End-to-End Latency Analysis and Optimal Block Size of Proof-of-Work Blockchain Applications

Francesc Wilhelmi®, Sergio Barrachina-Muñoz®, and Paolo Dini®

Abstract—Due to the increasing interest in blockchain technology for fostering secure, auditable, decentralized applications, a set of challenges associated with this technology need to be addressed. In this letter, we focus on the delay associated with Proof-of-Work (PoW)-based blockchains, whereby participants validate the new information to be appended to a distributed ledger via consensus to confirm transactions. We propose a novel end-to-end latency model based on batch-service queuing theory that characterizes timers and forks for the first time. Furthermore, we derive an estimation of the optimal block size analytically. Endorsed by analytical and simulation results, we show that the optimal block size approximation is a consistent method that leads to close-to-optimal performance by significantly reducing the overheads associated with blockchain applications.

Index Terms—Blockchain, decentralized applications, proof-of-work, queuing theory.

I. INTRODUCTION

Blockchain has emerged as a groundbreaking approach to enable secure, trustworthy, and resilient decentralized applications. The accountability properties of blockchain favor the interactions between several untrusted partners for building collaborative applications, thus leading to innovative scenarios in e-health, finance, governance, or communications [1]. A prominent example where blockchain is used to boost collaboration among untrusted parties is Federated Learning (FL) by securing data sharing for model training [2]. Blockchain yields remarkable properties like security, decentralization, and integrity, which are essential for the distributed learning operation in FL. However, as discussed in [3], blockchain incurs additional delays and overheads, therefore limiting its efficiency and calling for optimization of parameters such as the block size. Overall, the adoption of blockchain technology raises several concerns, especially for public blockchains, where any interested party is allowed to participate [4], [5]. For instance, Proof-of-Work (PoW)-based blockchains incur high delay and energy costs to secure data.

The optimization of a blockchain is tied to the trilemma involving decentralization (not relying on central control), security (operating as expected in presence of attackers), and scalability (mitigating an increasing number of transactions). In this regard, it is challenging to improve one dimension without compromising the others. This letter contributes to the optimization of blockchains by addressing the minimization of the transaction confirmation latency in PoW-based systems. A similar problem was addressed in [6], which provided an end-to-end latency evaluation for blockchain-enabled FL to optimize the block generation rate (\(\lambda\)). Also in the domain of FL, [7] derived the optimal block size by minimizing the fork probability. In [8], the optimal block generation ratio was derived through a genetic algorithm. Alternatively, [9] studied the communication cost of blockchain and provided a deep reinforcement learning (DRL) mechanism to improve the system utility. DRL was also used in [10] to improve the performance of a blockchain by selecting the best set of block producers, consensus algorithm, block size, and block interval.

Different from the literature, we focus on block size optimization to minimize the transaction confirmation latency of a blockchain. This problem substantially differs from optimizing the block generation rate, which depends on the computational capacity of miners and the difficulty of the consensus problem to be solved. Instead, controlling the block size allows studying the trade-off between mining velocity and communication overhead. For small block sizes, the block propagation and the waiting times decrease, but more overhead is incurred due to block headers. In contrast, increasing the block size contributes to adding extra waiting delays. Hence, optimizing the block size is particularly useful to capture the varying blockchain traffic, thus enhancing throughput and mitigating congestion.

The main contribution of this letter is twofold. First, we propose an end-to-end latency model based on batch-service queuing theory to characterize the delay of a PoW-based blockchain. Unlike the existing literature, our model includes timers and forks, which are important aspects that affect the election of the block size, as shown later in this letter. Second, we derive an estimation of optimal block size analytically, which is evaluated through numerical results and simulations. Our results show that the proposed block size approximation is a consistent method to obtain close-to-optimal performance.

The rest of the letter is structured as follows: Section II gives a brief overview of blockchain technology and PoW. Section III describes the latency model and provides an optimization framework for the block size. Section IV showcases the accuracy of the proposed optimization model through numerical results. Section V concludes the letter.

