produces a new block, it appends a leave to the block it is standing on, moves to it, and notifies the network about its current position and new distance to the root. When a worker receives a notification, it compares its distance to the root with the incoming position and switches to it whenever the one received is larger. To illustrate the use of the standard protocol with a simple example, consider the blockchains depicted in figures 1 and 2, respectively. In the former, either w_1 or w_4 produced block 6, but the other workers are not yet aware of its existence. In the latter, most of the workers are synchronized with the longest branch, which is typical of a slow blockchain system, resulting in a tree with few and short branches.

Algorithm 1: Standard protocol for each worker w_i in a blockchain.

---

1 $B_i \leftarrow [\text{origin}]$
2 **do forever**
3   **do in parallel, stop on first to occur**
4   Task 1: $b \leftarrow$ produce a subsequent block for B_i
5   Task 2: $B' \leftarrow$ notification from another worker
6   **end**
7   **if** Task 1 was completed **then**
8   append b to B_i
9   notify workers in the network about B_i
10   **else if** B' is longer than B_i **then**
11       $B_i \leftarrow B'$
12   **endif**

$$0 \prec 1 \prec 2 \prec 4 \prec 5^{w_7} \prec 6^{w_0,\ldots,w_6}$$
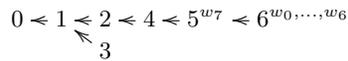$$\nwarrow 3$$

Figure 2: Example of a typical slow system with few and short branches.

Some final remarks on inter-node communication, important for blockchain implementations enforcing the standard protocol, are due. Note that message communication in the standard protocol is required to include enough information about the position of a worker to be located in the tree. The degree of this information depends, generally, on the system's design itself. On the one hand, sending the complete sequence from root to end as part of such a message is a precise way of doing it, but also an expensive one in terms of bandwidth, computation, and time. On the other hand, sending only the last block as part of the message is modest on resources, but it could represent a communication

conundrum whenever the worker being notified about a new block x is not yet aware of x's parent block. This does not happen very often in slow blockchains, but can be frequent in fast systems –such as the ones analyzed in a later section of this manuscript. The common solution, in this case, is to use subsequent messages to query about the previous blocks of x, as needed, thus extending the average duration of inter-working communication.

3   A Random Network Model for Blockchains

The network model proposed in this paper generates a rooted tree representing a global blockchain from a collection of linked lists representing local blockchains (see Definition 2). It consists of three main mechanisms, namely, growth, attachment, and broadcast. By growth it is meant that the number of blocks in the network increases by one at each step. Attachment refers to the fact that new blocks connect to an existing block, while broadcast refers to the fact that the new connected block is announced to the network. In mathematical terms, the network model is parametric in a natural number m specifying the number of workers, and two probability distributions α and β governing the growth, attachment, and broadcast mechanisms of a blockchain. Internally, the growth mechanism creates a new block to be assigned at random among the m workers by taking a sample from α (the time it took to produce such a block) and broadcasts a synchronization message to the entire network, whose reception time is sampled from β (the time it takes the other workers in the network to update their local blockchains with the new block).

A network at a given discrete step n is represented as a rooted tree $T_n = (V_n, E_n)$, with nodes $V_n \subseteq \mathbb{N}$ and edges $E_n \subseteq V_n \times V_n$, and a map $w_n : \{0, 1, \ldots, m-1\} \to V_n$. A node $u \in V_n$ represents a block u in the network and an edge $(u, v) \in E_n$ represents a directed edge from block u to its *parent* block v. The assignment $w_n(w)$ denotes the position (i.e., the last block in the local blockchain) of worker w in T_n.