II. POW-BASED BLOCKCHAIN OPERATION

A blockchain network is composed of a set of nodes (e.g., miners) that communicate through P2P links. Blockchain nodes are responsible for maintaining, updating, and verifying the status of clients’ information (e.g., economic transactions
or infrastructure ownership status). The information stored in a blockchain is organized in cryptographically chained blocks, a principal requirement for ensuring the immutability and inviolability of the system. Each block is connected to its predecessor by adding a reference to a previous block hash. As a result, any change on any block would alter the entire chain, which would not be accepted by the rest of the miners. While many functions exist for encoding blockchain data and chaining blocks, the most popular ones are based on Merkle tree [11]. Following a Merkle tree structure, a block is composed of the nonce (the result of mining), the previous block hash, the Merkle root (a composition of hashes from previous blocks), the timestamp, and a body with transactions.

Transactions are grouped in blocks of variable size \( b = h + nt \), where \( h \) is the header size (constant), \( n \) is the number of transactions included in the block, and \( t \) is the size of a single transaction. To generate a new block, miners execute PoW, whereby a computation-intensive task is completed using capacity \( \lambda \). While PoW requires significant computation and storage capabilities, it provides a high level of security to fully decentralized applications. Fig. 1 illustrates the PoW operation. If two or more miners (e.g., 1 and 3) solve a block simultaneously, different replicas are propagated to the operation. If two or more miners (e.g., 1 and 3) solve a block fully decentralized applications. Fig. 1 illustrates the PoW operation:

![Fig. 1](image)

A. Transaction Confirmation Latency

The transaction confirmation latency of a blockchain can be derived by considering the following delays:

1) **Queuing delay** \( (T_q) \): Clients submit transactions to the closest miner following a Poisson process where the time between transactions is given by an exponential distribution with parameter \( \mu \). Transactions wait in a pool (assumed to be perfectly shared) before being included in a block, i.e., when the block size \( b \) is achieved or when timer \( \tau \) expires. We resort to the queue model of Sect. III-B to characterize the queuing delay.

2) **Block generation delay** \( (T_{bg}) \): Miners run PoW to find the candidate block’s nonce. Mining is modeled through an exponential random variable with parameter \( \lambda \), so the expected mining time \( T_{bg} \) for \( M \) miners is \( (M \lambda)^{-1} \). Note that the block size does not affect the mining time.

3) **Block propagation delay** \( (T_{bp}) \): Miners append mined blocks to their ledger copy. To generate a new block, miners execute PoW simultaneously, different replicas are propagated to the operation. If two or more miners (e.g., 1 and 3) solve a block before the winner’s one is completely propagated. If this occurs, unconfirmed miners may mistakenly add non-winner blocks to their ledger version. Since the time between blocks is a Poisson inter-arrival process, the fork probability is given by \( p_{fork} = 1 - e^{-\lambda(M-1)T_{bp}} \).

B. Queue Model

As done in [12], we consider a finite-length \( M/M^s/1/K \) queue (see Fig. 2) – where \( s \) denotes the number of transactions to be served in a batch – to derive the transaction confirmation latency of a blockchain. For that purpose, we model the queue using a Markov chain where states indicate the number of queued transactions before a departure. We apply the Poisson arrivals see time averages (PASTA) property to obtain the steady-state queue distribution.

The expected queue delay is calculated as

\[
T_q = \frac{\sum_{k=0}^{K} k \pi_k}{\mu(1 - \pi^K)},
\]

where \( \pi^k \) is the queue’s steady-state distribution, which indicates the probability of each state \( k \). To get \( \pi^k \), we first derive the departures distribution as \( \pi^d = \pi^d P \), being \( \pi^d 1^T = 1 \). The transition-probability matrix \( P \) is shown at the bottom of the next page. Being \( s(i) \) the number of served transactions from departure state \( i \), transition probabilities \( p_{i,j} \) and \( \overline{P}_{i,j} \) are computed as

\[
p_{i,j} = \frac{\lambda}{\lambda + \mu} (\frac{\mu}{\mu + \lambda})^{j-(i-s(i))}, \quad \text{and} \quad (3)
\]

\[
\overline{P}_{i,j} = 1 - \sum_{l=0}^{\infty} p_{i,l} \cdot \tau.
\]

To capture the effect of forks in the model, we consider the set \( T_f \) of non-overlapping transactions that are included in a forked block. These transactions need to be re-added to the queue, thus contributing to increasing its occupancy (so as the delay) by affecting to the transition probabilities shown above in Eq. (3) and (4). In particular, we define \( s(i) \) as

\[
s(i) = (1 - p_{fork}) \cdot \min(i, b) - p_{fork} \cdot |T_f|.
\]

Finally, the steady-state queue distribution \( \pi^k \) for \( k < K \) is

\[
\pi^k = \frac{1}{\mu 1_d} \sum_{i=0}^{k} \overline{P}_{i}^{j=K-s(i)+1} \sum_{j=k-s(i)+1}^{K} p_{i,j}.
\]

![Fig. 2](image)
where \(T_d\) is the expected inter-departure time, \(\varsigma_{t}^i\) is the timer expiration probability from departure state \(i\), and \(\Pr(n|\tau)\) is the probability that \(n\) packets arrive to the queue before the timer expires. The probability of finding the queue full is calculated as \(\pi^K = 1 - \sum_{k=0}^{K-1} \pi^k\).

### C. Optimal Block Size

We now derive the optimal block size \(b^*\) that minimizes the transaction confirmation time in a blockchain system. To that purpose, we use the latency model provided in Sect. III-A together with the following assumption.

**Assumption 1:** The waiting timeout is set to an arbitrarily large number so that the number of mined transactions is always set to the block size, \(b\).

The waiting timer breaks the convexity of the latency function by generating abrupt changes to it, as shown in Section IV through numerical results. When applying Assumption 1 to disregard the timeout, it is possible to find a convex function that is a close fit to data obtained from the model. In particular, we approximate \(T_q\) using the Lagrange interpolation and that is a close fit to data obtained from the queuing model. Assuming fixed \(\mu\) and \(\lambda\) parameters (provided by the scenario), we approximate \(T_q\) as

\[
T_q(b) \approx \hat{T}_q(b) = \sum_{j=0}^{N} y_j \mathcal{L}_{N,j}(b),
\]

where \(y_j\) is the actual queuing delay and \(\mathcal{L}_{N,j}(b)\) are the Lagrange basis polynomials, defined as \(\prod_{k=0, k \neq j}^{N} \frac{b-b_k}{b_j-b_k}\).

Being \(C \in \mathbb{R}^+\) a positive constant capturing the P2P end-to-end capacity, the overall transaction confirmation delay, \(T_{bp}\), is approximated as

\[
\hat{T}_{bp} = \frac{T_q(b) + 1/M \lambda + b/C}{e^{-\lambda(M-1)b/C}}.
\]

Following Eq. (8), which is convex for \(b \geq 0\), the optimal block size \(b^*\) can be derived directly. The accuracy of this approximation is discussed in Section IV.

### IV. Simulation Results and Validation

We now assess the accuracy of the proposed block size optimization model.\(^1\) We study situations where Assumption 1 above does not hold (i.e., with realistic timer and fork behaviors), which allows comparing the performance of the actual optimal block size with our proposed approximation. We also validate the proposed latency model through simulations [14]. Table I collects the parameter setup used for both the model and the simulator.

![Simulation Results and Validation](image)

**Fig. 3.** Transaction confirmation latency for different blockchain parameters: (a) \(\tau = 1\) s, (b) \(\tau = 100\) s. The delays associated to the estimated optimal block size \((b^*)\) are represented by the black circles.