Definition 3. *Let α and β be positive and non-negative probability distributions respectively. The algorithm used in the network model starts with $V_0 = \{b_0\}$, $E_0 = \{\}$ and $w_0(w) = b_0$ for all workers w, being $b_0 = 0$ the root block, and at each step $n > 0$, it evolves as follows:*

Growth. *A new block b_n (or, simply, n) is created with production time α_n sampled from α. That is, $V_n = V_{n-1} \cup \{n\}$.*

Attachment. *Uniformly at random, a worker $w \in \{0, 1, \ldots, m-1\}$ is chosen for the new block to extend its local blockchain. A new edge appears so that $E_n = E_{n-1} \cup \{(w_{n-1}(w), n)\}$, and w_{n-1} is updated to form w_n with the new assignment $w \mapsto n$, that is, $w_n(w) = n$ and $w_n(z) = w_{n-1}(z)$ for any $z \neq w$.*

Broadcast. *The worker w broadcasts the extension of its local blockchain with the new block n to all other workers z with times $\beta_{n,z}$ sampled from β.*

The rooted tree generated by the model in Definition 3 begins with the block 0 (the root) and adds new blocks $n = 1, 2, \ldots$ to some of the workers. At each

step $n > 0$, a worker w is selected at random and its local blockchain, $0 \leftarrow \cdots \leftarrow$ $w_{n-1}(w)$, is extended to $0 \leftarrow \cdots \leftarrow w_{n-1}(w) \leftarrow n = w_n(w)$. This results in a concurrent random global behavior, inherent to distributed blockchain systems, not only because the workers are chosen randomly due to the proof-of-work scheme, but also because the communication delay brings some workers out of sync. It is important to note that the steps $n = 0, 1, 2, \ldots$ in the model are logical time steps, not to be confused with the sort of time units sampled from the variables α and β. More precisely, the model does not mention explicitly the time advancement but assumes that workers are synchronized at the corresponding point in the logical future. For instance, if w sends a synchronization message of a newly created block n to another worker z, at the end of logical step n and taking $\beta_{n,z}$ time, then the model assumes that the message will be received by z during the logical step $n' \geq n$ that satisfies $\sum_{i=n+1}^{n'} \alpha_i \leq \beta_{n,z} < \sum_{i=n+1}^{n'+1} \alpha_i$.

Other reasonable assumptions are implicitly made in the proposed model, namely: (i) the computational power of all workers is similar; and (ii) any broadcasting message includes enough information about the new block and its previous blocks, that no re-transmission is required to fill block gaps or, equivalently, that these re-transmission times are included in time sampled from β. Assumption (i) justifies why the worker producing the new block is chosen uniformly at random. Thus, instead of simulating the proof-of-work of the workers in order to know who will produce the next block and at what time, it is enough to select a worker uniformly and to take a sample time from α. Assumption (ii) helps in keeping the model description simple because, otherwise, it would be mandatory to explicitly define how to proceed when a worker is severely out of date, that is, if it requires several messages to synchronize.

In practice, the distribution α that governs the time it takes for the whole network as a single entity to produce a block is exponential with mean $\overline{\alpha}$. This is due to the proof-of-work scheme. Since this scheme is based on finding a nonce that makes a hashing function fall into a specific set of targets, the process of producing a block is statistically equivalent to waiting for a success in a sequence of Bernoulli trials. Such waiting time would then correspond –at first– to a discrete geometric distribution. However, it can as well be approximated by a continuous exponential distribution function because the time between trials is very small compared to the average time between successes (usually fractions of micro seconds against several seconds or minutes). On the other hand, the choice of the distribution function β that governs the communication delay, and whose mean is denoted by $\bar{\beta}$, can heavily depend on the system under consideration and its communication details (e.g., hardware, protocol).

4   Algorithmic Analysis of Blockchain Efficiency

This section presents an algorithmic approach to the analysis of blockchain efficiency. The algorithms presented in this section are used to estimate the proportion of valid blocks that are produced, based on the network model introduced in Section 3, both for blockchains with bounded and unbounded number of workers

up to a given number of steps. In general, although presented in this section for the specific purpose of measuring blockchain efficiency, these algorithms can be easily modified to compute other metrics of interest.

Definition 4. *Let $T_n = (V_n, E_n)$ be a blockchain obtained from Definition 3. The* proportion of valid blocks p_n *in* T_n *is defined as the random variable:*

$$p_n = \frac{\max\{\text{dist}(0, u) \mid u \in V_n\}}{|V_n|}.$$

The proportion of valid blocks p *produced for a blockchain (in the limit) is defined as the random variable:*

$$p = \lim_{n \to \infty} p_n.$$

And their expected values are denoted with $\bar{p}_n$ and $\bar{p}$ respectively.