\(^1\)All the source code and complementary results are open and can be found at [https://github.com/fwilhelm/blockchain_block_size_optimization](https://github.com/fwilhelm/blockchain_block_size_optimization)

| Parameter | Description | Value |
|-----------|-------------|-------|
| \(b\)     | block size | 1-10 trans. |
| \(t, h\)  | transaction/block header length | 5 kbits, 20 kbits |
| \(K\)     | queue size | 10 trans. |
| \(M\)     | number of miners | \{1,10\} |
| \(C\)     | P2P links’ capacity | 5 Mbps |
| \(t_s\)   | sim. time | 100,000 s |

**TABLE I**

Blockchain Parameters Used in Numerical Evaluation

- **Parameter**: \(b\)  
  - **Description**: block size  
  - **Value**: 1-10 trans.
- **Parameter**: \(t, h\)  
  - **Description**: transaction/block header length  
  - **Value**: 5 kbits, 20 kbits
- **Parameter**: \(K\)  
  - **Description**: queue size  
  - **Value**: 10 trans.
- **Parameter**: \(M\)  
  - **Description**: number of miners  
  - **Value**: \{1,10\}
- **Parameter**: \(C\)  
  - **Description**: P2P links’ capacity  
  - **Value**: 5 Mbps
- **Parameter**: \(t_s\)  
  - **Description**: sim. time  
  - **Value**: 100,000 s

---

\[\begin{array}{cccccccc}
0 & 1 & \ldots & b & \ldots & K-b & \ldots & K \\
0 & 0 & \ldots & b & \ldots & K-b & \ldots & K \\
& \vdots & & \vdots & & \vdots & & \vdots \\
K-b & 0 & \ldots & p_{K-b,b} & \ldots & \overline{p}_{b+1,K+1} & \ldots & 0 \\
K & 0 & \ldots & 1 & \ldots & 0 & \ldots & 0 \\
\end{array}\]

- **Expression**:
  - \[\begin{bmatrix}
  p_{0,0} & p_{0,1} & \ldots & p_{0,b} & \ldots & p_{0,K-b} & \ldots & \overline{p}_{0,K} \\
  p_{1,0} & p_{1,1} & \ldots & p_{1,b} & \ldots & p_{1,K-b} & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
  p_{b,0} & p_{b,1} & \ldots & p_{b,b} & \ldots & \overline{p}_{b,K-b} & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
  p_{K-b,0} & p_{K-b,1} & \ldots & p_{K-b,b} & \ldots & \overline{p}_{b+1,K+1} & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
  0 & 0 & \ldots & 1 & \ldots & 0 & \ldots & 0 \\
\end{bmatrix}\]
μ major differences occur for high values of arrival rate when forks are not possible. As shown next, the transformation of the model function may deviate from the simulator outputs, for \(M = 1\) without forks and \(M = 10\) with forks. In contrast, for \(\tau = 1\) s (left figure), the actual optimum is not fully achieved. Still, the approximation serves as a consistent heuristic and leads to near-optimal end-to-end latency.

Setting the timer \(\tau = 100\) s (right figure). In this case, Assumption 1 is fulfilled due to the high timer value (blocks are mined when reaching the established-set block size). In contrast, for \(\tau = 1\) s (left figure), the actual optimum is not fully achieved. Still, the approximation serves as a consistent heuristic and leads to near-optimal end-to-end latency.

To reinforce this statement, we compare the performance obtained by the actual optimal block size (computed by brute force using simulations) and its model approximation. As shown in Fig. 4, the highest gap is found when timers are high and forks are not possible. As shown next, the major differences occur for high values of \(\mu\). In contrast, when considering forks, the actual and approximated optimal block sizes lead to very close results. Despite the convex transformation of the model function may deviate from the actual queue latency, the estimated trend is the same, so the optimal block size can be estimated in most cases.

To conclude, Fig 5 compares the model and simulator outputs, for \(\mu = \{0.1, 0.25\}\), \(\lambda = \{0.25, 5\}\) Hz, \(\tau = \{1, 5, 100\}\) s, and \(b = [1, 10]\). Without forks (Fig. 5(a)), the model deviates from simulations if the queue is saturated with transactions (\(\mu = 0.25\) and \(\lambda = 0.25\), shown in green). For high block sizes, the model becomes unstable, given the high amount of incoming traffic. Notice that simulations consider a finite amount of time, so their results include the transitory phase before the queue is filled. Moreover, model and simulator results diverge for \(M = 10\) and low waiting timers (\(\tau = 1\) s and \(\tau = 5\) s). This confirms that the effects caused by forks cannot be fully captured by our model. The fork probability varies depending on the state from which a departure occurs, so the higher the number of transactions to be mined, the higher the fork probability. However, to capture this, it is required to have a-priori knowledge of the queue’s steady-state distribution, which is in fact the output of the model. Yet, the fork probability is estimated by the model based on the input block size \(b\), which is assumed to be fixed. This, however, does not hold for shorter timers, whereby partially-filled blocks can be mined before reaching the block size, thus leading to a lower fork probability. This effect is exacerbated as the timer and number of arrivals decrease.

V. Conclusion

To consolidate the adoption of blockchain in practical settings, in this letter, we have addressed the optimization of the block size. Adjusting the block size is essential to minimize the latency of PoW-based blockchain applications. However, the complex influence of timers and forks on the queuing delay prevents deriving the optimal block size in closed-form. Nevertheless, we have shown that an approximation of the queue delay function allows estimating the optimal block size, thus leading to near-optimal delay estimations. As future work, we envision to improve the queuing model through tailored data-driven approaches. Moreover, we will explore the joint optimization of the block size with other blockchain parameters (e.g., block generation rate), whereby further enhancements are expected.

REFERENCES

[1] J. Al-Jaroodi and N. Mohamed, “Blockchain in industries: A survey,” IEEE Access, vol. 7, pp. 36500–36515, 2019.
[2] D. C. Nguyen et al., “Federated learning meets blockchain in edge computing: Opportunities and challenges,” IEEE Internet Things J., vol. 8, no. 16, pp. 12806–12825, Aug. 2021.
[3] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, “Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles,” IEEE Trans. Veh. Technol., vol. 69, no. 4, pp. 4298–4311, Apr. 2020.
[4] M. Vukolić, “The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication,” in Proc. Int. Workshop Open Problems Netw. Secur., Zurich, Switzerland, 2015, pp. 112–125.
[5] M. Swan, Blockchain: Blueprint for a New Economy. Sebastopol, CA, USA: O’Reilly Media, 2015.
[6] H. Kim, J. Park, M. Bennis, and S.-L. Kim, “Blockchain- on-device federated learning,” IEEE Commun. Lett., vol. 24, no. 6, pp. 1279–1283, Jun. 2020.
[7] S. R. Pokhrel and J. Choi, “Federated learning with blockchain for autonomous vehicles: Analysis and design challenges,” IEEE Trans. Commun., vol. 68, no. 8, pp. 4734–4746, Aug. 2020.
[8] L. Feng, “BAFL: A blockchain-based asynchronous federated learning framework,” IEEE Trans. Comput., vol. 71, no. 5, pp. 1092–1103, 2021.
[9] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, “Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks,” IEEE Trans. Ind. Informat., vol. 17, no. 7, pp. 5098–5107, Jul. 2021.
[10] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, “Performance optimization for blockchain-enabled industrial Internet of Things (IIoT) systems: A deep reinforcement learning approach,” IEEE Trans. Ind. Informat., vol. 15, no. 6, pp. 3559–3570, Jun. 2019.
[11] R. C. Merkle, “A digital signature based on a conventional encryption function,” in Proc. Conf. Theory Appl. Cryptograph. Berlin, Germany: Springer, 1987, pp. 369–378.
[12] F. Wilhelmi and L. Giupponi, “Discrete-time analysis of wireless blockchain networks,” in Proc. PIMRC, Sep. 2021, pp. 1011–1017.
[13] M. Abramowizt and I. A. Stegun, Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables, vol. 55. Washington, DC, USA: United States Government Publishing Office, 1970.
[14] F. Wilhelmi and L. Giupponi, “Blockchain-oriented batch service queue simulator,” Comput. Softw., Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Barcelona, Spain, 2021, doi: 10.5281/zenodo.4680438.