Note that these random variables are particularly useful to determine some important properties of blockchains. For instance, the probability that a newly produced block becomes valid in the long run is exactly $\bar{p}$, the average rate at which the longest branch grows can be approximated by $\bar{p}/\bar{\alpha}$, the rate at which invalid blocks are produced is approximately $(1 - \bar{p})/\bar{\alpha}$, and the expected time for a block receiving one confirmation $\bar{\alpha}/\bar{p}$. Although p_n and p are random for any single simulation, their expected values $\bar{p}_n$ and $\bar{p}$ can be approximated by averaging several Monte Carlo simulations.

The three algorithms presented in this section, which are sequential and single threaded, are designed to compute the value of p_n under the assumption of the standard protocol (Algorithm 1). They can be used for computing $\bar{p}_n$ and, thus, for approximating $\bar{p}$ with large values of n. The first and second algorithms *exactly* compute the value of p_n for a bounded number of workers. The difference is that the first one simulates the three mechanisms present in the network model (i.e., growth, attachment, and broadcast –see Definition 3), while the second one takes a more time-efficient approach for computing p_n. The third one is a fast approximation algorithm for p_n in the context of an *unbounded* number of workers; this algorithm is of special interest for studying the efficiency of large and fast blockchain systems because its time complexity does not depend on the number of workers in the network.

4.1   Network Simulation with a Priority Queue

Algorithm 2 simulates the network model with m workers running concurrently under the standard protocol up to n logical steps. This algorithm uses a list B of m block sequences that reflect the internal blockchains of each worker. The sequences are initially limited to the origin block 0 and can be randomly extended during the simulation. Each iteration of the main loop consists of four stages: (i) waiting for a new block to be produced, (ii) simulating the reception of messages during a period of time, (iii) adding a block to the blockchain of a randomly

selected worker, and (iv) broadcasting the new position of the selected worker in the shared blockchain to the others in the network. The priority queue pq is used to queue messages for future delivery, thus simulating the communication delays. Messages have the form (t', i, B'), where t' represents the message future (physical) arrival time, worker i is the recipient, and the content B' informs about a (non-specified) worker having the sequence of blocks B'. The statements $\alpha()$ and $\beta()$ draw samples from α and β, respectively.

Algorithm 2: Simulation of m workers using a priority queue.

---

1 $t \leftarrow 0$
2 $B \leftarrow [\,[0], [0], ..., [0]\,]$ *(m block sequences, 0 is the origin)*
3 pq $\leftarrow$ empty priority queue
4 **for** $k \leftarrow 1, ..., n-1$ **do**
5 $\quad t \leftarrow t + \alpha()$
6 $\quad$ **for** $(t', i, B') \in$ pq with $t' < t$ **do**   *(receive notifications)*
7 $\quad \quad$ pop (t', i, B') from pq
8 $\quad \quad$ **if** B' is longer than B_i **then** $B_i \leftarrow B'$ **endif**
9 $\quad$ **end**
10 $\quad j \leftarrow$ random_worker()   *(producer)*
11 $\quad$ append a new block (k) to B_j
12 $\quad$ **for** $i \in \{0, ..., m-1\} \setminus \{j\}$ **do**   *(publish notifications)*
13 $\quad \quad$ push $(t + \beta(), i, B_j)$ to pq
14 $\quad$ **end**
15 **end**
16 $s \leftarrow \underset{s \in B}{\arg\max} |s|$ *(longest sequence)*
17 **return** $|s|/n$

The overall complexity of Algorithm 2 depends, as usual, on specific assumptions on its concrete implementation. First, it is assumed that the time complexity to query $\alpha()$ and $\beta()$ is $O(1)$, which is typical in most computer programming languages). However, the time complexity estimates presented next may be higher depending on their specific implementations; e.g., if a histogram is used instead of a continuous function for sampling these variables. If the statement $B_i \leftarrow B'$ is implemented creating a copy in $O(n)$ time and the append statement is $O(1)$, then the overall time complexity of the algorithm is $\Omega(mn^2)$. If $B_i \leftarrow B'$ merely copies the list reference in $O(1)$ and the append statement creates a copy in $O(n)$, the complexity is improved down to $O(mn\log(mn))$, under the assumption of having a standard priority queue with log-time insertion and removal. In either case, the spatial complexity is $O(mn)$.

One key advantage of this algorithm, with respect to the other ones presented next, is that it can be slightly modified to return the blockchain s instead of the proportion p_n. This would enable a richer analysis to be carried out in the form of other metrics different to p. For example, assume I denotes the random variable that describes the quantity of invalid blocks that are created between

consecutive blocks. $E[I]$ can be estimated from $\bar{p}$ because $E[I] \approx (1 - \bar{p})/\bar{\alpha}$. However, building a complete blockchain can be used to estimate not only $E[I]$, but also a complete histogram of I and all the properties it may posses.

4.2 A Faster Simulation Algorithm

Algorithm 3 can be a faster alternative to Algorithm 2. It uses a different encoding for the collection of local blockchains: it stores their length instead of the sequences themselves and suppresses the need for a priority queue.

Algorithm 3: Simulation of m workers using a matrix d

---

1 $t_0, h_0, z_0 \leftarrow 0, 1, 0$
2 $\mathbf{d_0} \leftarrow \langle 0, 0, ..., 0 \rangle$ *(m elements)*
3 for $k \leftarrow 1, ..., n - 1$ **do**
4 $j \leftarrow$ random_worker()
5 $t_k \leftarrow t_{k-1} + \alpha()$
6 $h_k \leftarrow 1 + \max\{h_i \,|\, i < k \,\wedge\, t_i + d_{i,j} < t_k\}$ (Algorithm 4)
7 $z_k \leftarrow \max(z_{k-1}, h_k)$
8 $\mathbf{d_k} \leftarrow \langle \beta(), ..., \beta(), \overbrace{0}^{j\text{'th position}}, \beta(), ..., \beta() \rangle$
9 end
10 return z_{n-1}

The variable t_k represents the (absolute) time at which the block k is created, the variable h_k the length of the local blockchain after being extended with block k, and z_k the cumulative maximum given by $z_k := \max\{h_i \,|\, i \leq k\}$.

The spatial complexity of Algorithm 3 is $O(mn)$ due to the computation of matrix d and its overall time complexity is $O(nm + n^2)$ when Algorithm 4 is not used. This is because there are n iterations, each requiring $O(n)$ and $O(m)$ for computing h_k and $\mathbf{d_k}$, respectively. However, if Algorithm 4 is used for computing h_k, the average overall complexity is reduced. Although the complexity of Algorithm 4 is theoretically $O(k)$ in the worst case, the experimental evaluations suggest an average below $O(\bar{\beta}/\bar{\alpha})$ (constant respect to k). Thus, the average runtime complexity of Algorithm 3 is actually bounded by $O\left(nm + \min\{n^2, n + n\bar{\beta}/\bar{\alpha}\}\right)$, and this corresponds to $O(nm)$, unless the blockchain system is extremely fast ($\bar{\beta} \gg \bar{\alpha}$).

4.3 An Approximation Algorithm for Unbounded Number of Workers

Algorithms 2 and 3 are proposed to compute the value of p_n for a *fixed* number m of workers in the blockchain. Of course, these algorithms can be used to compute p_n for different values of m. However, the time complexity of these

---

Algorithm 4: Fast computation of h_k given t_i, z_i, h_i and $\mathbf{d_i}$ for all $i < k$

---

1 $x, i \leftarrow 1, k-1$
2 **while** $i \geq 0$ and $x < z_i$ **do**
3   **if** $t_i \leq t_k - d_{i,j}$ and $h_i > x$ **then**
4 $x = h_i$
5   **endif**
6 $i \leftarrow i-1$
7 **end**
8 **return** $1 + x$ *(computes $h_k := 1 + \max\{h_i \mid i < k \land t_i + d_{i,j} < t_k\} \cup \{1\}$)*

---

two algorithms heavily depends on the value of m, which presents a practical limitation when faced with the task of analyzing large blockchain systems. This section presents an algorithm for approximating p_n for an unbounded number of workers and formal observations that support this claim.

Recall the definition of p_n, from the beginning of this section, used as a measure of efficiency in terms of the proportion of valid blocks in the blockchain $T_n = (V_n, E_n)$ produced up to step n:

$$p_n = \frac{\max\{\text{dist}(0, u) \mid u \in V_n\}}{|V_n|}.$$

This definition assumes a fixed number m of workers. That is p_n, can be better written as $p_{m,n}$ to represent the proportion of valid blocks in the blockchain T_n *with* m workers. For the analysis of large blockchains, the challenge is then in finding an efficient way to estimate $p_{m,n}$ as m and n grow. To be more precise, the challenge is in finding an efficient algorithm for approximating the random variables p_n^* and p^* defined as follows:

$$p_n^* = \lim_{m \to \infty} p_{m,n} \qquad \text{and} \qquad p^* = \lim_{n \to \infty} p_n^*.$$

The approach presented next to tackle the given challenge is to modify Algorithm 3 by removing the matrix d. The idea is to replace the need for the $d_{i,j}$ values by an approximation based on the random variable β in order to compute h_k in each iteration of the main loop. The first observation is that the first row can be assumed to be 0 wherever it appears because $d_{0,j} = 0$ for all j. For the remaining rows, an approximation is introduced by observing that if an element X_m is chosen at random from the matrix d of size $(n-1) \times m$ (i.e., matrix d without the first row), the cumulative distribution function of X_m is given by

$$P(X_m \leq r) = \begin{cases} 0 & , r < 0 \\ \frac{1}{m} + \frac{m-1}{m} P(\beta() \leq r) & , r \geq 0, \end{cases}$$

where $\beta()$ is a sample from β. This is because the elements X_m of d are either samples from β, whose domain is $\mathbb{R}_{\geq 0}$, or the value 0 with a probability of $1/m$

since there is one zero per row. Therefore, given that the following functional limit converges uniformly (Theorem 1),

$$\lim_{m \to \infty} \left(r \overset{f_m}{\mapsto} P(X_m \leq r) \right) = \left(r \overset{f}{\mapsto} P(\beta() \leq r) \right),$$

each $d_{i,j}$ can be approximated by directly sampling the distribution β and then Algorithm 4 can be used for computing h_k by replacing $d_{i,j}$ with $\beta()$.

Theorem 1. *Let $f_k(r) := P(X_k \leq r)$ and $g(r) := P(\beta() \leq r)$. The functional sequence $\{f_k\}_{k=1}^{\infty}$ converges uniformly to g.*

Proof. Let $\epsilon > 0$. Define $n := \left\lceil \frac{1}{2\epsilon} \right\rceil$ and let k be any integer with $k > n$. Then

$$
\begin{aligned}
\sup |f_k - g| &= \sup \left\{ \left| \frac{1}{k} + \left(\frac{k-1}{k} - 1 \right) P(\beta() \leq r) \right| : r \geq 0 \right\} \\
&\leq \frac{1}{k} + \frac{1}{k} \sup \{ P(\beta() \leq r) : r \geq 0 \} \\
&= \frac{1}{k} + \frac{1}{k} \\
&< \frac{2}{n} \leq \epsilon.
\end{aligned}
$$

$\square$

With the help of the above observations, the need for the bookkeeping matrix is removed, and both variables j and d can be discarded from Algorithm 3 to obtain Algorithm 5. Note that this new algorithm computes p_n^*, an approximation of $\lim_{m \to \infty} p_{m,n}$ in which the matrix entries $d_{i,j}$ are replaced by new samples from β each time they are needed, thus ignoring the arguably negligible hysteresis effects.

Algorithm 5: Approximation for $\lim_{m \to \infty} p_{m,n}$ simulation

---

1 $t_0, h_0, z_0 \leftarrow 0, 0, 0$
2 for $k \leftarrow 1, ..., n-1$ **do**
3 $t_k \leftarrow t_{k-1} + \alpha()$
4 $h_k \leftarrow 1 + \max \{ h_i \mid i < k \wedge t_i + \beta() < t_k \} \cup \{1\}$ (Algorithm 4*)
5 $z_k \leftarrow \max(z_{k-1}, h_k)$
6 end
7 return z_{n-1}
Algorithm 4* stands for Algorithm 4 with $\beta()$ instead of $d_{i,j}$ (approximation)

The complexity of Algorithm 5, if implemented without using Algorithm 4, is $O(n^2)$-time and $O(n)$-space. But if the pruning algorithm is used, the time complexity drops below $O(n + n\bar{\beta}/\bar{\alpha}))$ according to experimentation, which can be considered $O(n)$ as long as the blockchain system is not extremely fast (i.e., when $\bar{\beta} \gg \bar{\alpha}$).

5   Empirical Evaluation of Blockchain Efficiency

This section presents an experimental evaluation of blockchain efficiency in terms of the proportion of valid blocks produced by the workers for the global blockchain. The proposed model in Section 3 is used as the mathematical basis for the evaluation, while the algorithms in Section 4 are used for the experimental evaluation. The main claim made in this section is that the efficiency of a blockchain, under some assumptions later identified in this section, can be expressed as a ratio between $\bar{\alpha}$ and $\bar{\beta}$. This section also presents experimental evidence on why Algorithm 5 –a fast approximation algorithm for computing the proportion of valid blocks in a blockchain with an unbounded number of workers– is an accurate tool for computing $p^* = \lim_{n,m \to \infty} p_{m,n}$.

The speed of a blockchain can be characterized by the relationship between the expected values of α and β.

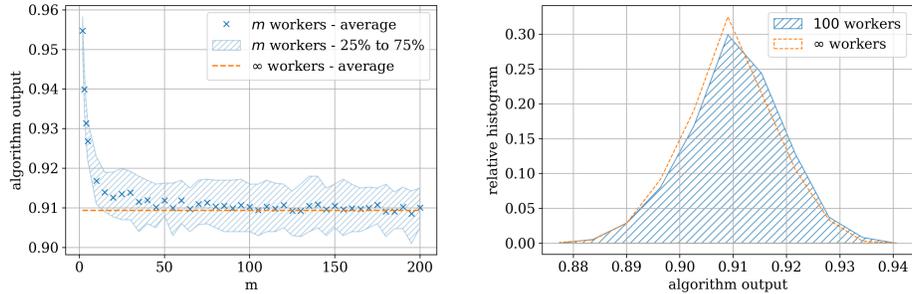**Definition 5.** *Let α and β be the distributions in Definition 3. A blockchain is called:*

- slow *whenever $\bar{\alpha} \gg \bar{\beta}$.*
- chaotic *whenever $\bar{\alpha} \ll \bar{\beta}$.*
- fast *whenever $\bar{\alpha} \approx \bar{\beta}$.*

The classification in Definition 5 captures the intuition about the behavior of a global blockchain in terms of how alike the times for producing a block and for local block synchronization are. The Bitcoin implementation can be classified as a slow blockchain because the time between the creation of two consecutive blocks is much larger than the time it takes the local blockchains to synchronize. In chaotic blockchains, a dwarfing synchronization time means that basically no or very little synchronization is possible, resulting in a blockchain in which rarely any block would be part of "the" valid chain of blocks. A fast blockchain, however, is one in which both the times for producing a block and message broadcasting are similar. The two-fold goal of this section can now be better explained. First, it is to analyze the behavior of $\bar{p}^*$ for any of these three classes of blockchains. On the other, it is to understand how the trade-off between block production speed and broadcasting time affect the efficiency of the data structure by means of a formula.

In favor of readability, the experiments presented in this section identify algorithms 3 and 5 as $A_m(\_)$ and $A_\infty(\_)$, respectively. Furthermore, the claims and experiments presented next assume that the distribution α is exponential, which is the case for proof-of-work systems.

Claim 1 *Unless the system is chaotic, the hysteresis effect of the matrix entries $d_{i,j}$ in Algorithm 3 is negligible. Moreover, $\lim_{m,n \to \infty} A_m(n) = A_\infty(n)$.*

Note that Theorem 1 implies that if the hysteresis effect of the random variables $d_{i,j}$ is ignored, then it would necessarily hold that $\lim_{m,n \to \infty} A_m(n) = A_\infty(n)$. However, it does not prove that the equation holds in general, but only

(a) Evolution of A_m to A_∞ as m grows. At least 100 samples per point.

(b) High similarity between the pdf's of A_{100} and A_∞. At least 1000 samples in total.

Figure 3: Algorithmic simulation of $n = 1000$ blocks with $\overline{\alpha} = 1$, $\overline{\beta} = 0.1$, and β exponential.

if the hysteresis effect is negligible. Experimental evaluation suggests that this is the case indeed, as stated in Claim 1.

Figure 3 summarizes the average output of the A_m algorithm and the region that contains half of these outputs, for several values of m. The output of this algorithm seems to approach that of A_∞, not only for the expected value (Figure 3.(a)), but also in p.d.f. shape (Figure 3.(b)). These experiments were performed with several distribution functions for β with similar results. In particular, the exponential, chi-squared, and gamma (with $k \in \{1, 1.5, 2, 3, 5, 10\}$) probability distribution functions were used, all with different mean values, and the plots were similar to the ones already depicted in Figure 3.

However, as the quotient $\overline{\beta}/\overline{\alpha}$ grows beyond values of 1, the convergence of A_m becomes much slower and the approximation error can be noticeable. For instance, this is the case of a blockchain system in which 10 blocks are produced in average during the transmission of a synchronization message (i.e., a somewhat chaotic system), as depicted in Figure 4. Even after considering 1000 workers, the p.d.f.'s are shifted. The error can be due to the hysteresis effect that is ignored by the A_∞ algorithm, or it might be a consequence of the slow rate of convergence. In any case, the output of these systems is very low, thus making them unstable and useless in practice.

An intuitive conclusion about blockchain efficiency and speed of block production is that slower systems tend to be more efficient than faster ones. That is, faster blockchain systems have a tendency to overproduce blocks that will not be valid.

Claim 2 *If the system is not chaotic, then*

$$\overline{p}^* = \frac{\overline{\alpha}}{\overline{\alpha} + \overline{\beta}}.$$
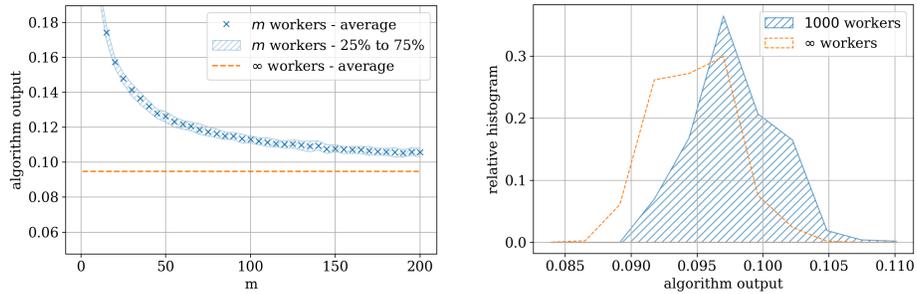
Figure 4: The convergence is slower and the approximation error is larger in chaotic systems: with 1000 workers there is still an average output shift of around 0.005.

Figure 5 presents an experimental evaluation of the proportion of valid blocks produced in a blockchain in terms of the ratio $\frac{\bar{\beta}}{\bar{\alpha}}$. In both plots, the horizontal axis represents how fast blocks are produced in comparison with how slow synchronization is achieved. If the system is slow, then efficiency is high because most of valid produced blocks tend to be valid. If the system is chaotic, however, then efficiency is low because the newly produced blocks are highly likely to become invalid. Finally, note that for non-slow blockchains, say for $10^{-1} \leq \frac{\bar{\beta}}{\bar{\alpha}}$, block production can be good whenever $\bar{\alpha} > \bar{\beta}$. As a matter of fact, this is an important observation since the experiments suggest that even if synchronization of local blockchains takes in average a tenth of the time it takes to produce a block in general, the proportion of produced blocks that become valid is near 90%. In practice, this observation can bridge the gap between the current use of blockchains as slow systems and the need for faster blockchains to cope with the challenges in the years to come.
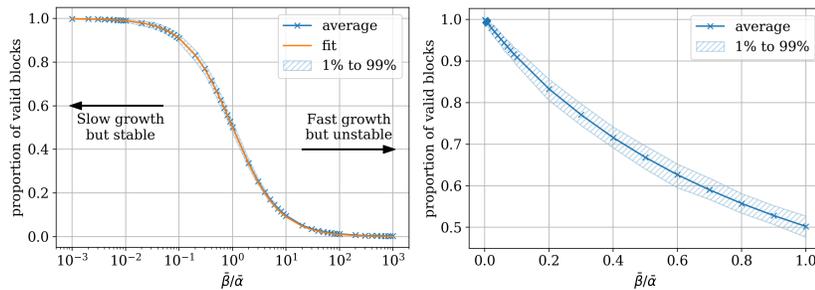

Figure 5: Effect of speed in the proportion of valid blocks.

6   Concluding Remarks

This paper presented a network model for blockchains and has shown how the proposed simulation algorithms can be used to analyze the efficiency (in terms of production of valid blocks) of fast blockchain systems. The model is parametric on: (i) the number of workers (or nodes); and (ii) two probability distributions governing the time it takes to produce a new block and the time it takes the workers to synchronize their local copies of the blockchain. The simulation algorithms are probabilistic in nature and can be used to compute the expected value of several metrics of interest, both for a fixed and unbounded number of workers, via Monte Carlo simulations. It is proven, under some reasonable assumptions, that the fast approximation algorithm for an unbounded number of workers yields accurate estimates in relation to the other two exact (but much slower) algorithms. Claims –supported by extensive experimentation– have been proposed, including a formula to measure the proportion of valid blocks produced in a blockchain in terms of the two probability distributions of the model. The model, algorithms, experiments, and experimentally backed insights contributed by this paper can be seen as useful mathematical tools for specifying, simulating, and analyzing the design of fast blockchain systems in the broader picture of the years to come.

Future work on the analytic analysis of the experimental observations contributed in this work should be pursued; this includes proving the claims. Furthermore, the generalization of the claims to non proof-of-work schemes, i.e. to different probability distribution functions for specifying the time it takes to produce a new block may also be considered. Finally, the study of different forms of attack on blockchain systems can be pursued with the help of the proposed model.

Bibliography

[1] Zainab Alhadhrami, Salma Alghfeli, Mariam Alghfeli, Juhar Ahmed Abedlla, and Khaled Shuaib. Introducing blockchains for healthcare. In *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pages 1–4. IEEE, 2017.

[2] Tomaso Aste, Paolo Tasca, and Tiziana Di Matteo. Blockchain technologies: The foreseeable impact on society and industry. *Computer*, 50(9):18–28, 2017.

[3] Thanh Bui and Tuomas Aura. Application of public ledgers to revocation in distributed access control. In *International Conference on Information and Communications Security*, pages 781–792. Springer, 2018.

[4] Usman W Chohan. The limits to blockchain? Scaling vs. Decentralization. 2019.

[5] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[6] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[7] Ye Guo and Chen Liang. Blockchain application and outlook in the banking industry. *Financial Innovation*, 2(1):24, 2016.

[8] Omar Gutiérrez, Jeffreys J Saavedra, Pedro M Wightman, and Augusto Salazar. Bc-med: Plataforma de registros médicos electrónicos sobre tecnología blockchain. In *2018 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pages 1–6. IEEE, 2018.

[9] Heng Hou. The application of blockchain technology in e-government in china. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–4. IEEE, 2017.

[10] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing iot devices using blockchain platform. In *2017 19th international conference on advanced communication technology (ICACT)*, pages 464–467. IEEE, 2017.

[11] Elena Karafiloski and Anastas Mishev. Blockchain solutions for big data challenges: A literature review. In *IEEE EUROCON 2017-17th International Conference on Smart Technologies*, pages 763–768. IEEE, 2017.

[12] Sundas Munir and Mirza Sanam Iqbal Baig. Challenges and security aspects of blockchain based online online multiplayer games, 2019.

[13] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[14] Svein Ølnes, Jolien Ubacht, and Marijn Janssen. Blockchain in government: Benefits and implications of distributed ledger technology for information sharing, 2017.

[15] Carlos Pinzón and Camilo Rocha. Double-spend attack models with time advantage for bitcoin. *Electronic Notes in Theoretical Computer Science*, 329:79–103, 2016.

[16] Janusz J Sikorski, Joy Haughton, and Markus Kraft. Blockchain technology in the chemical industry: Machine-to-machine electricity market. *Applied Energy*, 195:234–246, 2017.

[17] Alex Tapscott and Don Tapscott. How blockchain is changing finance. *Harvard Business Review*, 1(9):2–5, 2017.

[18] Horst Treiblmaier. Toward more rigorous blockchain research: Recommendations for writing blockchain case studies. *Frontiers in Blockchain*, 2:3, 2019.